MASTER EN SCIENCES INFORMATIQUES

Système expert fiscal

Dereppe, Stéphane

Award date:
2007

Système expert fiscal

Stéphane Dereppe

Promoteur : Prof. Jean-Marie Jacquet

Je tiens à remercier tout particulièrement mon promoteur, Monsieur J-M Jacquet, pour sa disponibilité et pour l'aide qu'il m'a apportée lors de la réalisation de ce mémoire.
Je remercie mon ami, F. Derzelle, pour l'aide qu'il m'a apportée de part ses connaissances en HTML/PHP.
Je remercie enfin ma famille ainsi que l'ensemble de mes amis pour le soutien qu'ils m'ont apporté durant l'ensemble de mes études.

Résumé

Dans ce mémoire, nous développons un système expert pour le calcul de l'impôt des personnes physiques en Belgique. Une interface Web est developpée afin de permettre aux utilisateurs d'encoder facilement leurs données personnelles. Après un rappel théorique sur la programmation logique avec contraintes, nous utilisons ce paradigme pour modéliser les lois fiscales nécessaires au calcul de l'impôt. Les règles modélisées seront interprétées par le moteur d'inférence GNU Prolog.

Abstract

In this report, we develop an expert system for the computation of the people's tax in Belgium. A Web interface is developed to allow users to easily encode their own data. After a theorical recall about the logic programming with constraints, we use this paradigm to model the tax laws needed to compute the tax. The rules will be interpreted by the GNU Prolog inference engine.

Table des matières

Chapitre 1

Introduction

On peut définir un système expert comme un outil capable de reproduire les mécanismes cognitifs d'un expert dans un domaine particulier[3]. En d'autres termes, il s'agit d'un logiciel capable de répondre à des questions en effectuant un raisonnement à partir de faits et de règles connus. Un système expert se compose de 3 parties :

- une base de faits ;
- une base de règles et
- un moteur d'inférence.

Le moteur d'inférence est capable d'utiliser faits et règles pour produire de nouveaux faits, jusqu'à parvenir à la réponse à la question experte posée.

Le but de ce mémoire est de développer un système expert pour la déclaration fiscale et plus particulièrement le calcul de l'impôt des personnes physiques en utilisant la programmation logique avec contraintes. Nous verrons en effet que ce paradigme de programmation est relativement bien adapté et permet de modéliser les règles fiscales de manière assez naturelle. Comme moteur d'inférence, nous utiliserons GNU Prolog, un compilateur Prolog libre possédant un solveur de contraintes pour des domaines finis développé par Daniel Diaz.

Après un rappel sur les notions élémentaires de la programmation (logique) sous contraintes, nous présenterons l'architecture générale du système expert avant de détailler plus précisement chaque composant :

- l'interface ;
- la base de données ;
- le programme exécutable et
- les règles fiscales modélisées sous forme de contraintes et prédicats Prolog.

Nous présenterons ensuite un exemple de calcul d'impôt avant de décrire les limites et les améliorations possibles du système expert construit. Pour terminer, nous comparerons brièvement l'application avec des travaux relatifs existants.

Chapitre 2

Programmation logique avec contraintes

2.1 Introduction

Fondée sur une base théorique très puissante à savoir la logique des prédicats du premier ordre (les clauses de Horn), la programmation logique constitue un paradigme de programmation simple et déclaratif qui a fait l'objet de beaucoup de recherches ces dernières années. En outre, elle est de plus en plus utilisée dans l'industrie pour résoudre des problèmes d'optimisation et de recherche combinatoire.

La principale différence entre les langages de programmation logique tel que Prolog et les langages de programmation classique comme Pascal est que ces derniers sont de nature impérative c'est à dire qu'il faut décrire la façon de résoudre le problème dans un algorithme alors que la programmation logique est de nature déclarative : il suffit d'indiquer au système les données du problème à traiter. Le moteur d'inférence, qui permet de mettre en relation ces données, rend alors toutes les solutions. Cela nous permet donc de résoudre une large gamme de problèmes relativement complexes sans devoir recourir à des techniques algorithmiques.

La plupart des langages de programmation avec contraintes sont des extensions de langages de programmation logique tel que Prolog qui remplacent l'unification effectuée par le moteur d'inférence par une vérification de la satisfaisabilité des contraintes en utilisant un solveur de contraintes dédié.

Comme exemple introductif de programme logique avec contraintes, prenons le problème crypto-arithmétique suivant :

$$\begin{array}{cccccc} & & S & E & N & D \\ + & & M & O & R & E \\ \hline = & M & O & N & E & Y \end{array}$$

où chaque lettre représente un chiffre différent et où l'équation arithmétique est correcte. Voici le programme logique avec contraintes qui permet de résoudre ce problème[1] :

```
1   sendmoremoney(S,E,N,D,M,O,R,Y):-
2       fd_domain([S,E,N,D,M,O,R,Y], 0, 9),
3       contraintes([S,E,N,D,M,O,R,Y]).
4
5   contraintes([S,E,N,D,M,O,R,Y]) :-
6       S #\= 0,
7       M #\= 0,
8       fd_all_different([S,E,N,D,M,O,R,Y]),
9       1000*S + 100*E + 10*N + D +
10      1000*M + 100*O + 10*R + E #=
11      10000*M + 1000*O + 100*N + 10*E + Y,
12      fd_labeling([S,E,N,D,M,O,R,Y]).
```

Etant donné le but `sendmoremoney([S,E,N,D,M,O,R,Y])`, le programme va déterminer rapidement que $S = 9$, $E = 5$, $N = 6$, $D = 7$, $M = 1$, $O = 0$, $R = 8$ et $Y = 2$.

Dans le premier prédicat `fd_domain()`, on déclare le domaine des variables alors que les contraintes sur ces variables sont regroupées dans le deuxième prédicat `contraintes()`. La fonction auxiliaire `fd_labeling()` permet d'invoquer un solveur de contraintes complet ce qui signifie qu'on garantit que le programme trouvera une solution si elle existe. La contrainte `fd_all_different()` permet de préciser que les variables doivent avoir des valeurs différentes.

Dans ce chapitre, nous allons brièvement rappeler quelques notions de la logique des prédicats et de programmation logique avec Prolog. Nous enchaînerons ensuite avec une définition formelle des contraintes et nous citerons quelques propriétés désirables pour celles-ci ainsi que pour des solveurs de contraintes. Nous terminerons en parlant de la programmation logique avec contraintes et en donnant un exemple un peu plus complexe que celui présenté ci-dessus.

2.2 La logique des prédicats : définitions

La logique des prédicats est un système formel reposant sur

- une syntaxe (un langage) ;
- une sémantique déclarative (une interprétation des formules)
- une sémantique procédurale (une théorie de preuve, soit un mode de dérivation de formules valides à partir d'autres formules)

[1] La syntaxe utilisée est celle de GNU Prolog

Les *symboles* utilisés sont :

- les constantes : jean, un, deux, ... ;
- les foncteurs (constructeurs) ayant une arité : endroit/2, prof/2, ... ;
- les prédicats (symboles de relations) ayant une arité : père/2, frère/2, ... ;
- les variables : X, $Cours$, ... ;
- les quantificateurs : $\forall$, $\exists$;
- les connecteurs : $\wedge$, $\vee$, $\neg$, $\Rightarrow$, $\Leftarrow$, $\Leftrightarrow$;
- les symboles auxiliaires : (,).

On appelle **alphabet** l'ensemble des constantes, variables, foncteurs et prédicats utilisés.

Nous allons maintenant définir formellement plusieurs notions qui nous seront utiles par la suite pour expliquer le mécanisme de fonctionnement de Prolog.

Définition* 1** *Un objet dans l'univers est représenté par un* ***terme*. L'ensemble des termes comprend les constantes, les variables et les termes composés c'est à dire de la forme*

$$f(t_1, t_2, \ldots, t_m)$$

où f est un foncteur d'arité m et les t_i sont des termes.

Définition* 2** *Un* ***atome *(ou formule atomique) est une formule de la forme*

$$p(t_1, t_2, \ldots, t_m)$$

où p est un prédicat d'arité m et les t_i sont des termes.

Définition* 3** *Une* ***formule bien formée *est une formule atomique. De plus si F et G sont des formules bien formées, alors $\neg F$, $F \wedge G$, $F \vee G$, $F \Rightarrow G$, $F \Leftarrow G$, $F \Leftrightarrow G$, $\forall X\ F$, $\exists X\ F$, (F) où X est une variable qui figure dans F, sont également des formules bien formées.*

Les formules suivantes sont des formules bien formées.

$$\forall X \forall Y\, mère(X) \wedge parent(X, Y) \Rightarrow aime(X, Y)$$
$$\forall X oiseau(X) \wedge \neg vole(X) \Rightarrow manchot(X)$$

Par contre la formule suivante n'est pas bien formée :

$$\exists X (mère(X)\ \forall X \wedge aime(X, Y)).$$

Définition* 4** *Une variable est* ***liée *dans une formule bien fondée si toute occurence de cette variable est sous la portée d'une quantification. Elle est* ***libre*** *sinon.*

Par exemple, dans la formule suivante, X est une variable liée et Y est une variable libre.

$$\forall X p(X, Y)$$

Définition* 5** *Une formule est* ***fermée *si toute variable y apparaissant est liée i.e il n'y a pas de variable libre.*

Définition* 6** *Un terme ou une formule est* ***clos(e) *si il(elle) ne contient aucune variable.*

Définition* 7** *Un* ***domaine*, noté $\mathcal{D}$, comprend l'ensemble des objets dont la théorie est censée parler.*

Définition* 8** *Une* ***interprétation*, notée $\mathcal{I}$, d'un alphabet A est composée d'un domaine non-vide $\mathcal{D}$ et d'une fonction qui associe*

- *à chaque constante $c \in A$ un élément $c_{\mathcal{I}} \in \mathcal{D}$*
- *à chaque foncteur $f/n \in A$ une fonction $f_{\mathcal{I}} : \mathcal{D}^n \to \mathcal{D}$*
- *à chaque prédicat $p/n \in A$ une relation $p_{\mathcal{I}} \subseteq \mathcal{D}^n$*

où $\mathcal{D}^n$ représente le n-produit cartésien $\mathcal{D} \times \ldots \times \mathcal{D}$.

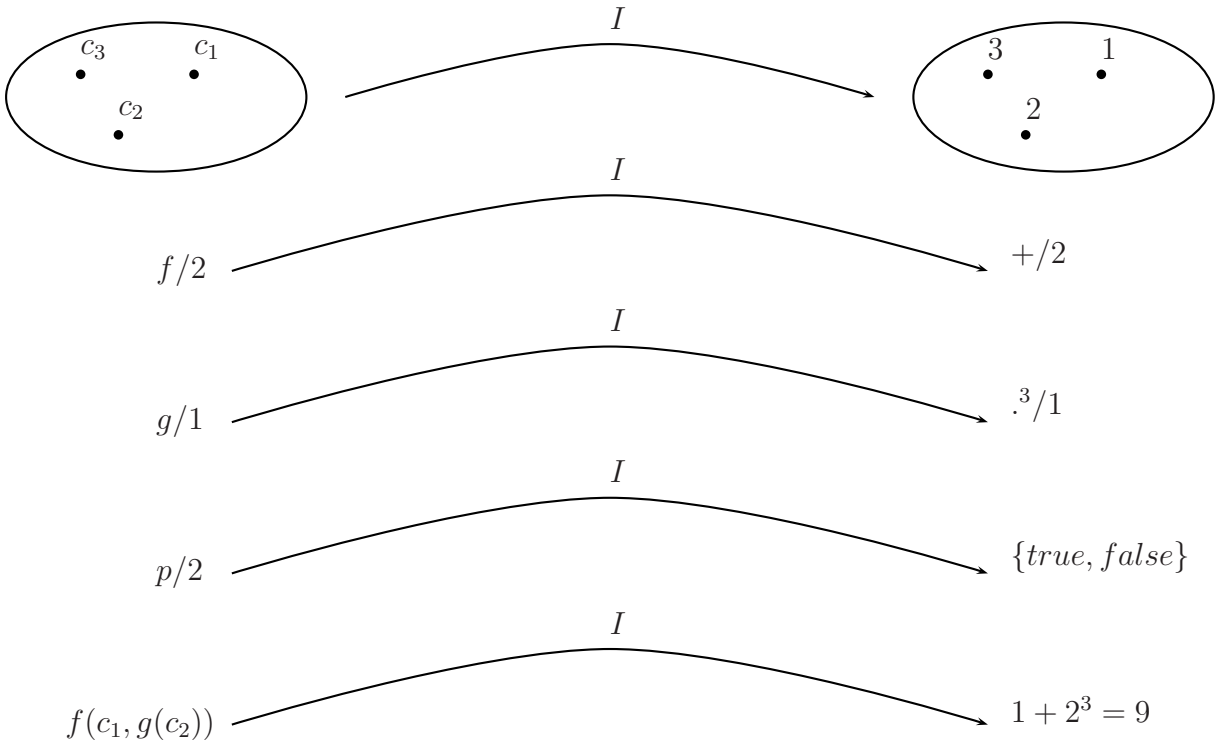

FIG. 2.1 – Exemple d'interprétation

Dans la figure 2.1, les entiers 1,2 et 3 sont associés aux constantes c_1, c_2 et c_3. Les opérations d'addition et d'élévation au cube sont associées respectivement au foncteur $f/2$ et $g/1$ alors Le prédicat $p/2$ prendra les valeurs $true$ ou $false$ une fois qu'il aura été évalué.

Autrement dit, une interprétation associe un objet de l'univers à chaque terme clos et une valeur de vérité (vraie ou fausse) à chaque atome.

Par exemple, l'atome *mère(X)* est vrai si le terme X (qui représente une personne grâce à une interprétation (voir définition 2.1)) est une mère.

Définition* 9** *Soit P un ensemble de formules bien formées fermées. On appelle* ***modèle *de P une interprétation $\mathcal{I}$ telle que chaque formule de P est vraie par rapport à $\mathcal{I}$.*

Définition* 10** *Un ensemble de formules qui ne possède pas de modèle est dit* ***insatisfaisable*.*

Définition* 11** *Soit P un ensemble de formules bien formées fermées et F une formule fermée. On appelle F* ***conséquence logique *de P, noté*

$$P \models F$$

si et seulement si F est vrai dans chaque modèle de P.

Notons également que

$$P \models F \textit{si et seulement si } P \cup \{\neg F\} \textit{ est insatisfaisable.} \tag{2.1}$$

Définition* 12** *On appelle* ***règle d'inférence *une règle qui permet de dériver une nouvelle formule (la conclusion) à partir d'un ensemble de formules données (les prémisses).*

Citons comme exemple de règles d'inférence :

- le modus ponens
$$\frac{F, \quad F \Rightarrow G}{G}$$
- le modus tollens
$$\frac{F \Rightarrow G \quad , \quad \neg G}{\neg F}$$

Définition* 13** *Soit R un ensemble de règles d'inférence et soit P un ensemble de formules fermées. Une formule F est dite* ***dérivable *de P par R, noté*

$$P \vdash_R F$$

si on peut obtenir F à partir de P en n'utilisant que les règles de R.

Définition* 14** *Une* ***clause *est une formule de la forme*

$$\forall(A_1 \vee \ldots \vee A_m \vee \neg A_{m+1} \vee \ldots \vee \neg A_{m+n})$$

où $A_1, \ldots, A_m, A_{m+1}, \ldots, A_{m+n}$ *sont des atomes et* $m, n \geq 0$

Remarquons immédiatemment que,

$$\begin{gathered}(A_1 \vee \ldots \vee A_m \vee \neg A_{m+1} \vee \ldots \vee \neg A_{m+n}) \\ \Leftrightarrow \\ (A_1 \vee \ldots \vee A_m) \vee \neg(A_{m+1} \wedge \ldots \wedge A_{m+n}) \\ \Leftrightarrow \\ (A_1 \vee \ldots \vee A_m) \Leftarrow (A_{m+1} \wedge \ldots \wedge A_{m+n)}\end{gathered} \tag{2.2}$$

Définition* 15** *Une* ***règle *est une clause dont m=1 et n > 0 :*

$$\forall(A_1 \Leftarrow A_2 \wedge \ldots A_{1+n})$$

Définition* 16** *Un* ***fait *est une clause dont m=1 et n=0 :*

$$\forall A_1$$

Définition* 17** *Un* ***but *est une clause dont m=0 et n≥0 :*

$$\forall(\neg(A_1 \wedge \ldots \wedge A_n))$$

Définition* 18** *On appelle* ***but vide *un but dont m=0 et n=0. Le but vide est noté* $\square$.

Définition* 19** *Un* ***programme défini P *est un ensemble fini de faits et de règles.*

Afin d'illustrer ces dernières définitions, considérons l'exemple de programme défini suivant :

$$fier(X) \Leftarrow parent(X, Y), nouveau\_né(Y) \tag{2.3}$$

$$parent(mathieu, marie) \tag{2.4}$$

$$nouveau\_né(marie) \tag{2.5}$$

L'équation 2.3 est une *règle* alors que les équations 2.4 et 2.5 sont des *faits*.

Définition* 20** *Une* ***clause de Horn *est une clause contenant au plus un atome non nié i.e. m≤1.*

2.3 Substitutions, unifications et dérivations SLD

En partant d'un ensemble fini de clauses P et d'une requête Q, pour prouver que Q est une conséquence logique de P on utilise la relation 2.1 :

$$\begin{gathered} P \models Q \\ \Leftrightarrow \\ P \cup \{\neg Q\} \textit{ est insatisfaisable} \\ \Leftrightarrow \\ P \cup \{\neg Q\} \models \square \end{gathered}$$

Autrement dit, nous devons utiliser une règle d'inférence R pour dériver le but vide $\square$ à partir du but initial $\neg Q$ et de l'ensemble de règles et de faits de P.

$$P \cup \{\neg Q\} \vdash_R \square$$

La règle d'inférence utilisée sera le modus tollens généralisé :

$$\frac{\neg(p_1 \wedge \ldots \wedge p_i \wedge \ldots \wedge p_n), \quad p_i \leftarrow q_1 \wedge \ldots \wedge q_k}{\neg(p_1 \wedge \ldots \wedge q_1 \wedge \ldots \wedge q_k \wedge \ldots \wedge p_n)} \tag{2.6}$$

Nous devons néanmoins trouver un moyen de traiter les variables présentes dans les clauses. C'est dans ce but que nous allons introduire les notions de substitution et d'unification.

2.3.1 Substitution

Une **substitution** est un ensemble $\sigma = \{X_1/t_1, \ldots, X_n/t_n\}$ tel que

- $X_1, \ldots, X_n$ sont des variables ;
- $t_1, \ldots, t_n$ sont des termes ;
- $X_i \neq X_j, \ \forall i, j \in \{1, \ldots, n\} | i \neq j$,
- $X_i \neq t_i, \ \forall i \in \{1, \ldots, n\}$.

Soit $\sigma = \{X_1/t_1, \ldots, X_n/t_n\}$ une substitution. On appelle **domaine** de σ, noté $dom(\sigma)$, l'ensemble de variables $\{X_1, \ldots, X_n\}$. On appelle **codomaine** de σ, noté $codom(\sigma)$, l'ensemble des variables $\{Y_1, \ldots, Y_m\}$ qui figurent dans les termes $t_1, \ldots, t_n$.
Soit t un terme. L'**application** de σ à t, notée $t\sigma$ est le nouveau terme obtenu en remplaçant toute occurence de X_i par t_i pour tout $i \in \{1, \ldots, n\}$.

Voici un exemple de substitution :

$$\sigma = \{X/f(1, Y), Y/2, Z/g(X)\}$$

Si nous appliquons cette substitution σ au terme $t = p(h(X, Y), Z, g(Z))$, nous obtenons

$$t\sigma = p(h(f(1, Y), 2), g(X), g(g(X)))$$

Soient $\alpha = \{X_1/t_1, \ldots, X_m/t_m\}$ et $\beta = \{Y_1/u_1, \ldots, Y_n/u_n\}$ 2 substitutions. La **composition** de α par β, notée $\beta \circ \alpha$ ou $\alpha\beta$ est la substitution obtenue à partir de

$$\{X_1/t_1\beta, \ldots, X_m/t_m\beta\} \cup \{Y_1/u_1, \ldots, Y_n/u_n\}$$

en supprimant

- tous les $X_i/t_i\beta$ tels que $X_i = t_i\beta$;
- tous les Y_j/u_j tels que $Y_j \in dom(\alpha) = \{X_1, \ldots, X_m\}$

Par exemple, si $\alpha = \{X/Y, Y/Z\}$ et $\beta = \{X/3, Z/Y, T/4\}$ sont deux substitutions, alors la composition de α et de β donne

$$\alpha\beta = \{X/Y, Z/Y, T/4\}.$$

2.3.2 Unification

Définissons une **structure** comme un atome ou un terme. Soient 2 structures s et s'. Alors, on appelle **unificateur** de s et s' une substitution θ telle que $s\theta = s'\theta$. Notons que si deux structures s'unifient, il existe souvent plus d'un unificateur. Cela nous amène à la notion d'unificateur le plus général (noté mgu[2]).

On appelle **unificateur le plus général** de s et s' un unificateur θ tel que pour tout autre unificateur σ, il existe une substitution γ telle que

$$\sigma = \theta\gamma$$

Nous allons maintenant développer une méthode pour permettre de trouver de manière systématique l'unificateur le plus général de deux structures. Cette méthode est appelée **algorithme de Herbrand**.

Voici quelques définitions et notions indispensables à la compréhension de l'algorithme :

- On appelle **système d'équations** un ensemble

  $$\{s_1 = t_1, \ldots, s_n = t_n\}$$

  où s_i et t_i sont des structures pour tout $i \in \{1, \ldots, n\}$.
- On appelle **unificateur d'un tel système** un unificateur θ tel que

  $$s_i\theta = t_i\theta \text{ pour toute égalité } s_i = t_i \text{ telle que } i \in \{1, \ldots, n\}$$

  .

[2]Most General Unifier

- On dit que deux systèmes d'équations sont **équivalents** si et seulement si ils ont les mêmes unificateurs.
- Un système d'équations est en **forme résolue** si et seulement si il a la forme

$$\{X_1 = t_1, \ldots, X_n = t_n\}$$

où les X_i sont des variables disctinctes qui ne figurent dans aucun t_i.
- Soit un système d'équations en forme résolue

$$\mathcal{E} = \{X_1 = t_1, \ldots, X_n = t_n\}$$

alors la substitution

$$\theta = \{X_1/t_1, \ldots, X_n/t_n\}$$

est un **mgu** de $\mathcal{E}$

Etant donné deux structures s et s', le but de l'algorithme de Herbrand va être de transformer le système d'équations

$$\mathcal{E} = \{s = s'\}$$

en un système d'équations en forme résolue $\mathcal{E}'$ équivalent à $\mathcal{E}$.

L'algorithme va donc prendre en entrée un système d'équations $\mathcal{E}$ et fournira en sortie :

- un autre système d'équations équivallent et en forme résolue si il existe un unificateur pour $\mathcal{E}$;
- **échec** sinon.

Algorithme de Herbrand

repeat
  sélectionner une équation $s = t \in \mathcal{E}$
  case $s = t$ a la forme
    - $f(s_1, \ldots, s_n) = g(t_1, \ldots, t_k) \Rightarrow$ arrêt avec **échec**
    - $f(s_1, \ldots, s_n) = f(t_1, \ldots, t_k) \Rightarrow \mathcal{E} \leftarrow (\mathcal{E} \backslash \{s = t\}) \cup \{s_i = t_i\} \ \forall i$
    - $X = X \Rightarrow \mathcal{E} \leftarrow (\mathcal{E} \backslash \{X = X\})$
    - $t = X \Rightarrow \mathcal{E} \leftarrow (\mathcal{E} \backslash \{t = X\}) \cup \{X = t\}$
    - $X = t$ où $t \neq X$
      $\Rightarrow$ si X est un sous-terme de t, alors arrêt avec **échec**
        sinon, remplacer dans $\mathcal{E}$ chaque occurence de X par t
until il ne reste aucun moyen de réécrire une équation en $\mathcal{E}$
arrêter avec $\mathcal{E}$

Si l'algorithme se termine en produisant un système d'équations, celui-ci est en forme résolue $\{X_1 = t_1, \ldots, X_n = t_n\}$ et on obtient l'unificateur le plus général $\{X_1/t_1, \ldots, X_n/t_n\}$.

2.3.3 Dérivation SLD

Si nous repartons de l'équation de la règle d'inférence (2.6), on obtient :

$$\frac{\forall\neg(A_1 \wedge \ldots \wedge A_i \wedge \ldots \wedge A_n),\ \forall(B_0 \leftarrow B_1 \wedge \ldots \wedge B_k)}{\forall\neg(A_1 \wedge \ldots \wedge A_{i-1} \wedge B_1 \wedge \ldots \wedge B_k \wedge A_{i+1} \wedge \ldots \wedge A_n)\theta}$$

où

- A_i s'appelle l'**atome sélectionné** dans le but $\forall\neg(A_1 \wedge \ldots \wedge A_n)$;
- $\forall(B_0 \leftarrow B_1 \wedge \ldots \wedge B_k)$ est une clause de P ;
- $\theta = mgu(A_i, B_0)$.

La réduction progressive d'une requête à une autre s'appelle la **résolution SLD**. Une **dérivation SLD** est une suite de requêtes et de *mgus*

$$G \xrightarrow{\theta_1} G_1 \xrightarrow{\theta_2} \ldots \xrightarrow{\theta_n} G_n$$

On appelle **réfutation SLD** une dérivation SLD dont $G_n = \square$. La **réponse calculée** de la réfutation est la composition des *mgus* $\theta_1\theta_2 \ldots \theta_n$ limitée aux variables de G.
Remarquons qu'une dérivation SLD comprend 2 choix :

1. Dans chaque requête non-vide, il faut choisir un seul atome à résoudre.
   → choix d'une **règle de sélection des atomes**
2. Lorque l'atome sélectionné s'unifie avec la tête de plusieurs clauses, il faut en choisir une. C'est ce choix qui déterminera la solution (il peut y avoir plusieurs solutions).
   → choix d'une **règle de sélection des clauses**

Notons également qu'une dérivation SLD peut échouer ou être infinie. Si elle échoue, c'est que $G_n \neq \square$.

2.4 Prolog

Maintenant que nous avons rappelé les notions nécessaires, nous allons présenter le mécanisme d'exécution de Prolog.

Tout programme Prolog est un ensemble de clauses de Horn et chaque requête adressée au moteur d'inférence est une question existentielle. Prolog utilise la résolution SLD en choisissant comme règle de sélection des atomes la **leftmost selection** (c'est à dire l'atome le plus à gauche) et comme ordre des clauses, l'ordre de celles-ci dans le programme (c'est à dire du haut vers le bas).

Lorsqu'on introduit une requête, la résolution s'arrête

- quand on obtient la requête vide ($\square$) : le système retourne alors la réponse calculée et l'utilisateur peut forcer la recherche d'une (éventuelle) réponse suivante (backtracking) ;

- quand l'atome sélectionné ne s'unifie avec aucune clause : le système revient à un point de choix[3] et continue la résolution. Si il n'y a plus de point de choix, le système retourne la réponse **no**.

Il est donc évident que l'ordre des clauses et des atomes dans une clause a une grande importance. L'ordre des clauses détermine principalement l'ordre dans lequel les réponses seront trouvées alors que l'ordre des atomes dans une clause détermine surtout l'efficacité de la recherche de réponse. Dans le cas où une résolution SLD infinie est possible, l'ordre choisi peut également déterminer si une réponse sera trouvée.

2.5 Contraintes

Dans cette section, nous allons formaliser le concept de contrainte, définir différents types de contraintes ainsi que les opérations impliquant des contraintes à savoir :

- la satisfaisablilité
- l'optimisation

2.5.1 Contraintes et valuation

Les contraintes sont des formalisations mathématiques de relations qui existent entre des objets du monde réel dans un domaine particulier. Par exemple, la relation

$$X = Y + 2 \tag{2.7}$$

est une contrainte qui possède 2 *variables* (X et Y). La variable X pourrait représenter l'âge de l'aîné d'une famille alors que la variable Y représenterait l'âge de sa petite soeur. La contrainte (2.7) représente alors le fait que l'aîné a 2 ans de plus que sa soeur cadette. Cette contrainte n'est pas seulement vraie ou fausse, la valeur de vérité dépend des valeurs par lesquelles nous allons remplacer X et Y. Par conséquent, dès que nous sommes en présence de contraintes qui contiennent des variables, nous devons nous demander si il existe une substitution qui permet de rendre l'assertion vraie. Dans notre exemple, c'est évidemment le cas (prendre $X = 15$ et $Y = 13$). Dans d'autres situations, la réponse pourra être *non* ou bien encore *je ne sais pas.*

Comme nous l'avons déjà vu, les contraintes sont formées de *variables* (souvent représentées par des lettres majuscules) mais également de symboles fonctionnels ($+$, $-$, $*$, ...), de symboles relationnels ($=$, $\leq$, $\geq$, $<$, $>$, ...) et de constantes (0,2,-4, ...).

La forme des contraintes ainsi que leur signification sont spécifiées dans un **domaine de contraintes** $\mathcal{D}$. En d'autres termes, $\mathcal{D}$ détermine la syntaxe des contraintes et spécifie les

[3]point où l'atome s'unifiait avec plusieurs clauses et où le système a dû faire un choix

règles à suivre pour créer des contraintes dans ce domaine. Par exemple, si nous choisissons le domaine des nombres réels, nous obtenons comme ensemble de symboles fonctionnels $+$, $*$, $-$ et $/$ alors que l'ensemble des constantes est formé de tous les nombres en virgule flottante. L'ensemble des symboles relationnels est quant à lui constitué des relations $=$, $<$, $\leq$, $>$ et $\geq$ qui prennent tous 2 arguments. Le domaine de contrainte détermine également la sémantique de tous ces symboles. Par exemple, la contrainte $5 < 3$ n'est pas vérifiée alors que la contrainte $1 < 8$ est vérifiée.

A partir d'un domaine de contraintes $\mathcal{D}$, nous pouvons définir la notion de contrainte primitive. Une **contrainte primitive** est une contrainte avec un seul symbole relationnel de $\mathcal{D}$ et le nombre correct d'arguments. Ces arguments sont construits à partir des constantes et des symboles fonctionnels de $\mathcal{D}$ et de variables. Notons que la plupart des relations sont binaires i.e. elles nécessitent 2 arguments.

Généralement, les contraintes sont juste des conjonctions de contraintes primitives i.e. de la forme $c_1 \wedge \ldots \wedge c_n$ avec $n \geq 0$. Le symbole $\wedge$ désigne la conjonction. Par conséquent, une contrainte $c_1 \wedge \ldots \wedge c_n$ est vérifiée si chacune des contraintes primitives $c_1, \ldots, c_n$ est vérifiée. La contrainte vide ($n = 0$) est vérifiée. On distingue deux contraintes particulières : **vraie** et **fausse**. La contrainte vraie est toujours vérifiée alors que la contrainte fausse n'est jamais vérifiée.

Etant donné une contrainte, nous pouvons déterminer la valeur des variables pour laquelle la contrainte est vérifiée. En remplaçant chaque variable par une valeur, nous pouvons transformer la contrainte initiale en contrainte sans variable et donc évaluer la véracité de celle-ci. Par exemple, pour la contrainte

$$Y = 2X + Z \ \wedge \ X = 3A, \tag{2.8}$$

l'assignation des valeurs $20, 9, 2, 3$ respectivement aux variables Y, X, Z, A nous donne la contrainte $20 = 2 * 9 + 2 \ \wedge \ 9 = 3 * 3$ qui après simplification s'avère être vraie. Par contre, l'assignation des valeurs $20, 9, 2, 5$ respectivement aux variables Y, X, Z, A donne la contrainte $20 = 2 * 9 + 2 \ \wedge \ 9 = 3 * 5$ qui est fausse.

On définit une **valuation** θ pour un ensemble de variables V comme une assignation de valeurs du domaine de contraintes aux variables de V. Si $V = \{X_1, \ldots, X_n\}$ alors la valuation $\theta = \{X_1 \mapsto d_1, \ldots, X_n \mapsto d_n\}$ signifie que chaque X_i reçoit la valeur d_i.

Soit C une contrainte, définissons

- $var(C)$ = l'ensemble des variables apparaissant dans la contrainte C ;
- $\theta(C)$ = la contrainte obtenue en appliquant la valuation θ à la contrainte C.

Si θ est une valuation pour l'ensemble de variables V telle que $var(C) \subseteq V$, alors θ est une solution de la contrainte C si $\theta(C)$ est vraie dans le domaine de contraintes.
On dit qu'une contrainte est **satisfaisable** si elle possède une solution. Dans le cas contraire,

on dit qu'elle est **insatisfaisable**. Par exemple, nous savons déjà que la contrainte (2.8) possède une solution, elle est dont satisfaisable. Par contre, une contrainte du type

$$X \leq 3 \ \wedge \ X = Y \ \wedge Y > 4$$

ne possède pas de solution ; elle est donc insatisfaisable.

Nous considérons les contraintes comme des chaînes de caractères, des suites syntaxiques. Par conséquent, $X = Y$ et $Y = X$ sont des contraintes différentes. Malgré cela, on remarque que ces deux contraintes contiennent la même information. C'est pourquoi nous définissons l'équivalence entre deux contraintes.
Deux contraintes C_1 et C_2 sont **équivalentes** (notation : $C_1 \leftrightarrow C_2$) si elles ont le même ensemble de solutions.

2.5.2 Satisfaisabilité des contraintes

Etant donné une contrainte, une des questions les plus naturelles que nous pouvons nous poser est "Quelles sont les solutions de cette contraintes ?". Nous pouvons également nous demander si il *existe* une solution : c'est ce qu'on appelle le problème de la satifaisabilité de la contrainte. Evidemment, la satisfaisabilité est une question plus basique dans le sens où si on trouve une solution, on connait également la réponse au problème de satisfaisabilité.

On appelle **solveur de contraintes** un algorithme qui permet de répondre au problème de satisfaisabilité. Comme nous le verrons, la plupart des solveurs permettent également de trouver une solution.

La manière la plus évidente de résoudre le problème de satisfaisabilité est d'énumérer toutes les valuations possibles pour la contrainte et de tester si on obtient une solution. L'inconvénient de cette méthode est que, dans la plupart des cas, il peut y avoir une infinité de valuations à tester sans pour autant arriver à une solution. Par exemple, dans le cas de la contrainte $X > Y$ où X et Y sont des nombres naturels, si nous désirons évaluer chaque valuation possible dans l'ordre suivant, nous n'atteindrons jamais une solution.

$$\begin{array}{ll} \{X \mapsto 1, Y \mapsto 1\} & false \\ \{X \mapsto 1, Y \mapsto 2\} & false \\ \{X \mapsto 1, Y \mapsto 3\} & false \\ \vdots & \end{array}$$

Par contre si nous testons les valuations dans un autre ordre, nous pouvons trouver une solution après un certain nombre d'essais.
Le problème avec les méthodes basées sur l'énumération des valuations est qu'elles utilisent les contraintes de manière passive[4]. En effet, elles testent le résultat d'une valuation sur

[4]De plus, il est impossible d'utiliser de telles méthodes avec des contraintes sur des nombres réels car il est impossible d'énumérer toutes les valuations possibles.

les contraintes à la place d'utiliser ces dernières pour construire une solution. Notons que la manière dont on utilise les contraintes pour construire une solution dépend fortement du type de contrainte que l'on veut traiter, c'est à dire de son domaine. Un des domaines les plus utilisés dans l'industrie est le domaine de l'arithmétique linéaire.

Une **expression linéaire** est de la forme $a_1x_1 + a_2x_2 + \ldots + a_nx_n$ où a_i est un nombre ($i \in \{1, \ldots, n\}$) et chaque x_i est une variable ($i \in \{1, \ldots, n\}$).
Une **équation linéaire** est de la forme $e_1 = e_2$ avec e_1 et e_2 qui sont des expressions linéaires.
Une **inéquation linéaire** est de la forme $e_1 < e_2$, $e_1 > e_2$, $e_1 \leq e_2$, $e_1 \geq e_2$ avec e_1 et e_2 qui sont des expressions linéaires.
Une **contrainte linéaire** est une conjonction d'équations et inéquations linéaires.

Avec ce type de domaine, le solveur utilise l'*éliminiation de Gauss-Jordan*. La principale opération de l'algorithme de Gauss-Jordan consiste à prendre une équation c, de la transformer en une équation équivalente de la forme $x = e$ où x est une variable et e une expression linéaire qui ne contient pas x, puis de remplacer dans toutes les autres équations chaque occurence de x par e. On répète cette opération jusqu'à ce qu'on trouve une solution ou jusqu'à ce qu'on ait la preuve qu'il n'existe pas de solution.

Il existe bien d'autres domaines de contraintes dont nous ne parlerons pas ici. Citons par exemple les contraintes booléennes, les contraintes de séquences, les contraintes sur les arbres, ... Par contre, nous reviendrons plus tard sur le domaine qui va nous interesser dans le cadre de ce mémoire à savoir le domaine des contraintes finies (voir section 2.5.4).

2.5.3 Propriétés des solveurs de contraintes

Dans cette section, nous présentons quelques propriétés dont peuvent profiter les solveurs de contraintes. La plus importante d'entre elles étant la complétude.

Un solveur **complet** est capable de déterminer si une contrainte est satisfaisable (true) ou insatisfaisable (false). En général, peu de solveurs vérifient cette propriété car la complexité de l'algorithme du solveur deviendrait trop grande. C'est pourquoi, nous utiliserons la plupart du temps des solveurs **incomplets** qui, à une question de satisfaisabilité, peuvent répondre **unknow**. Nous pouvons maintenant donner une définition formelle de solveur.

Un **solveur** *solv* pour un domaine de contraintes $\mathcal{D}$ prend en entrée n'importe quelle contrainte C dans $\mathcal{D}$ et retourne les valeurs *true*, *false* ou *unknow*. Lorque $solv(C)$ retourne *true*, la contrainte C doit être satisfaisable et lorsque $solv(C)$ retourne $false$, la contrainte C doit être insatisfaisable. Le solveur de contraintes doit répondre correctement à la question de satisfaisabilité ou bien répondre *unknow*[5].

[5]Cette définition autorise des solveurs très faibles. En effet, n'importe quelle fonction renvoyant *unknow*

Une autre propriété désirable pour un solveur est qu'il devrait toujours donner la même réponse pour une conjonction de contraintes primitives et ce, peu importe leur ordre. En outre, si une contrainte C_1 est insatisfaisable, alors chaque conjonction de C_1 avec une autre contrainte C_2 doit aussi être insatisfaisable. Cette propriété est appelée **monotonie**. Nous voudrions également que le résultat obtenu ne dépende pas du nom des variables.

Si le solveur satisfait ces 3 dernières propriétés, on dit qu'il est **bien-fondé**.

2.5.4 Domaine des contraintes finies

Dans cette section, nous nous intéressons à un des domaines de contraintes les plus importants : le domaine des contraintes finies. Il s'agit de contraintes dans lesquelles les valeurs possibles qu'une variable puisse prendre est restreint à un ensemble fini. Le domaine des contraintes booléennes en est un cas particulier car les seules valeurs que peuvent prendre les variables sont 0 ou 1. Un autre exemple de domaine de contraintes finies est le domaine des entiers dans lequel chaque variable est contenue dans un intervalle fini.

Le domaine des contraintes finies est beaucoup utilisé en programmation avec contraintes car il permet au programmeur de modéliser de manière naturelle une grande variété de problèmes. Par exemple, supposons que nous voulions colorer la carte de l'Australie présentée à la figure 2.2 avec 3 couleurs différentes : rouge, jaune et bleu. Comme nous pouvons le voir, il y a 7 états[6] et nous souhaitons colorer chacun de ces états de tel sorte que tous les états adjacents aient des couleurs différentes. C'est bien un exemple de problème avec contraintes finies car la couleur de chaque région appartient à l'ensemble fini $\{rouge, jaune, bleu\}$.

FIG. 2.2 – Carte de l'Australie

lorsqu'elle reçoit en entrée une contrainte est un solveur.

[6]Ces états sont New South Wales, Northern Territory, Queensland, South Australia, Tasmania, Victoria et Western Australia

Le domaine des contraintes finies permet en outre de résoudre beaucoup de problèmes de la vie courante, notamment des problèmes d'organisation d'horaires et de routages. Trouver des solutions à ce type de problème est d'une importance capitale pour certaines entreprises, d'où l'intérêt croissant pour ce type de domaine de contraintes.

Un **problème de satisfaisabilité de contraintes (CSP)**[7] (dans le domaine des contraintes finies) consiste en

- une *contrainte* C sur les variables $x_1, \ldots, x_n$ et
- un *domaine* D qui fait correspondre chaque variable x_i à un ensemble fini de valeurs, noté $D(x_i)$, qu'elle peut prendre.

La contrainte devient donc $C \wedge x_1 \in D(x_1) \wedge \ldots \wedge x_n \in D(x_n)$.

Il est toujours possible de déterminer la satisfaisabilité d'un CSP par une recherche systématique de toutes les combinaisons possibles des différentes valeurs à partir du moment où le nombre de combinaisons possibles est fini. Cependant, étant donné que la résolution de CSP est un problème *NP-complet*, chaque solveur complet aura une complexité exponentielle. Nous allons maintenant présenter un tel solveur. Par la suite, nous présenterons des solveurs incomplets mais qui ont l'avantage de posséder une complexité polynomiale.

Solveur complet (backtracking)

La manière la plus simple de déterminer la satisfaisabilité d'un CSP est le **backtracking chronologique**. Le principe d'une telle méthode est de choisir une variable et, pour chaque valeur possible de cette variable, de vérifier la satisfaisabilité de la contrainte en remplaçant la variable par cette valeur. En appelant cet algorithme de manière récursive, nous pouvons déterminer si la contrainte est satisfaisable ou pas.

Par exemple, si nous avons la contrainte suivante : $X < Y \wedge Y < Z$ avec les variables X, Y et Z ayant pour domaine $\{1, 2\}$, nous commençons par itérer sur les valeurs de X. Nous donnons à X la valeur 1 et nous obtenons la contrainte $1 < Y \wedge Y < Z$. Nous ne pouvons pas conclure que cette contrainte est insatisfaisable. Nous affirmons donc qu'elle est **partiellement satisfaisable**. Nous procédons ensuite à un appel récursif de l'algorithme sur la contrainte résultante.
Dans cet appel récursif, nous travaillons sur la variable Y et nous lui donnons la valeur 1. La contrainte obtenue $(1 < 1 \wedge 1 < Z)$ n'est pas partiellement satisfaisable. Nous choisissons donc une nouvelle valeur pour Y à savoir 2. La contrainte $1 < 2 \wedge 2 < Z$ est partiellement satisfaisable car chaque contrainte primitive sans variable est satisfaisable. Nous appelons donc de manière récursive l'algorithme sur cette contrainte.
Nous donnons à Z la valeur 1 ce qui nous donne une contrainte qui n'est pas partiellement satisfaisable : $1 < 2 \wedge 2 < 1$. La prochaine itération attribue la valeur 2 à la variable Z. La

[7] Constraint satisfaction problem

contrainte résultante n'est de nouveau pas partiellement satisfaisable. Le deuxième appel récursif retourne donc la valeur $false$ ce qui indique que la contrainte $1 < 2 \wedge 2 < Z$ n'est pas satisfaisable. Ensuite, le premier appel récursif retourne également la valeur $false$ car toutes les valeurs possibles pour Y ont été essayées sans succès. L'appel original essaie donc maintenant d'attribuer la valeur 2 à X. La contrainte $2 < Y \wedge Y < Z$ est partiellement satisfaisable donc on appelle récursivement l'algorithme sur cette contrainte. Les deux itérations ($Y = 1$ et $Y = 2$) échouent et l'appel récursif retourne $false$. Comme nous avons essayé toutes les valeurs pour X, l'appel original retourne $false$ ce qui indique que la contrainte de départ $X < Y \wedge Y < Z$ est insatisfaisable.

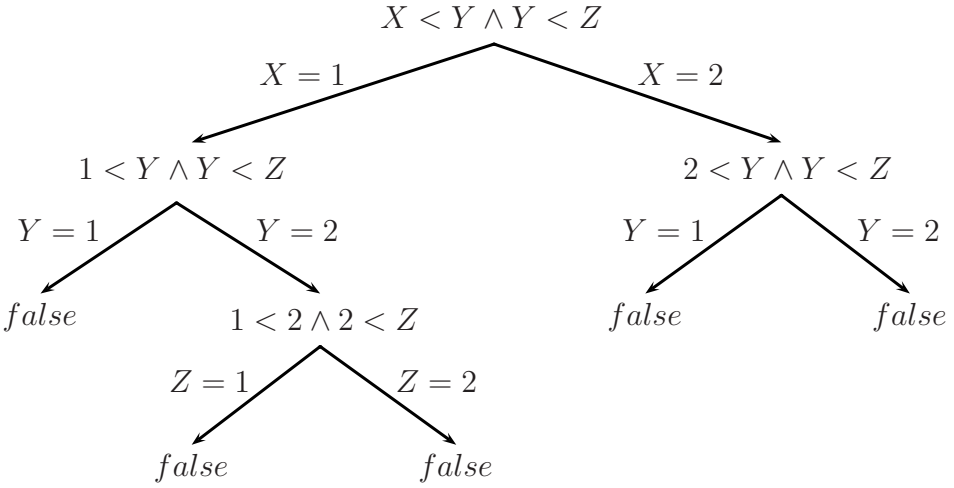

FIG. 2.3 – Arbre de recherche pour le backtracking chronologique

Comme le montre ce simple exemple, la méthode backtracking peut rapidement s'avérer coûteuse (voir figure 2.3). En effet, dans le pire des cas, nous atteignons une complexité exponentielle.

Consistance aux arcs et aux noeuds

Dans cette section, nous nous intéressons aux solveurs de CSP qui ont une complexité polynomiale mais qui sont incomplets. L'idée de ce type de solveurs est que, si le domaine pour une variable est vide, alors la contrainte est insatisfaisable. Nous allons donc transformer le CSP en un autre équivalent de telle sorte que le domaine de chaque variable soit diminué. Pour effectuer ces transformations, nous utilisons le domaine des variables pour essayer d'éliminer des valeurs du domaine d'autres variables. Cette technique est appelée méthode de la **consistance**. Remarquons que les méthodes basées sur la consistance peuvent être combinées avec le backtracking de manière à obtenir un solveur complet. La consistance est alors appliquée à chaque itération de manière à réduire le domaine des variables et donc, accélérer la recherche.

Il existe deux types de domaines remarquables :

- les **domaines faux** : lorsque le domaine d'une variable est l'ensemble vide ;
- les **domaines de valuations** : lorque le domaine de chaque variable est un singleton.

Soient X, Y, Z trois variables. Le domaine D_1, tel que $D_1(X) = \{1, 2\}$, $D_1(Y) = \{1\}$ et $D_1(Z) = \emptyset$, est un domaine faux alors que le domaine D_2, tel que $D_2(X) = \{1\}$, $D_2(Y) = \{2\}$ et $D_2(Z) = \{1\}$ est un domaine de valuation.
Etant donné un domaine de valuation D pour un contrainte C, nous pouvons définir la fonction $satisfaisable(C, D)$ qui renvoie $true$ ou $false$ suivant que C est satisfaisable ou pas avec la valuation D.

Une contrainte primitive c est **consistante aux noeuds** pour un domaine D si

- $|var(c)| \neq 1$ ou
- $|var(c)| = \{X\}$, alors chaque $d \in D(X)$ est une solution de c.

Un CSP avec une contrainte $c_1 \wedge \ldots \wedge c_n$ et un domaine D est consistant aux noeuds si chaque contrainte primitive c_i est consistante aux noeuds avec D pour $1 \leq i \leq n$.

Une contrainte primitive c est **consistante aux arcs** pour un domaine D si

- $|var(c)| \neq 2$ ou
- $|var(c)| = \{X, Y\}$, alors pour chaque $d_x \in D(X)$, il existe un $d_y \in D(Y)$ tel que $\{X \mapsto d_x, Y \mapsto d_y\}$ est une solution de c et pour chaque $d_y \in D(Y)$, il existe un $d_x \in D(X)$ tel que $\{X \mapsto d_x, Y \mapsto d_y\}$ est une solution de c.

Un CSP avec une contrainte $c_1 \wedge \ldots \wedge c_n$ et un domaine D est consistant aux arcs si chaque contrainte primitive c_i est consistante aux arcs avec D pour $1 \leq i \leq n$.

Illustrons ces définitions avec la contrainte suivante :

$$X < Y \wedge Y < Z \wedge Z \leq 2 \tag{2.9}$$

pour le domaine D où $D(X) = D(Y) = D(Z) = \{1, 2, 3\}$.
Cette contrainte n'est pas consistante aux noeuds car la valeur 3 pour la variable Z n'est pas consistante avec la contrainte primitive $Z < 2$. Par contre, si on modifie le domaine de Z telle que $D(Z) = \{1, 2\}$, la contrainte devient bien consistante aux noeuds.
Cette contrainte n'est pas consistante aux arcs car si on donne la valeur 3 à X, il n'existe pas de valeur pour Y qui vérifie la contrainte primitive $X < Y$.

Nous allons maintenant présenter deux algorithmes qui permettent de transformer un CSP en un CSP équivalent qui est consistant aux noeuds et aux arcs.

Soient

- x une variable,

- C une contrainte,
- D un domaine,
- $c_1, \ldots, c_n$ des contraintes primitives et
- d un domaine de valeurs.

Voici les deux fonctions qui permettent d'obtenir un CSP consistant aux noeuds :

`consistant_noeuds`(C, D)
  Soit C `de la forme` $c_1 \wedge \ldots \wedge c_n$
  Pour $i := 1$ `jusque` n `faire`
    $D :=$ `consistant_noeuds_primitive`(c_i, D)
  return D

`consistant_noeuds_primitive`(c, D)
  Si $|vars(c)| = 1$ **alors**
    Soit $\{x\} = vars(c)$
    $D(x) := \{d \in D(x) | \{x \mapsto d\}$ `est une solution de` $c\}$
  return D

Soient

- x, y des variables,
- C une contrainte,
- D, W des domaines,
- $c_1, \ldots, c_n$ des contraintes primitives et
- d_x, d_y des domaines de valeur.

Voici les deux fonctions qui permettent d'obtenir un CSP consistant aux arcs :

`consistant_arcs`(C, D)
  Soit C `de la forme` $c_1 \wedge \ldots \wedge c_n$
  Repeter
    $W := D$
    pour $i := 1$ `jusque` n `faire`
      $D :=$ `consistant_arcs_primitive`(c_i, D)
  jusque $W \equiv D$
  return D

`consistant_arcs_primitive`(c, D)
  Si $|vars(c)| = 2$ **alors**

Soit $\{x, y\} = vars(c)$
$D(x) := \{d_x \in D(x) | \exists\ d_y \in D(y), \{x \mapsto d_x, y \mapsto d_y\}$ `est une solution de` $c\}$
$D(y) := \{d_y \in D(y) | \exists\ d_x \in D(x), \{x \mapsto d_x, y \mapsto d_y\}$ `est une solution de` $c\}$
return D

En utilisant les algorithmes des consistances aux noeuds et aux arcs, nous pouvons construire un solveur incomplet pour déterminer si un CSP est satisfaisable. On applique d'abord l'algorithme de consistance aux noeuds puis celui de consistance aux arcs. Si on obtient un faux domaine, alors le CSP est insatisfaisable. Si le domaine résultant est un domaine de valuation, le problème est satisfaisable et la solution est la valuation obtenue. Dans tous les autres cas, on ne sait rien dire et on retourne donc la valeur *unknown*.

Reprenons la contrainte (2.9) avec le domaine $D(X) = D(Y) = D(Z) = \{1, 2, 3\}$. En appliquant l'algorithme de consistance aux noeuds, on modifie le domaine de la variable Z tel que $D(Z) = \{1, 2\}$. La consistance aux arcs s'obtient de la manière suivante :

1. On traite la contrainte primitive $X < Y$ à la première itération. Le domaine devient $D(X) = \{1, 2\}$ et $D(Y) = \{2, 3\}$. La valeur 3 est enlevée du domaine de X car il n'existe pas de valeur dans le domaine de Y qui vérifie $3 < Y$. De façon similaire, on enlève la valeur 1 du domaine de Y car il n'existe pas de valeur dans le domaine de X qui vérifie $X < 1$.
2. En examinant la contrainte $Y < Z$, le domaine de Y devient vide car il n'y a aucune valeur strictement inférieure à une des valeurs de Z. Le domaine de Z devient également vide.
3. A la prochaine itération, le domaine de X devient l'ensemble vide.
4. Comme il n'y a plus de changement dans le domaine, l'algorithme s'arrête et renvoie la valeur $false$.

Comme nous l'avons déjà dit, la consistance aux noeuds et aux arcs peut être combinée avec l'algorithme de backtracking pour obtenir un solveur complet. Tout d'abord, on utilise les algorithmes de consistance aux noeuds et aux arcs pour restreindre le domaine. Ensuite, le backtracking choisit une valeur pour une variable et la consistance est de nouveau appliquée. Si on obtient un domaine faux, le CSP résultant est insatisfaisable et le backtracking doit choisir une autre valeur possible. On procède de cette manière jusqu'à ce qu'on trouve une solution ou jusqu'à ce que toutes les valeurs possibles soient testées.

Consistance aux bornes

L'inconvénient de la consistance aux noeuds et aux arcs est qu'on ne peut l'appliquer qu'à des contraintes primitives avec une et deux variables. Chaque contrainte primitive possédant plus de 2 variables sera ignorée lors de la vérification de la consistance. Le concept de consistance aux hyper-arcs étend celui de consistance aux arcs de manière à pouvoir l'appliquer à n'importe quelle contrainte et ce quel que soit le nombre de variables

de cette contrainte.

Une contrainte primitive c est **consistante aux hyper-arcs** pour un domaine D si pour chaque variable $x \in vars(c)$ et domaine de valuation $d \in D(x)$, il existe une affectation pour les variables restantes $x_1, \ldots, x_k$ telle que $d_j \in D(x_j)$ pour $1 \leq j \leq k$ et $\{x \mapsto d, x_1 \mapsto d_1, \ldots, x_k \mapsto d_k\}$ est une solution.
Un CSP avec une contrainte $c_1 \wedge \ldots \wedge c_n$ et un domaine D est consistant aux hyper-arcs si chaque contrainte primitive c_i est consistante aux hyper-arcs pour le domaine D pour $1 \leq i \leq n$.

La consistance aux hyper-arcs est une généralisation de la consistance aux noeuds et aux arcs. Dans le cas d'une contrainte primitive avec deux variables, la consistance aux hyper-arcs est équivalente à la consistance aux arcs et dans le cas d'une contrainte primitive avec une variable, elle est équivalente à la consistance aux noeuds. Le problème de cette méthode est qu'elle est beaucoup trop coûteuse pour être employée à chaque étape dans un solveur incomplet. En effet, déterminer si une contrainte primitive est consistante aux hyper-arcs nécessite un algorithme de complexité exponentielle. Nous avons donc besoin d'une autre condition de consistance pour les contraintes impliquant plus de 2 variables. C'est pourquoi nous introduisons la notion de consistance aux bornes.

Nous définissons un **intervalle** $[l \ldots u]$ comme un ensemble d'entiers $\{l, l+1, \ldots, u\}$ si $l \leq u$. Sinon, cela représente l'ensemble vide.
Si D est un domaine sur les entiers, $min_D(x)$ est l'élément minimal de $D(x)$ et $max_D(x)$ est l'élément maximal de $D(x)$.

Une contrainte primitive est dite **arithmétique** si chacune de ses variables a pour domaine un intervalle fini d'entiers.

Un contrainte primitive c est **consistante aux bornes** pour un domaine D si pour chaque variable $x \in vars(c)$, il y a :

- une assignation de nombres *réels* $d_1, d_2, \ldots, d_k$ aux variables restantes $x_1, x_2, \ldots, x_k$ telle que $min_D(x_j) \leq d_j \leq max_D(x_j)$ pour chaque nombre d_j $(j = 1, \ldots, k)$ et

  $$\{x \mapsto min_D(x), x_1 \mapsto d_1, \ldots, x_k \mapsto d_k\}$$

  est une solution de c ;
- une assignation de nombres *réels* $d'_1, d'_2, \ldots, d'_k$ aux variables $x_1, x_2, \ldots, x_k$ telle que $min_D(x_j) \leq d'_j \leq max_D(x_j)$ pour chaque nombre d'_j $(j = 1, \ldots, k)$ et

  $$\{x \mapsto max_D(x), x_1 \mapsto d'_1, \ldots, x_k \mapsto d'_k\}$$

  est une solution de c.

Un CSP arithmétique avec une contrainte $c_1 \wedge \ldots \wedge c_n$ et un domaine D est consistant aux bornes si chaque contrainte primitive c_i est consistante aux bornes avec D pour $1 \leq i \leq n$.

Par exemple, si nous considérons la contrainte

$$X = 3Y + 5Z \tag{2.10}$$

avec le domaine D tel que

$$D(X) = [2, \ldots, 7],\ D(Y) = [0, \ldots, 2],\ D(Z) = [-1, \ldots, 2].$$

Cette contrainte n'est pas consistante aux bornes pour D. En effet, si nous réécrivons la contrainte $5Z = X - 3Y$ ce qui est équivalent à (2.10) et si Z prend sa valeur maximale 2, alors la partie gauche de l'équation vaut 10. Cependant, la valeur maximale que la partie droite $X - 3Y$ peut atteindre est $7 - 3 \times 0 = 7$. L'intervalle de Z doit donc être modifié pour obtenir la consistance aux bornes. Le nouveau domaine D_1 est, quant à lui, consistant aux bornes :

$$D_1(X) = [2, \ldots, 7],\ D_1(Y) = [0, \ldots, 2],\ D_1(Z) = [0, \ldots, 1].$$

Etant donné un intervalle pour chaque variable d'une contrainte primitive, nous pouvons déduire des méthodes pour calculer un nouvel intervalle tel que la contrainte devienne consistante aux bornes. De telles méthodes sont appelées **règles de propagation**.

Les règles de propagation dépendent du type de contrainte primitive que l'on veut rendre consistante aux bornes. Sans entrer dans les détails, nous pouvons tout de même signaler que la détermination de ce type de règles peut rapidement devenir très couteux. C'est pour cette raison que, généralement, les solveurs utilisant la consistance aux bornes se limitent à des contraintes primitives ayant une forme relativement simple. Les contraintes plus complexes doivent donc être transformées en conjonction de contraintes simples avant d'être traitées par le solveur.

Nous allons à présent donner l'algorithme qui permet de transformer un CSP en un autre CSP équivalent consistant aux bornes. Soient :

- C une contrainte arithmétique ;
- C_0 un ensemble de contraintes primitives ;
- D et D_1 des domaines ;
- $c_1, \ldots, c_n$ des contraintes primitives et
- x une variable.

Soit également `consistant_bornes_primitive`(c, D) la fonction qui applique les règles de propagation pour la contrainte c avec le domaine D et qui retourne le nouveau domaine. Voici la fonction qui permet de rendre un CSP consistant aux bornes :

```
consistant_bornes(C, D)
    Soit C de la forme c_1 ∧ ... ∧ c_n
    C_0 := {c_1, ..., c_n}
    Tant que C_0 ≠ ∅ faire
        Choisir c ∈ C_0
        C_0 := C_0\{c}
        D_1 :=consistant_bornes_primitive(c, D)
        Si D_1 est un domaine faux alors return D_1
        Pour i := 1 jusque n faire
          Si il existe x ∈ vars(c_i) tel que D_1(x) ≠ D(x) alors
              C_0 := C_0 ∪ {c_i}
        D := D_1
    return D
```

L'algorithme travaille itérativement sur l'ensemble des contraintes actives C_0 qui est défini comme l'ensemble des contraintes qui peuvent ne pas être consistantes aux bornes avec le domaine D. La boucle "tant que" choisit une contrainte primitive active c dans cet ensemble et modifie le domaine D pour rendre c consistant aux bornes avec D. Si le nouveau domaine est faux, alors l'algorithme se termine sinon il continue jusqu'à ce que l'ensemble des contraintes actives soit vide c'est à dire que toutes les contraintes soient consistantes aux bornes avec D.

De manière analogue à ce qui a été fait pour la consistance aux noeuds et aux arcs, nous pouvons combiner la consistance aux bornes avec le backtracking pour obtenir un solveur complet.

En outre, la consistance aux bornes peut être combinée avec la consistance aux noeuds et aux arcs. En effet, lorsqu'une contrainte primitive ne possède que 2 variables, la consistance aux arcs donnera de meilleurs résultats que la consistance aux bornes. Par exemple, la contrainte $X^2 = 1 - Y^2 \wedge X \neq 0 \wedge Y \neq 0$ avec le domaine $D(X) = D(Y) = \{-1, 0, 1\}$ est consistante aux bornes alors qu'elle n'est pas consistante aux arcs. Dans le cas contraire, i.e. lorsqu'une contrainte possède plus de 2 variables, seule la consistance aux bornes fournira des résultats intéressants. C'est pourquoi la meilleure approche reste la combinaison des différentes techniques dont nous avons parlé.

Optimisation pour CSP arithmétiques

Jusqu'à présent, nous avons considéré le problème de trouver une solution d'un CSP alors que dans la plupart des problèmes, l'utilisateur ne voudra pas seulement trouver une solution mais voudra trouver une solution optimale.

Il existe une manière très simple de trouver une solution optimale en utilisant le back-

tacking. Il suffit d'effectuer une recherche de solution avec les techniques que nous avons considérées dans les sections précédentes et de stocker dans un variable θ_{best} la meilleure solution. θ_{best} est initialisé à $false$ de manière à ce que, si on ne trouve pas de solution et que θ_{best} n'est pas changé, il n'y ait pas d'erreur. Lorsqu'on trouve la première solution, on la stocke dans la variable et on lance une nouvelle recherche. Si on trouve une solution qui est meilleure que celle stockée dans θ_{best}, on l'enregistre en écrasant la valeur présente dans θ_{best}, sinon on continue la recherche jusqu'à ce qu'il n'y ait plus de solution.

2.6 Programmation logique avec contraintes

2.6.1 Introduction

Dans la section précédente, nous avons étudié les contraintes ainsi que plusieurs types de domaines. Nous nous sommes attardés en particulier sur les domaines finis et avons développé plusieurs techniques qui permettent de trouver une solution pour un problème avec contraintes dans un domaine fini. Nous allons maintenant introduire un paradigme de programmation qui utilise à la fois la logique des prédicats et les contraintes : la programmation logique avec contraintes.

Comme nous l'avons déjà signalé, les contraintes sont fort utiles pour modéliser le comportement attendu des entités d'un système. En guise d'exemple, considérons la loi d'Ohm

$$V = I \times R, \tag{2.11}$$

qui décrit la relation entre le voltage V, l'intensité I et la résistance R d'un circuit électrique. Dans les langages de programmation traditionnels comme le C, le programmeur ne peut pas directement utiliser cette relation. Si il veut calculer I à partir des valeurs V et R, il doit utiliser l'affectation `I := V /R`. De manière similaire, si il veut calculer la valeur de R à partir de V et I, il doit utiliser l'instruction suivante : `R := V/I`.

Demander à l'utilisateur de maintenir explicitement de telles relations dans des systèmes comprenant peu de contraintes peut être acceptable mais dans le cas de système avec un grand nombre de contraintes cela devient beaucoup trop difficile. C'est pour résoudre ce genre de problème que sont apparus les langages de programmation avec contraintes.

Les moteurs d'inférence de programmation logique comme Prolog possèdent des extensions qui leur permettent de réaliser des calculs arithmétiques. Par exemple, le prédicat Prolog `is/2` peut être utilisé de la façon suivante :

```
X is 2 + 3.
```

Par contre, si on essaie d'exprimer la contrainte (2.11) sous la forme

```
V is I*R,
```

nous devons nous assurer que les variables présentes dans la partie droite du prédicat soient instanciées au moment de l'évaluation de celui-ci. Dans le cas contraire, le système va renvoyer une erreur. C'est pourquoi plusieurs langages de programmation logique ont été étendus avec des solveurs de contraintes. Dans le cadre de ce mémoire, nous utiliserons GNU Prolog qui a été developpé par Daniel Diaz.

Si nous créons le fichier `ohm.pl`[8] suivant

```
ohm(V,I,R):- V#=I*R.
```

et que nous le chargeons dans le moteur GNU Prolog, nous pouvons alors exécuter les requêtes suivantes sans problème :

- `ohm(V,5,6)` qui va nous donner le résultat `V = 30`;
- `ohm(30,I,6)` qui va nous donner le résultat `I = 5`;
- `ohm(30,5,R)` qui va nous donner le résultat `R = 6`.

Comme pour les programmes logiques, la seule construction nécessaire pour réaliser un programme logique avec contraintes est la *règle*. Les règles permettent au programmeur de définir ses propres contraintes sous forme de contraintes arithmétiques. Elles sont également utilisées pour *évaluer* les buts entrés par l'utilisateur.

2.6.2 Règles

Comme nous venons de le signaler, les programmes logiques avec contraintes sont formés d'un ensemble de règles. Ces règles peuvent être des contraintes définies par l'utilisateur ou bien de simples prédicats. Cependant, un prédicat peut aussi être défini en terme d'autres prédicats. Ceux-ci peuvent donc être définis de manière récursive. Un prédicat peut également être défini par plus d'une règle. Par exemple, considérons la fonction factorielle :

$$N! = \begin{cases} 1 & \text{si } N = 0, \\ N \times (N-1)! & \text{si } N \geq 1. \end{cases}$$

Nous pouvons écrire le prédicat fac(N,F) qui est vrai si F vaut $N!$:

```
factoriel(0,1).
factoriel(N,F):-
    N1 #= N-1,
    F #= N*F1,
    factoriel(N1,F1).
```

Le système est également en mesure de répondre à des requêtes comme :

- `factoriel(5,X)` qui nous donne la réponse $X = 120$ qui est égale à 5! et

[8] La suite de symboles `#=` représente la contrainte d'égalité entre le membre de gauche et le membre de droite.

- `factoriel(X,120)` qui nous donne la réponse $X = 5$.

On remarque que le prédicat `factoriel` est appelé de manière récursive et qu'il est présent dans plus d'une règle.

2.6.3 Evaluation

La méthode d'évaluation d'un but dans un programme logique avec contraintes est basée sur le principe de dérivation SLD que nous avons développé à la section 2.3. La différence vient du fait que lorsqu'on développe un atome, on vérifie également les contraintes auxquelles sont soumises les variables de cet atome. Si les contraintes sont insatisfaisables, on arrête la résolution, sinon on continue la dérivation. A chaque étape de celle-ci, nous avons donc un ensemble de contraintes et un but en cours. Nous appelons ces 2 éléments **l'état** du calcul.

2.6.4 Modélisation avec un domaine de contraintes finies

Dans cette section, nous détaillons quelques contraintes spéciales qui sont fournies par les langages de programmation logique avec contraintes en les utilisant dans plusieurs exemples simples mais qui ont le mérite de les illustrer.

Domaine, labelling et optimisation

Considérons le problème du contrebandier avec un sac à dos. Un contrebandier a un sac à dos avec une capacité limitée de 9 unités par exemple. Il peut faire de la contrebande avec des bouteilles de whisky, des bouteilles de parfums et des cartouches de cigarettes qui occupent respectivement 4, 3 et 2 unités. Le profit qu'il tire de la vente d'une bouteille de whisky, d'une bouteille de parfum et d'une cartouche de cigarettes est de 15, 10 et 7 euros respectivement. Comment le contrebandier doit-il remplir son sac si il ne peut faire qu'un voyage et si il veut que son profit soit supérieur à 30 ? Nous pouvons modéliser ce problème avec des contraintes arithmétiques linéaires.
Soient

- W le nombre de bouteilles de whisky ;
- P le nombre de bouteilles de parfum ;
- C le nombre de fardes de cigarettes.

Si nous prenons comme domaine initial pour les variables W, P et C l'intervalle [0,. . . ,9], cette contrainte peut être modélisée à l'aide du prédicat suivant :

```
fd_domain([W,P,C],0,9).
```

Les autres contraintes sont modélisées comme suit :

```
4*W + 3*P + 2*C #<= 9, 15*W + 10*P + 7*C #>= 30
```

Si nous demandons une solution à GNU Prolog, celui-ci retournera simplement la valeur *unknown*. Cela est dû au fait que le solveur basé sur les techniques de consistance est *incomplet* et n'est donc pas assez puissant pour déterminer si la contrainte est satisfaisable. La raison pour laquelle GNU Prolog utilise un solveur incomplet pour résoudre les CSP est que, comme nous l'avons fait remarquer auparavant, les solveurs complets sont très coûteux en ressources et ne doivent donc pas être utilisés de manière systématique.

Nonobstant, nous désirons quand même trouver une solution à notre problème du contrebandier c'est pourquoi la plupart des systèmes de résolution de CSP (dont GNU Prolog) fournissent également un solveur complet qui utilise le backtracking. A cause du coût d'un tel solveur, celui-ci n'est pas utilisé automatiquement et doit être appelé explicitement par le programmeur. Dans le cas de GNU Prolog, le solveur est appelé par la contrainte

```
fd_labeling([W,P,C]).
```

En ajoutant cette contrainte nous obtenons le programme suivant :

```
1  sacados(W,P,C):-
2      fd_domain([W,P,C],0,9),
3      4*W + 3*P + 2*C #=< 9,
4      15*W + 10*P + 7*C #>= 30,
5      fd_labeling([W,P,C]).
```

et GNU Prolog nous donne les solutions suivantes :

$$W = 0 \wedge P = 1 \wedge C = 3,\ W = 0 \wedge P = 3 \wedge C = 0,$$
$$W = 1 \wedge P = 1 \wedge C = 1,\ W = 2 \wedge P = 0 \wedge C = 0.$$

Le programmeur peut également être intéressé par le profit maximum que le contrebandier peut tirer à la place d'obtenir simplement les solutions possibles avec un profit supérieur à 30. Pour ce faire, on utilise la contrainte d'optimisation suivante :

```
fd_maximize(fd_labeling([W,P,C],Profit)).
```

Le programme suivant ne nous donne qu'une seule réponse qui est la réponse optimale à savoir $W = 1$, $P = 1$, $C = 1$ et le profit est égale à 32.

```
sacadosoptimal(W,P,C,Profit):-
    fd_domain([W,P,C],0,9),
    4*W + 3*P + 2*C #=< 9,
    Profit #= 15*W + 10*P + 7*C,
    fd_maximize(fd_labeling([W,P,C]),Profit).
```

Contraintes complexes

Nous allons maintenant décrire deux contraintes complexes supplémentaires :

- `fd_element`
- `fd_all_different`

La contrainte *element* simule le comportement d'un tableau par une liste. La contrainte

$$element(I, [V_1, \ldots, V_m], X)$$

maintient la relation $X = V_I$. Avec GNU Prolog, cette contrainte peut être utilisée avec le prédicat `fd_element(I,List,X).`

La contrainte $alldifferent(Liste\_Variables)$ permet de spécifier que les variables contenues dans la liste $Liste\_Variables$ ne peuvent pas prendre des valeurs identiques. En GNU Prolog, nous devons utiliser le prédicat `fd_all_different(ListVar)`.

Pour conclure ce chapitre, nous allons présenter un exemple qui utilise de manière intensive la contrainte complexe $alldifferent$: le jeu du sudoku.

Pour rappel, le but de ce jeu est de compléter une grille de chiffres composée de 9 lignes et 9 colonnes de telle sorte que :

- il n'y a pas 2 chiffres les mêmes sur la même ligne ;
- il n'y a pas 2 chiffres les mêmes sur la même colonne ;
- il n'y a pas 2 chiffres les mêmes dans chacune des sous-grilles de 3 lignes et 3 colonnes.

Etant donné l'exemple suivant :

FIG. 2.4 – Exemple du jeu sudoku

Nous devons obtenir la solution suivante :

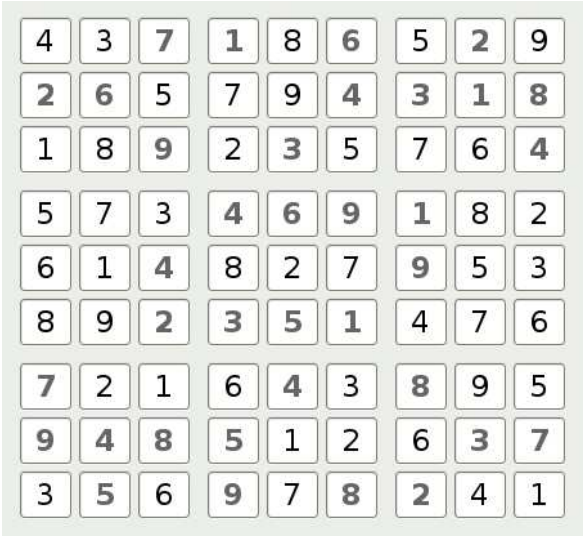

FIG. 2.5 – Solution de l'exemple de sudoku

Voici le programme qui permet de trouver une solution au problème du sudoku :

```
1   sudoku(X) :-
2       X = [[X11,X12,X13,X14,X15,X16,X17,X18,X19],
3            [X21,X22,X23,X24,X25,X26,X27,X28,X29],
4            [X31,X32,X33,X34,X35,X36,X37,X38,X39],
5            [X41,X42,X43,X44,X45,X46,X47,X48,X49],
6            [X51,X52,X53,X54,X55,X56,X57,X58,X59],
7            [X61,X62,X63,X64,X65,X66,X67,X68,X69],
8            [X71,X72,X73,X74,X75,X76,X77,X78,X79],
9            [X81,X82,X83,X84,X85,X86,X87,X88,X89],
10           [X91,X92,X93,X94,X95,X96,X97,X98,X99]],
11
12      fd_domain([X11,X12,X13,X14,X15,X16,X17,X18,X19,
13                 X21,X22,X23,X24,X25,X26,X27,X28,X29,
14                 X31,X32,X33,X34,X35,X36,X37,X38,X39,
15                 X41,X42,X43,X44,X45,X46,X47,X48,X49,
16                 X51,X52,X53,X54,X55,X56,X57,X58,X59,
17                 X61,X62,X63,X64,X65,X66,X67,X68,X69,
18                 X71,X72,X73,X74,X75,X76,X77,X78,X79,
19                 X81,X82,X83,X84,X85,X86,X87,X88,X89,
20                 X91,X92,X93,X94,X95,X96,X97,X98,X99] , 1, 9),
21
22      X11= _ ,X12= _ ,X13= 7 ,X14= 1 ,X15= _ ,X16= 6 ,X17= _ ,X18= 2 ,X19= _ ,
```

```
23      X21= 2 ,X22= 6 ,X23= _ ,X24= _ ,X25= _ ,X26= 4 ,X27= 3 ,X28= 1 ,X29= 8 ,
24      X31= _ ,X32= _ ,X33= 9 ,X34= _ ,X35= 3 ,X36= _ ,X37= _ ,X38= _ ,X39= 4 ,
25      X41= _ ,X42= _ ,X43= _ ,X44= 4 ,X45= 6 ,X46= 9 ,X47= 1 ,X48= _ ,X49= _ ,
26      X51= _ ,X52= _ ,X53= 4 ,X54= _ ,X55= _ ,X56= _ ,X57= 9 ,X58= _ ,X59= _ ,
27      X61= _ ,X62= _ ,X63= 2 ,X64= 3 ,X65= 5 ,X66= 1 ,X67= _ ,X68= _ ,X69= _ ,
28      X71= 7 ,X72= _ ,X73= _ ,X74= _ ,X75= 4 ,X76= _ ,X77= 8 ,X78= _ ,X79= _ ,
29      X81= 9 ,X82= 4 ,X83= 8 ,X84= 5 ,X85= _ ,X86= _ ,X87= _ ,X88= 3 ,X89= 7 ,
30      X91= _ ,X92= 5 ,X93= _ ,X94= 9 ,X95= _ ,X96= 8 ,X97= 2 ,X98= _ ,X99= _ ,
31
32      % contraintes sur les lignes
33      fd_all_different([X11,X12,X13,X14,X15,X16,X17,X18,X19]),
34      fd_all_different([X21,X22,X23,X24,X25,X26,X27,X28,X29]),
35      fd_all_different([X31,X32,X33,X34,X35,X36,X37,X38,X39]),
36      fd_all_different([X41,X42,X43,X44,X45,X46,X47,X48,X49]),
37      fd_all_different([X51,X52,X53,X54,X55,X56,X57,X58,X59]),
38      fd_all_different([X61,X62,X63,X64,X65,X66,X67,X68,X69]),
39      fd_all_different([X71,X72,X73,X74,X75,X76,X77,X78,X79]),
40      fd_all_different([X81,X82,X83,X84,X85,X86,X87,X88,X89]),
41      fd_all_different([X91,X92,X93,X94,X95,X96,X97,X98,X99]),
42
43      % contraintes sur les colonnes
44
45      fd_all_different([X11,X21,X31,X41,X51,X61,X71,X81,X91]),
46      fd_all_different([X12,X22,X32,X42,X52,X62,X72,X82,X92]),
47      fd_all_different([X13,X23,X33,X43,X53,X63,X73,X83,X93]),
48      fd_all_different([X14,X24,X34,X44,X54,X64,X74,X84,X94]),
49      fd_all_different([X15,X25,X35,X45,X55,X65,X75,X85,X95]),
50      fd_all_different([X16,X26,X36,X46,X56,X66,X76,X86,X96]),
51      fd_all_different([X17,X27,X37,X47,X57,X67,X77,X87,X97]),
52      fd_all_different([X18,X28,X38,X48,X58,X68,X78,X88,X98]),
53      fd_all_different([X19,X29,X39,X49,X59,X69,X79,X89,X99]),
54
55      % contraintes sur les carres
56
57      fd_all_different([X11,X12,X13,X21,X22,X23,X31,X32,X33]),
58      fd_all_different([X14,X15,X16,X24,X25,X26,X34,X35,X36]),
59      fd_all_different([X17,X18,X19,X27,X28,X29,X37,X38,X39]),
60      fd_all_different([X41,X42,X43,X51,X52,X53,X61,X62,X63]),
61      fd_all_different([X44,X45,X46,X54,X55,X56,X64,X65,X66]),
62      fd_all_different([X47,X48,X49,X57,X58,X59,X67,X68,X69]),
63      fd_all_different([X71,X72,X73,X81,X82,X83,X91,X92,X93]),
64      fd_all_different([X74,X75,X76,X84,X85,X86,X94,X95,X96]),
65      fd_all_different([X77,X78,X79,X87,X88,X89,X97,X98,X99]),
```

```
66
67      fd_labeling([X11,X12,X13,X14,X15,X16,X17,X18,X19,
68                   X21,X22,X23,X24,X25,X26,X27,X28,X29,
69                   X31,X32,X33,X34,X35,X36,X37,X38,X39,
70                   X41,X42,X43,X44,X45,X46,X47,X48,X49,
71                   X51,X52,X53,X54,X55,X56,X57,X58,X59,
72                   X61,X62,X63,X64,X65,X66,X67,X68,X69,
73                   X71,X72,X73,X74,X75,X76,X77,X78,X79,
74                   X81,X82,X83,X84,X85,X86,X87,X88,X89,
75                   X91,X92,X93,X94,X95,X96,X97,X98,X99]),
76      ecrire(X).
77
78  ecrire([]).
79  ecrire([H|T]) :-
80      write(H), nl, ecrire(T).
```

Le tableau de chiffre est modélisé par une liste de 9 listes de chiffres (lignes 2 à 10). Les données du problèmes que l'on veut traiter sont encodées aux lignes 22 à 30. Les 3 contraintes citées ci-dessus sont modélisées par des contraintes $alldifferent$ de la ligne 32 à la ligne 65. On déclare également le domaine pour chacune des variables comme étant l'intervalle [1,...,9] (ligne 12). Une fois qu'on a trouvé la solution, on appelle le prédicat `ecrire(X)` pour afficher correctement cette solution.

2.7 Conclusion

Dans ce chapitre, nous avons présenté le paradigme de programmation logique ainsi qu'une de ses extensions : la programmation logique avec contraintes. Après avoir développé la théorie nécessaire à la compréhension de ce paradigme et son mécanisme d'exécution, nous avons présenté ses avantages à travers plusieurs exemples simples. Le lecteur intéressé désirant des informations supplémentaires sur la programmation logique avec contraintes consultera [4], [3], [5], [6], ainsi que [7].

Par la suite, nous allons tenter d'appliquer le paradigme de programmation logique avec contraintes à un problème d'une certaine ampleur : le calcul de l'impôt des personnes physiques en Belgique en 2007.

Chapitre 3

Architecture

Dans ce chapitre, nous décrivons l'architecture logicielle du système expert que nous allons construire.

Après une brève introduction sur ce qu'est la déclaration d'impôt, nous présenterons les données que nous allons exploiter et nous exposerons les différentes technologies Web utilisées afin de mettre en place un système de déclaration d'impôt sur le Web.

3.1 La déclaration d'impôt

Chaque année, tout contribuable belge a l'obligation de rentrer sa déclaration d'impôt auprès de son service de taxation. La déclaration d'impôt comprend plusieurs renseignements personnels sur les revenus, la charge de famille, les pensions perçues,... du contribuable. Ces renseignements sont regroupés, selon leurs catégories, dans des cadres distincts.

La déclaration d'impôt pour l'exercice d'imposition 2007 contient 2 parties. La première étant destinée à tous les contribuables, la seconde étant uniquement destinée aux dirigeants d'entreprise, aux commerçants et aux titulaires de profession libérale. Dans le cadre de ce mémoire, nous nous occuperons uniquement de la partie 1 de la déclaration. Celle-ci possède 12 cadres.

En guise d'illustration, la figure 3.1 présente une partie du cadre 4 de la partie 1 de la déclaration d'impôt. Le cadre 4 est consacré, comme nous pouvons le voir dans la figure, aux revenus perçus par le contribuable (traitements, allocations de chômage, revenus de remplacement, ...).

Le contribuable doit donc remplir les différents champs présents dans chaque cadre et envoyer l'entièreté du document à son service de taxation qui établira une imposition en fonction des données collectées.

Cadre IV. - TRAITEMENTS, SALAIRES, ALLOCATIONS DE CHOMAGE, INDEMNITES LEGALES DE MALADIE-INVALIDITE, REVENUS DE REMPLACEMENT ET PREPENSIONS.

A. REMUNERATIONS ORDINAIRES.

1. Total des rémunérations :	1250	
2. Options sur actions ou parts, attribuées :		
a) en 2006 :	1249	
b) de 1999 à 2005 : montant qui devient imposable en 2006 :	1248	
3. Pécule de vacances anticipé :	1251	
4. Arriérés :	1252	
5. Indemnités de dédit :	1253	
6. Indemnités de reclassement :	1245	
7. Prime régionale de remise au travail :	1291	
8. Remboursement des frais de déplacement du domicile au lieu de travail :		
a) montant total :	1254	
b) exonération :	1255	
9. Forfait pour longs déplacements :	1256	
10. Cotisations sociales personnelles non retenues :	1257	
11. Autres frais professionnels (à ne compléter que si vous ne souhaitez pas l'application du forfait légal) :	1258	

B. ALLOCATIONS DE CHOMAGE.

1. Allocations sans complément d'ancienneté :		
a) allocations légales et complémentaires :	1260	
b) arriérés :	1261	

FIG. 3.1 – Partie du cadre 4 de la déclaration d'impôt

3.2 Structure des données

Comme nous pouvons le voir à la figure 3.1, chacun des champs à compléter possède un numéro identifiant appelé code. Par exemple, le total de rémunération est repris sous le code 1250 alors que le forfait pour les longs déplacements doit être renseigné sous le code 1256. Cette façon de procéder va nous permettre d'utiliser ces codes comme nom de variable à la place de nom plus explicite mais plus ambigu. Le total des rémunérations sera donc enregistré dans une base de données sous le champ `v1250`.

Etant donné que chaque contribuable possède un numéro national distinct, nous utiliserons celui-ci comme identifiant dans notre base de données.

3.3 Technologies

Dans cette section, nous présentons les différentes technologies que nous allons utiliser pour construire le système expert.

3.3.1 PHP

Le PHP[1] est un langage interprété (un langage de script) qui est exécuté du côté du serveur (contrairement au langage Javascript ou aux applets Java qui sont exécutés du côté client). La figure 3.2 illustre ce concept.

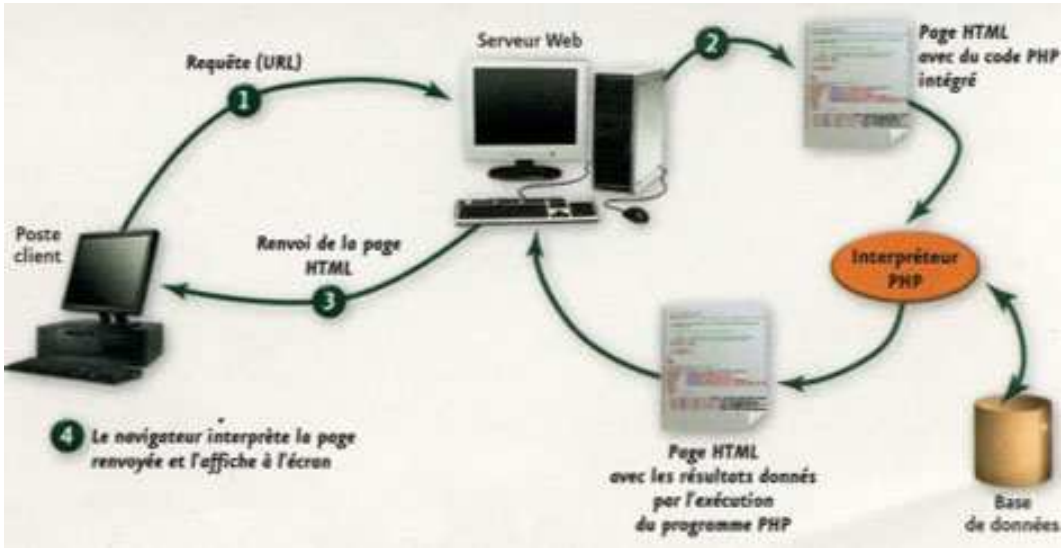

FIG. 3.2 – Architecture d'un serveur Web couplé avec un interpréteur PHP

Lorsqu'un serveur Web reçoit une requête où du PHP doit être interprété, il demande à l'interpréteur PHP de la faire avant de renvoyer la réponse au client. Celui-ci peut éventuellement interagir avec une base de données. Pour indiquer au serveur Web qu'il a du code PHP à interpréter dans la requête qu'il reçoit, il suffit d'utiliser l'extension ".php".

Le langage PHP permet de manipuler des variables à l'aide de boucles, de conditions if-then-else comme on le ferait dans un langage de programmation impératif à l'intérieur même d'une page html. Le PHP permet également de communiquer avec des bases de données et donc d'enregistrer des données récoltées dans un formulaire html.

Pour ce faire, les parties de code PHP qui devront être interprétées doivent être encadrées par les balises "`<?php`" et "`?>`".

Par exemple, pour se connecter à la base de données `sef` avec le nom d'utilisateur "root" et le mot de passe "test" hébergée sur la machine locale, nous placerons les instructions suivantes dans une page html.

[1]PHP : Hypertext Processor

```
<?php
mysql_connect(localhost,root,test) ;
mysql_select_db(‘‘sef’’) ;
?>
```

3.3.2 CGI [1]

Le CGI[2] est la partie d'un serveur Web qui peut communiquer avec les autres programmes qui tournent sur le serveur. Avec CGI, le serveur peut appeler des programmes avec des arguments spécifiques qui ont été entrés par l'utilisateur par le biais d'un formulaire html par exemple. La figure 3.3 tirée de [1] permet de comprendre facilement le fonctionnement de CGI. Un utilisateur soumet un formulaire html. Ensuite, le serveur transmet les données au script CGI qui exécute le programme demandé. Celui-ci renvoie alors le résultat au serveur Web qui le communique au client si nécessaire.

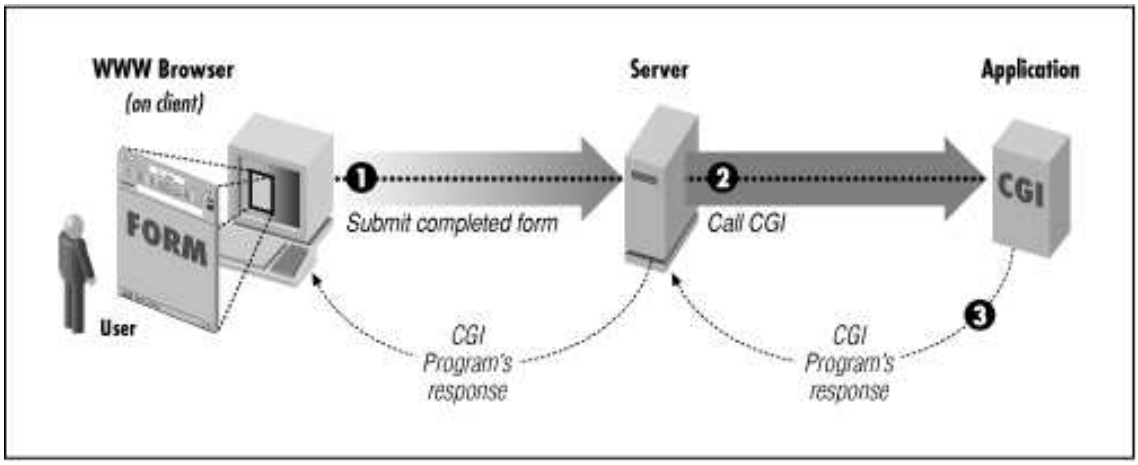

FIG. 3.3 – Diagramme d'application de CGI

La programmation en CGI n'est que de la programmation ordinaire à laquelle il faut ajouter quelques règles strictes "d'output" afin que le serveur puisse interpréter le résultat de l'exécution du programme correctement.

Nous pouvons utiliser presque n'importe quel langage de programmation pour créer des scripts CGI même si certains sont plus appropriés que d'autres. Dans le cadre de ce mémoire, nous utiliserons le langage C pour des raisons d'interfaçage avec GNU Prolog que nous expliquerons par la suite.

L'output du script que l'on va exécuter doit être interprétable par le serveur Web. Dans la plupart des cas, le retour attendu par le serveur sera du html. Pour le spécifier, notre

[2]Common Gateway Interface

programme devra contenir l'instruction C suivante :

```
printf("Content-type : text/html\n\n") ;
```

Dans notre script, nous devons ensuite récupérer les variables qui ont été entrées par l'utilisateur. Pour ce faire, nous utilisons la fonction `uncgi()`(voir Annexe A). Une fois qu'on a appelé cette fonction, nous pouvons accéder à toutes les variables récupérées dans le formulaire hmtl comme si il s'agissait de variables d'environnement. En C, cela se fait via l'instruction `getenv("WWW_nomvariable")`.Une fois que toutes les variables sont récupérées, nous créons l'en-tête html, exécutons les différentes tâches nécessaires puis construisons l'output à renvoyer au serveur sous forme de page html.

3.3.3 Interface C/GNU Prolog [2]

Outre le fait qu'il possède un solveur de contraintes finies, GNU Prolog permet également de créer des exécutables rapides à partir de programmes C et Prolog. En effet, il est possible d'appeler des fonctions C dans un prédicat Prolog mais aussi d'appeler des prédicats Prolog dans un programme C. C'est cette dernière possibilité que nous exploiterons dans le cadre de ce mémoire. Après avoir créé l'en-tête dans notre script CGI, nous allons démarrer le moteur d'inférence de GNU Prolog et appeler le prédicat qui va nous permettre de calculer la somme dûe/à recevoir par le contribuable lors de sa déclaration d'impôt. Pour ce faire, nous aurons besoin de plusieurs fonctions de la librairie `gprolog.h` :

- `Start_Prolog(int argc,char *argv[])` : cette fonction permet d'initialiser le moteur d'inférence de GNU Prolog ;
- `Stop_Prolog()` : cette fonction permet d'arrêter le moteur d'inférence de GNU Prolog ;
- `Find_Atom(String nom_du_predicat)` : cette fonction permet de retrouver la clé interne du prédicat "nom_du_predicat". Cette fonction renvoie un entier qui vaut -1 si le prédicat n'existe pas.
- `Pl_Query_Begin(Boolean b)` : cette fonction permet d'initialiser une requête Prolog. L'argument booléen doit valoir TRUE si on veut pouvoir libérer la mémoire utilisée pour exécuter cette requête après celle-ci.
- `Pl_Query_End(PL_RECOVER)` : cette fonction permet de terminer une requête. L'argument `PL_RECOVER` permet de libérer la mémoire allouée à cette requête.
- `Mk_Integer(int entier)` : cette fonction permet de créer des termes Prolog entiers. Il existe également des fonctions pour créer d'autres types de termes (string, booléen, ...). Afin de créer un terme non instancié, nous pouvons en outre utiliser la fonction `Mk_Variable()`. Toutes ces fonctions renvoient une variable de type `PlTerm` qui représente un terme Prolog.
- `Rd_Integer(PlTerm terme)` : cette fonction permet de récupérer les données calculées par le moteur d'inférence de GNU Prolog et retourne un entier. Il existe bien sûr d'autres fonctions qui permettent de récupérer d'autres types de données.

- `Pl_Query_Call(int func, int arite, PlTerm *arg)` : cette fonction permet d'effectuer une requête sur le predicat dont la clé interne est `func`, dont l'arité est `arite` et dont les arguments sont contenus dans le tableau de PlTerm `arg`. Cette fonction peut retourner
  - `PL_FAILURE` : une constante égale à 0 si la requête échoue ;
  - `PL_SUCCESS` : une constante égale à 1 si la requête réussit ;
  - `PL_EXCEPTION` : une constante égale à 2 si la requête lance une exception.

  Si la fonction nous a renvoyé `PL_SUCCESS`, nous pouvons utiliser la fonction `Pl_Query_Next_Solution()` pour demander une autre solution au moteur d'inférence.

Les listes sont des éléments relativement souvent utilisés en Prolog. Elles peuvent être représentées de 2 manières différentes :

$$[t_1, t_2, \ldots, t_n] \tag{3.1}$$

ou

$$[t_1|[t_2|[\ldots[t_n|[]]\ldots]]]. \tag{3.2}$$

Afin de créer des listes Prolog dans un programme C, nous pouvons utiliser les 2 instructions suivantes[3] :

- `Mk_Proper_List(int n, PlTerm *arg)` : cette fonction permet de créer une liste de PlTerm comme dans l'équation (3.1). Nous devons préciser dans l'argument `n` le nombre d'éléments que va contenir la liste. Le tableau de PlTerm `arg` contient alors les éléments de la liste.
- `Mk_List(PlTerm *arg)` : cette fonction permet de créer une liste de PlTerm comme dans l'équation (3.2). L'argument `arg` est un tableau de 2 éléments de type PlTerm : la tête de la liste et une *liste* qui représente la queue de la liste. Pour créer la liste vide [], il faut passer `NULL` comme argument à cette fonction.

Ces 2 fonctions renvoient un PlTerm qui contient la liste. Par exemple, si nous désirons créer un PlTerm qui contient la liste [1,2,3][4], nous pouvons utiliser les 2 jeux d'instructions suivants :

- ```
  PlTerm arg[3], liste ;
  arg[0] = Mk_Integer(1) ;
  arg[1] = Mk_Integer(2) ;
  arg[2] = Mk_Integer(3) ;
  liste = Mk_Proper_List(3,arg) ;
  ```

[3]Attention, pour utiliser ces 2 fonctions, le moteur d'inférence de GNU Prolog doit être démarré au préalable avec la fonction `Start_Prolog()`(voir ci-dessus)

[4]qui peut aussi être représenté [1[2|[3|[]]]]
  ```

- ```
  PlTerm arg1[2], arg2[2], arg3[2], liste1, liste2, liste ;
  arg1[0] = Mk_Integer(3) ;
  arg1[1] = Mk_List(NULL) ;
  liste1 = Mk_List(arg1) ;
  arg2[0] = Mk_Integer(2) ;
  arg2[1] = liste1 ;
  liste2 = Mk_List(arg2) ;
  arg3[0] = Mk_Integer(1) ;
  arg3[1] = liste2 ;
  liste = Mk_List(arg3) ;
  ```

Remarquons que certains compilateurs C (dont gcc) traitent les fonctions `main()` d'une manière particulière ce qui a pour effet de produire du code incompatible avec GNU Prolog. C'est pourquoi nous devons encapsuler/emballer[5] le code de notre fonction `main()` dans une autre fonction qui sera elle-même appelée par `main()`.

```
int Main_Wrapper(){
        ⋮
}
int main(){
      return (Main_Wrapper()) ;
}
```

3.4 Système expert fiscal

Dans cette section, nous précisons la manière dont les différentes technologies présentées auparavant vont être mises en oeuvre afin de pouvoir implémenter le système expert fiscal sur un serveur Web.

L'interface d'acquisition des données de l'utilisateur sera une interface web rédigée en html/php. Cette interface sera hébergée sur un serveur Apache (voir Annexe A pour la procédure d'installation). Comme nous l'avons vu, la partie 1 de la déclaration d'impôt est composée de 12 cadres. Nous aurons donc 12 pages web différentes qui permettront à l'utilisateur de rentrer des données. Chacune de ces pages sera déclarée comme un formulaire html et les données collectées seront envoyées au serveur Apache via la méthode POST :

```
<FORM method="POST">.
```

L'utilisation du PHP va nous permettre de traiter les différentes pages de manière dynamique et de stocker les variables dans une base de données MySQL.

[5] Wrap en anglais
```

Nous avons créé une table dans la base de données pour chaque cadre. Le numéro national du contribuable servira, pour chaque table, d'identifiant primaire c'est pourquoi nous utiliserons une variable de session pour le numéro national. Cette variable de session permettra de conserver la valeur du numéro national encodé par l'utilisateur au cadre 1 pendant toute la navigation dans les autres cadres. Avant de pouvoir utiliser la variable de session, nous devons signaler le début de la session dans chaque cadre avec l'instruction suivante :

```
session_start() ;
```

Les autres colonnes de la table étant les champs que les contribuables doivent remplir pour compléter leur déclaration d'impôt.

Afin de valider la variable de session, l'utilisateur est invité à entrer son numéro national dans l'interface et à le confirmer en cliquant sur le bouton "*Entrer*" (voir figure 3.4). Dès que cela est fait, le système insert une ligne dans chaque table de la base de données avec comme identifiant ce numéro national alors que les autres champs sont remplis avec des valeurs par défaut.

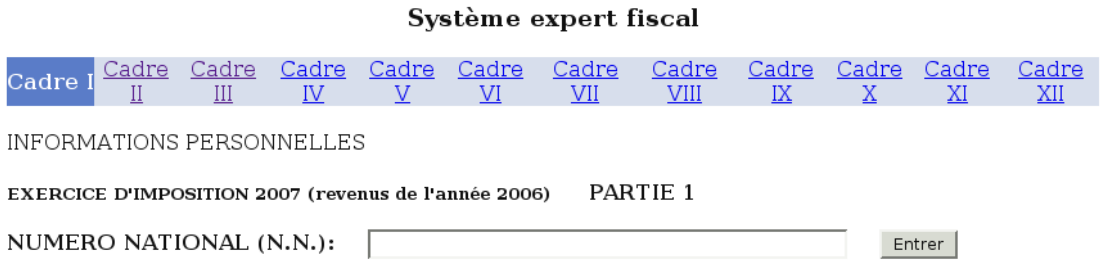

FIG. 3.4 – Interface graphique d'acquisition du numéro national de l'utilisateur

A chaque fois qu'un utilisateur aura rempli les champs d'un cadre, il cliquera sur le bouton "Enregistrer et passer au cadre suivant". A l'aide de quelques instructions PHP, nous mettons à jour les valeurs des variables qu'il vient de compléter/modifier dans la table correspondante et le cadre suivant s'affiche à l'écran. Par exemple pour le passage du cadre 1 au cadre 2, nous aurons les lignes suivantes au début du fichier `cadre2.php` :

```
1  mysql_connect(localhost, root, test) or die("erreur connexion");
2  mysql_select_db("sef");
3
4  $num_nat=$_SESSION['num_nat'];
5
6  if(isset($_POST['save']))
7          {
```

```
8                   $type_titulaire=$_POST['type_titulaire'];
9                   $num_compte=$_POST['num_compte'];
10                  $num_tel=$_POST['num_tel'];
11                  mysql_query("UPDATE CADRE1
12                              SET NumeroNational = '$num_nat',
13                              TypeTitulaire = '$type_titulaire',
14                              NumCompte = '$num_compte',
15                              NumTel = '$num_tel'
16                              WHERE NumeroNational = '$num_nat'");
17          }
```

Les 2 premières lignes permettent de se connecter à la base de données nommées `sef`. Les variables `$_SESSION['num_nat']`, `$_POST['type_titulaire']`, `$_POST['num_tel']` et `$_POST['num_compte']` représentent les valeurs rentrées par l'utilisateur au cadre 1 respectivement pour le numéro national, le type de titulaire, le numéro de téléphone et le numéro de compte. A la ligne 6, nous vérifions que la variable `$_POST['save']`, une variable cachée du formulaire de cadre1.php, possède une valeur. Cela nous permet de n'effectuer la mise à jour dans la base de données que si l'utilisateur a cliqué sur "Enregistrer et passer au cadre suivant" et pas si celui-ci navigue à travers les différents cadres à l'aide des boutons en haut de la page (voir figure 3.4).

Le fait de stocker les valeurs rentrées par l'utilisateur dans une base de données permet d'apporter plus de confort à l'utilisateur du système expert. En effet, la première fois qu'une personne utilise le système, il doit rentrer son numéro national. Celui-ci sera stocké dans une variable de session. Au fur et à mesure que l'utilisateur complètera des cadres, le système ajoutera des informations dans la base de données avec (pour rappel) le numéro national comme identifiant. Par la suite, si l'utilisateur décide de revenir en arrière d'un ou plusieurs cadres, avant d'afficher la page, le système va faire une requête sur la base de données pour voir si il a déjà rempli les champs de ce cadre. Si c'est le cas, on affichera les anciennes valeurs dans les champs correspondants. Sinon, on laissera les champs vides (valeurs par défaut).

Comme nous l'avons déjà signalé ci-dessus, les valeurs entrées par l'utilisateur du système expert seront enregistrées dans une base de données. Celle-ci a été créée à l'aide de l'interface *PhpMyAdmin*[8] et contient 12 tables, une pour chaque cadre de la déclaration d'impôt. Le code généré par *PhpMyAdmin* lors de l'exportation de cette base de données peut être trouvé à l'annexe B.4.

Une fois que tous les cadres auront été remplis, nous lancerons le calcul de l'impôt via CGI(voir section 3.3.2). Pour ce faire, nous récupérons toutes les données de la base de données et les stockons dans le formulaire du fichier `calcul.php` sous forme de variables cachées. Lorsqu'on cliquera sur le bouton "*Lancer le calcul de l'impôt*", le programme principal s'exécutera. Le fichier `calcul.php` peut être consulté à l'annexe B.2

Comme nous l'avons expliqué plus haut, le programme va commencer par extraire les données nécessaires au calcul de l'impôt. Ensuite, nous devrons transformer les variables en termes Prolog à l'aide des fonctions présentées à la section 3.3.3.

Après avoir préparé les termes Prolog, nous devrons simplement chercher les prédicats qui vont nous permettre de calculer l'impôt des personnes physiques et lancer des requêtes sur ces prédicats.

Une fois en possession des résultats, il faudra construire le code html que le script renverra au serveur afin que le *"client"* puissent les interpréter.

Le script CGI écrit en C pourra être consulté par le lecteur à l'annexe B.3.

Remarquons tout de même qu'il aurait été possible d'appeler le programme Prolog en utilisant d'autres technologies que le CGI pour obtenir une plus grande interactivité entre l'utilisateur et le système expert. Par exemple, le programme Prolog aurait pû être exécuté dans un thread tournant sur le serveur Web et dialogant plus directement avec l'interface utilisateur en utilisant par exemple la technologie *PiLLoW*[6] (voir [9] pour plus d'informations). Néanmoins, la solution du CGI, malgré son manque d'interactivité, nous a semblé plus simple à mettre en oeuvre.

[6]Programming in (Constraints) Logic Languages on the Web

Chapitre 4

Calcul de l'impôt des personnes physiques

Dans ce chapitre, nous développons la démarche qui nous a permis de construire la méthode de calcul de l'impôt des personnes physiques pour l'exercice d'imposition 2007. Nous expliquerons les règles à suivre et pour chacune des celles-ci, nous préciserons la manière dont nous l'avons modélisée en Prolog. L'entièreté du code Prolog pourra être consultée par le lecteur à l'annexe B.1.

Remarquons d'ores et déjà que les nombres utilisés pour représenter des sommes d'argent seront, la plupart du temps, multiplié par 100. En effet, GNU Prolog étant un solveur de contraintes à domaine fini, il n'est pas possible d'effectuer des calculs sur des réels. La multiplication par un facteur 100 nous permettra tout de même de garder une précision au centime près. Par exemple, le montant 522,89 EUR sera représenté par le nombre 52289 alors que 5000 EUR sera représenté par 500000.

4.1 Conditions sur l'impôt des personnes physiques

L'impôt n'est pas dû lorsque les revenus consistent exclusivement en pensions, allocations de chômage et prépensions à condition que le total des revenus ainsi perçus ne dépasse pas le maximum des allocations légales de chômage. Pour l'exercice d'imposition 2007, ce maximum s'élève à **12.618,06 EUR**.

L'impôt n'est pas dû non plus dans le cas où le contribuable a perçu en 2006 exclusivement des revenus :

- soit d'allocations de chômage (y compris le complément d'ancienneté accordé aux chômeurs âgés) si le montant de ces allocations ne dépasse pas la somme de **13.869,18 EUR** ;
- soit d'indemnités légales d'assurance maladie ou invalidité qui n'excèdent pas les dix neuvièmes du montant maximum des allocations légales de chômage à savoir

$12.618,06 \times 10/9 =$ **14.020,07 EUR**.

Modélisation

Nous pouvons distinguer 3 cas où le contribuable n'est pas redevable de l'impôt. Celui où il touche uniquement des allocations de chômage (lignes 6 à 9), celui où il touche uniquement des indemnités légales d'assurance maladie ou invalidité (lignes 10 à 13) et celui ou ses revenus sont composés uniquement de pensions, prépensions et allocations de chômage (lignes 14 à 17).
Le prédicat `impot_pas_du` nous renverra la réponse `yes` si l'impôt n'est pas dû et `no` dans le cas contraire.

```
6   impot_pas_du(Allocation_chomage, Pension, Prepension,Indemnite, Autre_revenu):-
7           Autre_revenu #= 0,
8           Pension + Prepension + Indemnite #= 0,!,
9           Allocation_chomage #=< 1386918.
10  impot_pas_du(Allocation_chomage, Pension, Prepension,Indemnite, Autre_revenu):-
11          Autre_revenu #= 0,
12          Pension + Prepension + Allocation_chomage #= 0,!,
13          Indemnite #=< 1402007.
14  impot_pas_du(Allocation_chomage, Pension, Prepension, Indemnite, Autre_revenu):-
15          Autre_revenu #= 0,
16          Indemnite #= 0,!,
17          Allocation_chomage + Pension + Prepension #=< 1261806.
```

4.2 Montant exempté d'impôt

Du revenu net imposable, une certaine quotité n'est pas imposée. En effet, tout contribuable a droit à un montant exempté d'impôt qui, dans certains cas, peut encore être augmenté :

- pour les enfants, enfants gravement handicapés ou autres personnes à charge ;
- pour les enfants qui n'ont pas atteint l'âge de 3 ans le 1er janvier 2007 et pour lesquels on n'a pas déduit de frais de garde ;
- pour un contribuable qui est imposé isolément[1] et qui a un ou plusieurs enfants à charge ;
- pour les personnes mariées ou les cohabitants légaux depuis 2006 dont un des conjoints n'avait pas de ressources nettes supérieures à **2610,00 EUR** ;
- pour les ascendants et les collatéraux jusqu'au deuxième degré qui font partie du ménage du contribuable ;

[1] Nous donnerons plus tard les critères pour qu'un contribuable soit considéré comme isolé

- pour un contribuable gravement handicapé, son conjoint ou d'autres personnes handicapées à charge.

L'impôt des personnes physiques dû est égal à la différence entre, d'une part, l'impôt calculé sur le revenu net imposable et, d'autre part, l'impôt calculé sur le montant exempté d'impôt.

4.2.1 Montant exempté d'impôt pour chaque contribuable

Chaque contribuable a droit à un montant exempté d'impôt de **5.940,00 EUR**. Ce montant est augmenté de **1260,00 EUR** si le contribuable est handicapé.

Modélisation

Lors du calcul du montant exempté d'impôt, ces sommes seront attribuées d'office à chaque contribuable. Autrement dit, tout contribuable bénificiera au minimum de 5.940,00 EUR (7.200,00 EUR si il est handicapé) de montant exempté d'impôt. Cette somme sera additionnée avec les éventuels bonus auxquels le contribuable a droit.

4.2.2 Montant exempté d'impôt pour enfant à charge

Pour les enfants à charge, le montant exempté d'impôt s'élève à :

- **1.260,00 EUR** pour 1 enfant ;
- **3.260,00 EUR** pour 2 enfants ;
- **7.280,00 EUR** pour 3 enfants ;
- **11.770,00 EUR** pour 4 enfants ;

Si il y a plus de 4 enfants à charge, le montant exempté d'impôt est de 11.770,00 EUR plus 4.490,00 EUR pour chaque enfant à charge au dessus du quatrième.
Ce montant est encore majoré de **470,00 EUR** pour chaque enfant qui, au 1er janvier 2007, n'avait pas atteint l'âge de 3 ans et pour lequel on n'a pas déduit de frais de garde. Les enfants gravement handicapés comptent pour 2 enfants à charge. Signalons également qu'un enfant décédé en 2006 est considéré comme étant à charge pour l'exercice d'imposition 2007.

Modélisation

Le prédicat `montant_exempte_enfant` va nous permettre de calculer le montant exempté d'impôt supplémentaire auquel a droit un contribuable en fonction du nombre d'enfants qu'il a à sa charge. Si il y a des enfants handicapés, il suffira de les compter pour 2.

```
161  montant_exempte_enfant(0,0).
162  montant_exempte_enfant(1,126000).
163  montant_exempte_enfant(2,326000).
```

```
164   montant_exempte_enfant(3,728000).
165   montant_exempte_enfant(4,1177000).
166   montant_exempte_enfant(Nb,Montant):-
167           Nb #> 4,
168           Nb_Enfant_Supp #= Nb-4,
169           Montant #= 1177000 + Nb_Enfant_Supp*449000.
```

Le bonus supplémentaire accordé pour des enfants de moins de 3 ans sera calculé par le prédicat suivant :

```
191   montant_exempte_enfant_3ans(N,Montant):-
192           Montant #= N * 47000.
```

4.2.3 Montant exempté d'impôt pour personnes à charge

Pour les personnes à charge (ascendants, collatéraux jusqu'au deuxième degré inclus et parents adoptifs) le montant exempté d'impôt est augmenté de **1260,00 EUR**. Les personnes gravement handicapées comptent pour deux. Si une ou plusieurs de ces personnes a plus de 65 ans, l'augmentation est de **2520,00 EUR**.

Modélisation

Les bonus accordés pour les personnes à charge seront calculés à l'aide des prédicats suivants :

```
199   montant_exempte_pers_charge(N,Montant):-
200           Montant #= N * 126000.
201   montant_exempte_pers_charge_65(N,Montant):-
202           Montant #= N * 252000.
```

4.2.4 Montant exempté d'impôt pour un contribuable qui est imposé isolément et qui a un ou plusieurs enfants à charge

Un contribuable qui est imposé isolément et qui a un ou plusieurs enfants à charge bénéficie d'un montant exempté d'impôt supplémentaire de **1.260,00 EUR**.

Un contribuable est considéré comme isolé pour l'exercice d'imposition 2007 si :

- au 1er janvier 2007, il n'est ni marié, ni cohabitant légal ;
- au 1er janvier 2007, il est veuf ou veuve ;
- il s'est marié ou a déposé un déclaration de cohabitation légale en 2006 ;
- il était séparé de fait avant le 1er janvier 2006 ;
- il est divorcé (ou assimilé[2]) ou séparé de corps en 2006 ;

[2]fin de la cohabitation légale

- il est le conjoint / cohabitant légal d'un fonctionnaire auprès d'une institution internationale qui a bénéficié, en 2006, de revenus professionnels dépassant 8.560,00 EUR, immunisé à la suite d'une convention internationale.

Modélisation

Nous devons d'abord construire un prédicat qui nous permettra de déterminer si un contribuable est considéré comme isolé et si il peut donc bénéficier de ce bonus. Toutes les informations nécessaires sont présentes dans le cadre II (Renseignement personnel et charges de famille). Nous allons représenter les cases qui ont été cochées dans ce cadre sous forme d'une liste dont les éléments seront les codes correspondants. Par exemple, un célibataire doit cocher la case sous le code 1001. L'état civil de celui-ci sera donc représenté par la liste [1001]. Par contre, si un contribubale s'est marié en 2006 et que les ressources nettes de son conjoint ne dépassent pas 2.610 EUR, il devra cocher les cases 1002, 1003 et 1004. Son état civil sera donc représenté par la liste [1002,1003,1004].

Voici le prédicat qui permet de déterminer si un contribuable peut être considéré comme isolé.

```
248  isole([H|T]):- H = 1001. /* 1001=code pour celibataire */
249  isole([H|T]):- H = 1010. /* 1010=code pour veuf/veuve */
250  isole([H1,H2|T]):-
251          H1 = 1002, /* 1002=code pour marie */
252          H2 = 1003. /* 1003=code pour marie depuis 2006*/
253  isole([H1,H2|T]):-
254          H1 = 1006, /* 1006=code pour cohabitant legal */
255          H2 = 1007. /* 1007=code pour cohabitant legal depuis 2006*/
256  isole([H1|T]):-
257          H1 = 1018, /* 1018=code pour separe de faits */
258          T = []. /* la separation de fait n'a pas eu lieu en 2006*/
259  isole([H1|T]):-
260          H1 = 1014. /* 1014=code pour divorce */
261  isole([H1|T]):-
262          H1 = 1016. /* 1016=code pour separe de corps */
263  isole([H1,H2|T]):-
264          H1 = 1002, /* 1002=code pour marie */
265          H2 = 1021. /* 1021=code pour marie a un fonctionnaire d'une
266                        institution internationale*/
267  isole([H1,H2|T]):-
268          H1 = 1006, /* 1006=code pour cohabitant legal */
269          H2 = 1021. /* 1021=code pour cohabitant legal d'1 fonctionnaire
270                        d'une institution internationale*/
```

A l'aide de ce prédicat, nous pouvons désormais calculer le bonus accordé à un isolé

qui a des enfants à charge :

```
224   montant_exempte_bonus_isole([H|T],Nb_enfant,Montant):-
225           Nb_enfant #> 0,
226           isole([H|T]),
227           Montant #= 126000.
228   montant_exempte_bonus_isole([H|T],Nb_enfant,0).
```

Le premier argument du prédicat est la liste de l'état civil. La ligne 228 fait en sorte qu'un isolé n'ayant aucun enfant à charge ne reçoive pas de bonus.

4.2.5 Montant exempté d'impôt pour cause de mariage en 2006

Un contribuable qui s'est marié ou a fait une déclaration de cohabitation légale en 2006 a droit à un montant exempté d'impôt supplémentaire de 1.260,00 EUR si, en 2006, son conjoint n'a pas bénéficié de plus de 2.610,00 EUR de ressources nettes.

Modélisation

Le bonus est calculé à l'aide des prédicats suivants :

```
229   montant_exempte_bonus_marie([H1,H2,H3|_],Montant):-
230           H1 = 1002, /* 1002=code pour marie */
231           H2 = 1003, /* 1003=code pour marie depuis 2006*/
232           H3 = 1004, /* 1004=code pour le conjoint a des ressources nettes
233                         de moins de 2610 euro*/
234           Montant #= 126000.
235   montant_exempte_bonus_marie(_,0).
```

Si les conditions ne sont pas remplies (ligne 235), le bonus vaut 0 EUR.

4.2.6 Répartition des montants exemptés d'impôt entre les conjoints

Lors d'une déclaration commune, le conjoint qui a le revenu imposable le plus bas n'a droit qu'à un montant exempté d'impôt de **5.940,00 EUR**. Ce montant est majoré de **1230,00 EUR** si il est handicapé. Les montants exemptés d'impôt restants sont toujours attribués au conjoint qui a le revenu le plus élevé.

Si le montant exempté d'impôt d'un conjoint est plus élevé que ses revenus imposables, la partie non utilisée est cumulée avec celle de l'autre conjoint.

Par exemple, si dans un ménage sans enfant à charge, le revenu net imposable du mari est de 15.500,00 EUR et celui de la femme de 2.975,00 EUR, alors les 5.940,00 EUR exemptés d'impôt auxquels la femme a droit sont limités à 2.975,00 EUR (son revenu). Le reste de la somme (5.940,00 - 2.975,00 = 2.965,00 EUR) est cumulée avec le montant

exonéré d'impôt du mari. Le mari bénéficie donc de 8.905,00 EUR (2.965,00 + 5.940,00) exemptés d'impôt.

Modélisation

Dans le cas d'une déclaration commune, la répartition des montants exemptés d'impôt peut se faire à l'aide des prédicats suivants :

```
286  repartition_montant_exempte(Revenus1,Revenus2,
287                              MontantExempteInitial1,MontantExempteInitial2,
288                              MontantExempteFinal1,MontantExempteFinal2):-
289          Revenus1 #>= MontantExempteInitial1,
290          Revenus2 #>= MontantExempteInitial2,
291          MontantExempteFinal1 #= MontantExempteInitial1,
292          MontantExempteFinal2 #= MontantExempteInitial2.
293  repartition_montant_exempte(Revenus1,Revenus2,
294                              MontantExempteInitial1,MontantExempteInitial2,
295                              MontantExempteFinal1,MontantExempteFinal2):-
296          Revenus1 #>= MontantExempteInitial1,
297          Revenus2 #< MontantExempteInitial2,
298          MontantExempteFinal1 #= MontantExempteInitial1 +
299          MontantExempteInitial2 - Revenus2,
300          MontantExempteFinal2 #= Revenus2.
301  repartition_montant_exempte(Revenus1,Revenus2,
302                              MontantExempteInitial1,MontantExempteInitial2,
303                              MontantExempteFinal1,MontantExempteFinal2):-
304          Revenus1 #< MontantExempteInitial1,
305          Revenus2 #>= MontantExempteInitial2,
306          MontantExempteFinal1 #=  Revenus1,
307          MontantExempteFinal2 #= MontantExempteInitial2 +
308          MontantExempteInitial1 - Revenus1.
309  repartition_montant_exempte(Revenus1,Revenus2,
310                              MontantExempteInitial1,MontantExempteInitial2,
311                              MontantExempteFinal1,MontantExempteFinal2):-
312          Revenus1 #< MontantExempteInitial1,
313          Revenus2 #< MontantExempteInitial2,
314          MontantExempteFinal1 #= MontantExempteInitial1 +
315          MontantExempteInitial2 - Revenus2,
316          MontantExempteFinal2 #= Revenus2.
```

Les variables se terminant par un 1 sont celles relatives au conjoint qui a les revenus les plus élevés. On voit que dans le cas où chaque conjoint a un montant exempté d'impôt supérieur à son revenu (lignes 309 à 316), on reporte le surplus sur le conjoint qui a le revenu le plus élevé.

4.3 Frais professionnels forfaitaires

Lors du calcul de l'impôt des personnes physiques, tout contribuable qui déclare des revenus professionnels a le droit de déduire des **frais professionnels**. Dans la plupart des cas, les salariés, les appointés, ... ont du mal à évaluer et à prouver leurs frais professionnels c'est pourquoi il est prévu un réglement forfaitaire qui permet d'obtenir les frais professionnels que l'on va pouvoir déduire à partir de nos revenus.

Les frais professionnels forfaitaires sont calculés à l'aide du tableau ci-dessous.

Lorsque le revenu brut dépasse	*Sans dépasser*	*Les frais professionnels forfaitaires s'élèvent à*	*Plus sur l'excédent*
0	4.790,00	26,1%	-
4.790,00	9.520,00	1.250,19	10,0%
9.520,00	15.850,00	1723,19	5,0%
15.850,00	-	2.039,69	3,0%

TAB. 4.1 – Taux du calcul des frais forfaitaires déductibles pour l'exercice d'imposition 2007

Les frais forfaitaires déductibles ne peuvent jamais dépasser la somme de 3.230,00 EUR.

Par exemple si un contribuable possède des revenus professionnels brut de 20.000 EUR, nous obtenons :

$$\text{Frais déductibles} : 2.039,69 + (20.000 - 15.850) \times 3\% = 2.164,19 \text{ EUR} .$$

Modélisation

Le tableau ci-dessus peut être modélisé par les prédicats suivants :

```
53  frais_professionnel_forfaitaire(RevenusBrut, Frais):-
54          RevenusBrut #=< 479000,
55          Frais #= RevenusBrut * 261 // 1000.
56  frais_professionnel_forfaitaire(RevenusBrut, Frais):-
57          RevenusBrut #> 479000,
58          RevenusBrut #=< 952000,
59          Frais #= 125019 + ((RevenusBrut - 479000) // 10).
60  frais_professionnel_forfaitaire(RevenusBrut, Frais):-
61          RevenusBrut #> 952000,
62          RevenusBrut #=< 1585000,
63          Frais #= 172319 + ((RevenusBrut - 952000) * 5 // 100).
64  frais_professionnel_forfaitaire(RevenusBrut, Frais):-
65          RevenusBrut #> 1585000,
66          Frais1 #= 203969 + ((RevenusBrut - 1585000) * 3 // 100),
```

```
67          limiter_somme_superieur(Frais1, 323000, Frais).
```

Nous verrons par la suite que le fait de limiter un somme supérieurement et inférieurement revient à plusieurs reprises durant le calcul de l'impôt c'est pourquoi nous avons créé un prédicat qui permet de le faire. Ce prédicat sera utilisé plusieurs fois par la suite.

```
426  limiter_somme_superieur(Montant, Limite, Nouveau_montant):-
427          Montant #=< Limite,
428          Nouveau_montant #= Montant.
429  limiter_somme_superieur(Montant, Limite, Nouveau_montant):-
430          Montant #> Limite,
431          Nouveau_montant #= Limite.
432  limiter_somme_inferieur(Montant, Limite, Nouveau_montant):-
433          Montant #>= Limite,
434          Nouveau_montant #= Montant.
435  limiter_somme_inferieur(Montant, Limite, Nouveau_montant):-
436          Montant #< Limite,
437          Nouveau_montant #= Limite.
```

Par exemple, pour le prédicat `limiter_somme_superieur`, soit le montant ne dépasse pas la limite spécifiée par la variable `Limite` et alors le montant reste inchangé, soit cette limite est dépassée et on doit modifier le montant (ligne 431).

4.3.1 Forfait special long déplacement

Si la distance entre le domicile et le lieu de travail du contribuable dépasse 75 km, ce dernier a droit à un forfait supplémentaire. Ce forfait est calculé à l'aide du tableau suivant.

Distance (trajet simple)	Forfait supplémentaire
De 75 à 100 km	75 EUR
De 101 km à 125 km	125 EUR
Plus de 125 km	175 EUR

TAB. 4.2 – Tableau d'attributions des forfaits pour longs déplacements

Modélisation

Nous n'avons pas eu besoin de modéliser cette loi fiscale dans la mesure où le contribuable doit calculer son forfait long déplacement lui-même avant de le transcrire dans sa déclaration d'impôt.

4.4 Calcul de l'impôt des personnes physiques

L'impôt des personnes physiques est calculé aussi bien sur le revenu net imposable que sur les montants exemptés d'impôt. Comme nous l'avons déjà signalé, l'impôt des personnes physiques dû est égal à la différence entre le résultat de ces deux calculs.

L'impôt est calculé à l'aide du tableau ci-dessous.

plus de	*mais pas plus de*	*l'impôt dû s'élève à*	*plus du supérieur*
-	7.290,00	25%	-
7.290,00	10.380,00	1.822,50	30,0%
10.380,00	17.300,00	2.749,50	40,0%
17.300,00	31.700,00	5.517,50	45,0%
31.700,00	-	11.997,50	50,0%

TAB. 4.3 – Taux de l'impôt des personnes physiques pour l'exercice d'imposition 2007

Par exemple, si un contribuable a un revenu net imposable de 44.510,00 EUR. L'impôt est calculé à l'aide de la dernière ligne du tableau.

$$\begin{aligned} \textit{Impôt} &= 11.997,50 + ((44.510,00 - 31.700,00) \times 50\%) \\ &= 11.997,50 + (12810 \times \frac{1}{2}) \\ &= 18402,50 \end{aligned}$$

La même méthode est appliquée pour calculer l'impôt sur le montant exempté d'impôt.

Modélisation

Le calcul de l'impôt sur une somme pour l'exercice d'imposition 2007 peut être calculé avec les prédicats suivants :

```
25  impot(Montant, Impot):-
26          Montant #=< 729000,
27          Impot #= (Montant * 25) // 100.
28  impot(Montant, Impot):-
29          Montant #> 729000,
30          Montant #=< 1038000,
31          Impot #= 182250 +(((Montant - 729000)* 30)// 100).
32  impot(Montant, Impot):-
33          Montant #> 1038000,
34          Montant #=< 1730000,
35          Impot #= 274950 +(((Montant - 1038000)* 40)// 100).
36  impot(Montant, Impot):-
37          Montant #> 1730000,
```

```
38          Montant #=< 3170000,
39          Impot #= 551750 +(((Montant - 1730000)* 45)// 100).
40  impot(Montant, Impot):-
41          Montant #> 3170000,
42          Impot #= 1199750 +(((Montant - 3170000)* 50)// 100).
43
```

On remarque que, suivant la tranche dans laquelle se trouve le montant sur lequel nous devons calculer l'impôt, la requête s'unifiera avec tel ou tel prédicat.

4.5 Impôt sur les rentes alimentaires perçues

Les rentes alimentaires perçues renseignées dans le cadre 6 de la déclaration sont considérées comme des revenus divers qui sont imposés séparément. Il existe trois types de rentes alimentaires perçues et chacun de ces types doit être imposé séparement à concurrence de 80% suivant le tarif progressif de la section 4.4.

Modélisation

Pour calculer l'impôt dû pour l'ensemble des rentes alimentaires, nous sommerons les 3 montants obtenus à l'aide du prédicat `impot` à partir de 80% des différentes rentes alimentaires.

4.6 Réduction d'impôt

L'impôt dû (calculé à la section 4.4) peut éventuellement être diminué à l'aide de réductions d'impôt qui peuvent être obtenues dans les différents cas suivants.

4.6.1 Réduction pour l'épargne-logement

Pour l'épargne logement (voir cadre VIII,D.1. et VIII,E.1.), il est accordé une réduction qui est calculée en fonction du taux le plus élevé de l'impôt dû par le contribuable. Si les montants qui entrent en ligne de compte pour la réduction portent sur plus d'un taux d'impôt, c'est le taux correspondant qui est pris en compte pour chaque partie de ces montants.

Par exemple si le mari d'un ménage possède un revenu net imposable de $22.310,00$ EUR dont $1.000,00$ EUR entrent en compte pour la réduction pour l'épargne logement, la différence calculée vaut $22.310,00 - 1.000,00 = 21.310,00$ EUR. Les 2 montants $22.310,00$ EUR et $21.310,00$ EUR se trouvent tous les deux dans la tranche de tarif des 45%. La réduction s'éleve donc à $1.000,00$ EUR $\times 45\% = 450,00$ EUR.
D'un autre côté, la femme de ce ménage possède un revenu net imposable de $17.500,00$

EUR dont 744,00 EUR entrent en compte pour la réduction liée à l'épargne logement. La différence de ces 2 montants vaut $17.500,00 - 744,00 = 16.756,00$ EUR. Les montans 17.500,00 EUR et 16.756,00 EUR se trouvent dans des tranches de tarifs différents. La réduction est donc obtenue comme suit :

$17.500,00 - 17.300 = 200,00$	$\times 45\%$	$= 90,00$
$17.300,00 - 16.756,00 = 544,00$	$\times 40\%$	$= 217,60$
Réduction		$= 307,60$ EUR

Il existe une limitation de la somme entrant en compte pour la réduction accordée pour une épargne logement. La limite est calculée en additionnant 15% de la première tranche de 1600 EUR des revenus professionnels et 6% de la somme excédentaire.

Modélisation

En premier lieu nous avons besoin, pour modéliser cette réduction, d'un prédicat nous permettant de connaître le taux dans lequel nous nous trouvons en fonction d'une somme. Il est également intéressant de connaître les bornes inférieures et supérieures qui délimitent ce taux. Voici le prédicat qui va nous permettre de le faire :

```
74  taux(Montant, BorneInf, BorneSup, Taux):-
75          Montant #=< 729000,
76          BorneInf #= 0,
77          BorneSup #= 729000,
78          Taux #= 25.
79  taux(Montant, BorneInf, BorneSup, Taux):-
80          Montant #> 729000,
81          Montant #=< 1038000,
82          BorneInf #= 729000,
83          BorneSup #= 1038000,
84          Taux #= 30.
85  taux(Montant, BorneInf, BorneSup, Taux):-
86          Montant #> 1038000,
87          Montant #=< 1730000,
88          BorneInf #= 1038000,
89          BorneSup #= 1730000,
90          Taux #= 40.
91  taux(Montant, BorneInf, BorneSup, Taux):-
92          Montant #> 1730000,
93          Montant #=< 3170000,
94          BorneInf #= 1730000,
95          BorneSup #= 3170000,
96          Taux #= 45.
97  taux(Montant, BorneInf, BorneSup, Taux):-
```

```
 98          Montant #> 3170000,
 99          BorneInf #= 3170000,
100          BorneSup #= 0,
101          Taux #= 50.
```

Par exemple, les lignes 85 à 90 représentent le fait que si un montant est compris entre 10.380 EUR et 17.300 EUR, il sera imposé au taux de 40%.

Le calcul de la réduction accordée pour des épargnes logements sera effectué grâce aux prédicats suivants :

```
341  reduction_epargne_logement(Revenu_net, Part_epargne, Reduction):-
342          taux(Revenu_net, _, _, Taux_revenu_net),
343          Difference #= Revenu_net - Part_epargne,
344          taux(Difference, _, _, Taux_difference),
345          Taux_revenu_net = Taux_difference,
346          Reduction #= (Part_epargne * Taux_revenu_net) // 100.
347  reduction_epargne_logement(Revenu_net, Part_epargne, Reduction):-
348          taux(Revenu_net, Min, _, Taux_revenu_net),
349          Difference #= Revenu_net - Part_epargne,
350          taux(Difference, _, _, Taux_difference),
351          Taux_revenu_net #\= Taux_difference,
352          Difference1 #= Revenu_net - Min,
353          Part_epargne1 #= Part_epargne - Difference1,
354          Reduction #= ((Difference1 * Taux_revenu_net) // 100) + Red1,
355          reduction_epargne_logement(Min, Part_epargne1, Red1).
```

Les lignes de 341 à 346 modélisent le cas où les taux à envisager sont les mêmes alors que les lignes 347 à 355 gèrent le cas où les taux sont différents. Il faut alors calculer la réduction pour chaque tranche. Cela est effectué grâce à un appel récursif au prédicat `reduction_epargne_logement` : la réduction totale est égale à la réduction sur la tranche la plus élevée plus la réduction sur le montant restant c'est-à-dire sur la/les tranches inférieures (voir ligne 354).

La terminaison de l'évaluation de ce prédicat est assurée car les arguments passés lors de l'appel récursif sont diminués à chaque fois. Nous sommes donc certains qu'à un moment, l'unification se fera avec le prédicat qui gère le cas où les taux sont identiques (au pire des cas, cela arrivera au taux minimum).

La limitation pour la réduction sera effectuée par le prédicat suivant :

```
361  limite_epargne_logement(Renumeration, Limite):-
362          Renumeration #>=  160000,
363          Tranche2 #= Renumeration - 160000,
```

```
364           Limite1 #= 24000 + (Tranche2 *6 // 100),
365           limiter_somme_superieur(Limite1,192000,Limite).
366   limite_epargne_logement(Renumeration, Limite):-
367           Renumeration #<  160000,
368           Limite #= Renumeration * 15 // 100.
```

4.6.2 Réduction pour l'épargne à long terme

La réduction pour l'épargne à long terme est calculée à un taux spécial moyen d'imposition. Ce taux est calculé à l'aide de la différence entre l'impôt sur le revenu net imposable globalement et l'impôt sur le montant exempté du **conjoint** ou de l'isolé. Le minimum est fixé à 30% et le maximum à 40%. En outre, le taux moyen d'imposition ne doit être calculé que pour un revenu imposable de 25.184, 57 EUR à 53.374, 98 EUR. Si le revenu est inférieur à 25.184, 57 EUR, le taux moyen vaut 30% et à partir d'un revenu de 53.374, 99 EUR, il est égal à 40%.

Pour illustrer cette réduction, prenons le cas d'un ménage sans enfant. Le montant exempté d'impôt de chacun des conjoints est donc de 5940, 00 EUR (voir section 4.2). Le mari possède un revenu net imposable globalement de 28.000, 00 EUR dont 1.200, 00 EUR entrent en compte pour l'épargne à long terme. Voici le calcul qui permet de déterminer la réduction accordée pour le mari :

impôt sur 28.000, 00EUR	=	10.332, 50 EUR
impôt sur 5.940, 00EUR	=	1485, 00 EUR
différence	=	8.847, 50 EUR

taux d'imposition moyen : $\frac{8.847{,}00 \times 100}{28.000{,}00} = 31, 598$

Le résultat obtenu est tronqué à un chiffre après la virgule. Le taux moyen est donc de 31, 5% et nous calculons que la réduction vaut 1.200, 00 EUR $\times 31.5\% = 378, 00$ EUR.
La femme quant à elle possède un revenu net imposable globalement de 11.160, 00 EUR dont 600, 00 EUR entrent en compte pour l'épargne à long terme. Nous calculons donc

impôt sur 11.160, 00EUR	=	3.061, 50 EUR
impôt sur 5.940, 00EUR	=	1.485, 00 EUR
différence	=	1.576, 50 EUR

taux d'imposition moyen : $\frac{1.576{,}00{,}00 \times 100}{11.160{,}00} = 14, 13$

Etant donné que le minimum autorisé est de 30%, la réduction accordée à la femme pour l'épargne à long terme sera donc 600, 00 EUR $\times 30\% = 180, 00$ EUR.

Modélisation

La réduction pour les épargnes long termes est calculées à l'aide des prédicats suivants :

```
376  reduction_epargne_long_terme(Revenu,Montant_exempte_CONJOINT,
377                               Part_epargne,Reduction):-
378          impot(Revenu,Impot_revenu),
379          impot(Montant_exempte_CONJOINT,Impot_montant_exempte_CONJOINT),
380          Tmp #= Revenu // 10,
381          Taux_imposition_moyen #= ((Impot_revenu - Impot_montant_exempte_CONJOINT)
382                                    * 100) // Tmp,
383          limiter_somme_superieur(Taux_imposition_moyen,400,Taux1),
384          limiter_somme_inferieur(Taux1,300,Taux),
385          Reduction #= (Part_epargne * Taux) // 1000.
386  reduction_epargne_long_terme(Revenu,Montant_exempte_CONJOINT,
387                               Part_epargne,Reduction):-
388          impot(Revenu,Impot_revenu),
389          impot(Montant_exempte_CONJOINT,Impot_montant_exempte_CONJOINT),
390          Impot_revenu #< Impot_montant_exempte_CONJOINT,
391          Reduction #= (Part_epargne * 300) // 1000.
```

On remarque que le taux est calculé en pourmille et non en pourcent afin de conserver une certaine précision. Comme nous sommes obligés d'avoir un taux compris entre 30 et 40%, nous utilisons les prédicats `limiter_somme_superieur` et `limiter_somme_inferieur`. Le prédicat de la ligne 386 gère le fait que l'impôt sur le revenu net peut être inférieur à l'impôt sur le montant exempté du conjoint. Dans ce cas, la soustraction de la ligne 381 échoue et nous calculons donc la réduction avec un taux de 30% via le prédicat de la ligne 386.

4.6.3 Réduction d'impôt pour des dépenses payées pour des prestations dans le cadre des agences locales pour l'emploi

Dans le cadre de la lutte contre le travail en noir, une réduction d'impôt est accordée pour les montants payés par un contribuable pour certaines prestations fournies par un chômeur inscrit dans une agence locale pour l'emploi. La réduction d'impôt qui s'applique ici est la réduction d'impôt pour épargne à long terme (voir section 4.6.2). Lorsque l'imposition est établie au nom des deux conjoints, les dépenses précitées sont réparties proportionnellement sur les revenus de chacun.

Modélisation

Nous utiliserons le prédicat de la section précédente (4.6.2) pour calculer les réductions accordées pour des dépenses payées pour des prestations dans le cadre des ALE.

4.6.4 Réduction d'impôt pour des dépenses pour des prestations payées avec des titres-services

Les titres-services sont un moyen de paiement par lequel une personne peut acheter des prestations de services qui portent principalement sur une aide ménagère. La réduction d'impôt est calculée avec un taux de 30%.

Remarquons que la somme des dépenses exposées à la section 4.6.3 et des dépenses pour des prestations payées avec des titres-services ne peut pas dépasser un montant de 2.310, 00 EUR par partenaire.

Modélisation

Pour calculer les réductions accordées dans le cadre des titres-services, nous utiliserons le prédicat suivant :

```
403  reduction_titre_service(Montant,Reduction):-
404          Reduction #= (Montant * 3)//10.
```

4.6.5 Réduction d'impôt pour le prêt gagnant-gagnant flamand

Le prêt gagnant-gagnant est un moyen d'inciter les personnes physiques à mettre à la disposition de PME établies en région flamande des moyens financiers. Une réduction d'impôt de 2, 5% avec un maximum de 1.250 EUR est accordée sur les montants prêtés. Si l'emprunteur ne peut pas rembourser sa dette, une réduction d'impôt de 30% sur le montant définitivement perdu avec un maximum de 15.000 EUR est accordée au prêteur. Cette réduction n'est accordée qu'aux habitants de la région flamande.

Modélisation

La réduction accordée pour les prêts gagnant-gagnant flamands est calculée avec les prédicats suivants :

```
1  reduction_pret_gagnant(Somme, Reduction):-
2          Reduction1 #= Somme * 25 // 1000,
3          limiter_somme_superieur(Reduction1,125000,Reduction).
4  reduction_pret_gagnant_perdu(Somme, Reduction):-
5          Reduction1 #= Somme * 3 // 10,
6          limiter_somme_superieur(Reduction1,1500000,Reduction).
```

On remarque de nouveau que la limitation de la réduction est effectuée avec le prédicat `limiter_somme_superieur`.

Le même principe sera appliqué pour les réductions des sections 4.6.6 et 4.6.7 et les crédits d'impôt des sections 4.8.3 et 4.8.4.

4.6.6 Réduction d'impôt pour les dépenses faites en vue d'économiser de l'énergie

Une réduction de 40% du montant des dépenses faites en vue d'économiser de l'énergie est accordée à chaque contribuable. Cette réduction est néanmoins limitée à 1.280, 00 EUR.

Modélisation

Nous utiliserons le même principe qu'à la section 4.6.5.

4.6.7 Réduction d'impôt pour les dépenses de rénovation dans une zone d'action positive des grandes villes

Dans le cadre de la rénovation d'une habitation située dans une zone d'action positive des grandes villes[3], la loi accorde une réduction d'impôt de 15% du coût de la dite rénovation. Toutefois, cette réduction est limitée à 640,00 EUR.

Modélisation

Nous utiliserons le même principe qu'à la section 4.6.5.

4.6.8 Réduction d'impôt pour l'acquisition d'un véhicule moins polluant

Afin d'inciter les contribuables à acquérir des véhicules moins polluants, une réduction d'impôt de 15% est accordée si le contribuable a acheté, lors de l'année 2006, un véhicule avec une émission maximale de 105 grammes de CO_2 par kilomètre. Cette réduction est limitée à 4190,00 EUR.
Par contre, si l'émission de CO_2 du nouveau véhicule est comprise entre 105 et 115 grammes par kilomètre, la réduction d'impôt est de 3% et ne peut pas dépasser 790,00 EUR.

Modélisation

Les réductions accordées pour l'achat de véhicules moins polluants peuvent être calculées à l'aide des prédicats suivants :

```
464  reduction_depense_vehicule_moins_105(Montant,Reduction):-
465          Reduction1 #= (Montant //100)* 15,
466          limiter_somme_superieur(Reduction1,419000,Reduction).
467  reduction_depense_vehicule_105_115(Montant, Reduction):-
468          Reduction1 #= (Montant //100)* 3,
469          limiter_somme_superieur(Reduction1,79000,Reduction).
```

[3]Commune ou partie délimitée d'une commune où l'habitat et le cadre de vie doivent être améliorés par des mesures spécifiques. Ces zones sont spécifiées dans un arrêté royale du 4 juin 2003.

Remarquons qu'aux lignes 465 et 468, lorsque que nous calculons la réduction suivant les taux en vigueur, nous effectuons la division en premier lieu. En effet, le coût d'un nouveau véhicule pouvant être considérable[4], si nous multiplions en premier lieu, nous risquons de dépasser la borne supérieure des entiers du moteur d'inférence.

4.6.9 Réduction d'impôt pour allocation de chômage

Si le revenu net imposable d'un contribuable ne dépasse pas 19.050 EUR et qu'il est composé *exclusivement* d'allocations de chômage, une réduction d'impôt de 2.006,88 EUR et de 1.718,76 EUR est accordée respectivement pour les personnes mariées et les isolés.

Si il y a d'autres sources de revenus que les allocations de chômage mais que le revenu net imposable ne dépasse pas 19.050 EUR, la réduction d'impôt est diminuée *proportionnellement* en fonction du rapport des allocations de chômage avec le total des revenus nets du ménage ou de l'isolé :

$$\text{Réduction proportionnelle pour un isolé} = 1.718,76 \text{ EUR} \times \frac{\text{allocations de chômage}}{\text{total des revenus nets}} = Y$$

$$\text{Réduction proportionnelle pour un ménage} = 2.006,88 \text{ EUR} \times \frac{\text{allocations de chômage}}{\text{total des revenus nets}} = Y$$

Si le revenu net imposable est supérieur à 19.050 EUR mais est inférieur à 23.780 EUR, alors la réduction d'impôt est diminuée *graduellement* :

$$Y \times \frac{\text{23.780 EUR} - \text{revenu net } \textbf{imposable} \text{ globalement}}{\text{4.730 EUR}}.$$ [5]

Le revenu net imposable est la différence entre le total des revenus nets et les montants déductibles.

Dans le cas d'un revenu net imposable supérieur ou égal à 23.780 EUR, aucune réduction d'impôt n'est accordée.

Modélisation

Etant donné que la réduction proportionnelle va être utilisée à plusieurs reprises, nous l'avons modélisé par le prédicat suivant :

```
584  reduction_proportionnelle(Base, Num, Den, Reduction):-
585          Num1 #= Num // 100,
586          Den1 #= Den // 100,
587          Base1 #= Base // 100,
588          Reduction #= Base1 * Num1 // Den1 * 100.
```

[4] Rappelons que nous multiplions déjà tous les montants par 100 !

[5] 23.780 EUR -19.050 EUR

Pour les mêmes raisons de risque d'overflow, nous divisons chaque variable par 100 avant d'effectuer le calcul final.

Les prédicats suivants permettent, à l'aide de la réduction proportionnelle, de calculer les réductions accordées pour les allocations de chômage.

```
490  reduction_impot_chomage(Etat_civil, Revenus, Revenus_imposable,
491                          Part_chomage, Reduction):-
492          Revenus_imposable #< 1905000,
493          Revenus_imposable = Part_chomage,
494          isole(Etat_civil),
495          Reduction #= 171876.
496  reduction_impot_chomage(Etat_civil, Revenus, Revenus_imposable,
497                          Part_chomage, Reduction):-
498          Revenus_imposable #< 1905000,
499          Revenus_imposable = Part_chomage,
500          \+isole(Etat_civil),
501          Reduction #= 200688.
502  reduction_impot_chomage(Etat_civil, Revenus, Revenus_imposable,
503                          Part_chomage, Reduction):-
504          Revenus_imposable #< 1905000,
505          Revenus_imposable #\= Part_chomage,
506          isole(Etat_civil),
507          reduction_proportionnelle(171876,Part_chomage,Revenus,Reduction).
508  reduction_impot_chomage(Etat_civil, Revenus, Revenus_imposable,
509                          Part_chomage, Reduction):-
510          Revenus_imposable #< 1905000,
511          Revenus_imposable #\= Part_chomage,
512          \+isole(Etat_civil),
513          reduction_proportionnelle(200688,Part_chomage,Revenus,Reduction).
514  reduction_impot_chomage(Etat_civil, Revenus, Revenus_imposable,
515                          Part_chomage, Reduction):-
516          Revenus_imposable #>= 1905000,
517          Revenus_imposable #< 2378000,
518          isole(Etat_civil),
519          reduction_proportionnelle(171876,Part_chomage,Revenus,Reduction1),
520          Num #= 2378000 - Revenus_imposable,
521          Den #= 2378000 - 1905000,
522          reduction_proportionnelle(Reduction1, Num, Den, Reduction).
523  reduction_impot_chomage(Etat_civil, Revenus, Revenus_imposable,
524                          Part_chomage, Reduction):-
525          Revenus_imposable #>= 1905000,
526          Revenus_imposable #< 2378000,
```

```
527           \+isole(Etat_civil),
528           reduction_proportionnelle(200688,Part_chomage,Revenus,Reduction1),
529           Num #= 2378000 - Revenus_imposable,
530           Den #= 2378000 - 1905000,
531           reduction_proportionnelle(Reduction1, Num, Den, Reduction).
532   reduction_impot_chomage(_, Revenus, Revenus_imposable, _, Reduction):-
533           Revenus_imposable #>= 2378000,
534           Reduction #= 0.
```

Dans tous ces prédicats (sauf le dernier qui n'accorde aucune réduction), nous devons faire une distinction entre les isolés et les autres contribubles. Cette distinction est faite à l'aide du prédicat `isolé` dont nous avons déjà parlé. Le prédicat \+ permet de nier un prédicat. Pour les revenus imposables de moins de 19.050 EUR, nous faisons également la différence entre les revenus uniquement constitués des allocations de chômage et les revenus mixtes à l'aide des contraintes aux lignes 493, 499, 505 et 511.

4.6.10 Réduction d'impôt pour indemnités légales d'assurance maladie-invalidité

Si un contribuable possède un revenu net imposable inférieur à 19.050 EUR qui est composé exclusivement d'indemnités légales d'assurance maladie-invalidité, celui-ci bénéficie d'une réduction d'impôt de 2.206,32 EUR. Cette réduction doit être diminuée si ses revenus imposables ne sont pas uniquement constitués d'indemnités légales d'assurances maladie-invalidité. En effet, dans le cas de revenus mixtes et si le revenu imposable ne dépasse pas 19.050 EUR, la réduction d'impôt est diminuée *proportionnellement* en fonction du rapport entre les indemnités et le total des revenus nets du contribuable.

$$\text{Réduction} = 2.206,32 \text{ EUR } \times \frac{\text{indemnités}}{\text{total des revenus nets}} = Y.$$

Si le total des revenus nets est compris entre 19.050 EUR et 38.090 EUR, la réduction d'impôt est diminuée *graduellement* :

$$\text{Réduction} = \frac{1}{3}Y + \left(\frac{2}{3}Y \times \frac{38.090 \text{ EUR } - \text{revenu net } \textbf{imposable}}{19.040 \text{ EUR }^{6}}\right). \tag{4.1}$$

Si le revenu net imposable est supérieur ou égal à 38.090 EUR, la réduction est calculée comme suit :

$$\text{Réduction} = \frac{1}{3} \times \left(2.206,32 \text{ EUR } \times \frac{\text{indemnite}}{\text{total des revenus}}\right).$$

[6] 38.090 EUR - 19.050 EUR

Modélisation

La réduction accordée pour des indemnités légales d'assurance maladie-invalidité est calculée par les prédicats suivants :

```
603  reduction_impot_indemnite(Revenus, Revenus_imposable,
604                            Part_indemnite, Reduction):-
605          Revenus_imposable #=< 1905000,
606          Revenus_imposable #= Part_indemnite,
607          Reduction #= 220632.
608  reduction_impot_indemnite(Revenus, Revenus_imposable,
609                            Part_indemnite, Reduction):-
610          Revenus_imposable #=< 1905000,
611          Revenus_imposable #\= Part_indemnite,
612          reduction_proportionnelle(220632, Part_indemnite, Revenus, Reduction).
613  reduction_impot_indemnite(Revenus, Revenus_imposable,
614                            Part_indemnite, Reduction):-
615          Revenus_imposable #> 1905000,
616          Revenus_imposable #< 3809000 ,
617          reduction_proportionnelle(220632, Part_indemnite, Revenus, Reduction1),
618          Terme1 #= Reduction1 // 3,
619          Facteur1 #= Reduction1 // 300,
620          Facteur2 #= (3809000 - Revenus_imposable) // 9520,
621          Terme2 #= Facteur1 * Facteur2,
622          Reduction #= Terme1 + Terme2.
623  reduction_impot_indemnite(Revenus, Revenus_imposable,
624                            Part_indemnite, Reduction):-
625          Revenus_imposable #>= 3809000 ,
626          reduction_proportionnelle(220632,Part_indemnite, Revenus, Reduction1),
627          Reduction #= Reduction1 //3.
```

Les prédicats et les contraintes représentées ci-dessus ne sont que la traduction des formules présentées à la section 4.6.10 modulo certaines simplifications arithmétiques indispensables afin d'éviter des problèmes de dépassement de bornes (overflow). Par exemple, nous simplifions le deuxième terme de l'équation (4.1) par un facteur 2 ce qui n'entraîne aucun changement dans le résultat final mais empêche d'atteindre des valeurs trop élevées lors de la multiplication.

4.6.11 Réduction d'impôt pour pensions, prépensions et autres revenus de remplacement

La réduction d'impôt sur ces revenus est de 1.718,76 EUR si le revenu net imposable du contribuable est composé uniquement d'une pension, d'une prépension ou d'un autre

revenu de remplacement est que ce revenu est inférieur à 19.050 EUR. En cas de revenu mixte inférieur à 19.050 EUR, la réduction est diminuée proportionnellement :

$$\text{Réduction} = 1.718,76 \text{ EUR } \times \frac{\text{montant des pensions, ...}}{\text{total des revenus nets}} = Y.$$

Si le revenu net imposable du contribuable est compris entre 19.050 EUR et 38.090 EUR, la réduction d'impôt est calculée comme suit :

$$\text{Réduction} = \frac{1}{3}Y + \left(\frac{2}{3}Y \times \frac{38.090 \text{ EUR } - \text{revenu net imposable}}{19.040 \text{ EUR}}\right).$$

Dans le cas d'un revenu supérieur à 38.090 EUR la réduction d'impôt est diminuée de la façon suivante :

$$\text{Réduction} = \frac{1}{3} \times \left(1.718,76 \text{ EUR } \times \frac{\text{montant pensions,...}}{\text{total des revenus}}\right).$$

Modélisation

La réduction accordée pour ce type de revenu sera modélisée de la même manière que la réduction de la section 4.6.10.

4.6.12 Réduction d'impôt pour rémunérations suite à la prestation de travail supplémentaire donnant droit à un sursalaire

La réduction d'impôt pour rémunérations suite à la prestation de travail supplémentaire donnant droit à un sursalaire est de 24.75% du montant brut des rémunérations qui a servi de base pour le calcul du sursalaire.

Le nombre maximal d'heures de travail supplémentaire est de 65. Au delà de cette limite, nous devons limiter la base de calcul proportionnellement à 65 heures :

$$\text{Réduction} = \frac{\text{Montant} \times 65}{\text{Nombre d'heures}} \times 24.75\%.$$

Modélisation

La réduction accordée pour des rémunérations de travail supplémentaire sera calculée à l'aide des deux prédicats suivants :

```
719  reduction_impot_sursalaire(Nombre_heure,Base_calcul, Reduction):-
720          Nombre_heure #=< 65,
721          Reduction #= ((Base_calcul // 100)  * 2475) // 100.
722  reduction_impot_sursalaire(Nombre_heure,Base_calcul, Reduction):-
723          Nombre_heure #> 65,
724          Reduction1 #= ((Base_calcul // 100)  * 2475) // 100,
725          Reduction #= Reduction1 * 65 // Nombre_heure.
```

4.6.13 Remarque sur les réductions liées aux revenus de remplacement

Pour toutes les réductions accordées pour des revenus de remplacement, nous devons, lors du calcul des réductions, limiter ces réductions à la part de l'impôt dû relatif à ces revenus. Autrement dit, la réduction d'impôt accordée pour un revenu de remplacement ne peut jamais être supérieure à l'impôt dû pour ce revenu.

4.7 Revenus imposables distinctement

Le contribuable a la possibilité d'opter pour une imposition distincte pour certains revenus. Dans ce cas, les revenus imposables disctinctement ne sont pas ajoutés aux revenus imposables globalement et ils sont taxés à un taux propre que nous allons détailler dans cette section. Remarquons également que si il est plus avantageux pour un contribuable de globaliser **tous** ses revenus, il en a le droit et le montant total sera alors imposé au tarif régressif (voir section 4.4).

Afin que le lecteur puisse rapidement voir de quel montant nous parlons, nous renseignerons les codes relatifs à ces montants dans le document préparatoire à la déclaration ainsi que le cadre dans lesquels ils se trouvent. En effet, le but de ce travail étant de calculer l'impôt dû, nous n'entrerons pas dans les détails concernant la manière de remplir le document préparatoire à la déclaration et nous partons du principe que le contribuable est capable de le faire correctement.

Néanmoins, pour certains revenus imposables distinctement, la connaissance du taux moyen de la dernière année antérieure pendant laquelle le contribuable a exercé une activité professionnelle est requise. Dans le cadre de ce mémoire, nous ne disposons évidemment pas de ces données[7]. Nous serons donc contraints d'utiliser pour nos calculs la globalisation de tous les revenus. Les éléments suivants sont donc fournis à titre purement indicatif.

4.7.1 Revenus imposables disctinctement au taux de 10%

Dans le cadre V concernant les pensions, les revenus concernant les contrats d'assurance-vie et d'assurance-groupe déclarés aux codes 1215/2215 ainsi que les revenus déclarés aux codes 1222/2222 sont imposables au taux de 10%.

4.7.2 Revenus imposables au taux de 10,38%

La prime de remise au travail régionale déclarée sous les codes 1291/2291 dans le cadre IV relatif aux traitements, salaires et revenus de remplacement doit être imposée à un taux de 10,38% dans le cas où cette prime ne dépasse pas le montant de 150,00 EUR.

[7]Contrairement à l'administration qui calcule réellement les impôts dû.

4.7.3 Revenus imposables au taux de 16,5%

Les montants déclarés sous les codes 1214, 2214, 1221 et 2221 dans le cadre V relatif aux pensions peuvent être imposés au taux de 16,5%.

4.7.4 Revenus imposables au taux de 33%

Les montants déclarés sous les codes 1213, 2213, 1220 et 2220 dans le cadre V relatif aux pensions peuvent être imposés au taux de 33%.

4.7.5 Revenus imposables au taux de la dernière année antérieure

Concernant le cadre IV (traitements, salaires et revenus de remplacement), les montants mentionnés aux codes 1252, 2252, 1261, 2261, 1263, 2263, 1268, 2268, 1272, 2272, 1282, 2282, 1253, 2253, 1254 et 2245 peuvent être imposés au taux de la dernière année.

Il en est de même pour les codes 1212, 2212, 1224 et 2224 du cadre V (pensions).

4.7.6 Revenus imposables au taux moyen de l'année

Dans le cadre IV, les pécules de vacances anticipés à déclarer sous les codes 1251 et 2251 ainsi que les montants mentionnés aux codes 1193 et 2193 du cadre VI (rentes alimentaires) peuvent être imposés au taux moyen de l'année.

4.7.7 Modélisation

Les revenus qui viennent d'être cités seront donc sommés et imposés globalement à l'aide du prédicat `impot` de la section 4.4.

4.8 Précomptes et crédits d'impôt

Les précomptes sont des sommes payées anticipativement par le contribuable qui seront soustraites à l'impôt des personnes physiques. Il existe plusieurs catégories de précomptes : ceux qui sont remboursables et ceux qui ne le sont pas.

Les précomptes non remboursables sont soustraits de l'impôt des personnes physiques. Les éventuelles sommes restantes (dans le cas d'un précompte trop important) ne sont pas remboursées au contribuable. Par contre, dans le cas des précomptes remboursables, si il y a un surplus, les sommes versées en trop par le contribuable lui seront remboursées.

4.8.1 Précompte immobilier

Le précompte immobilier est un précompte non remboursable qui est limité à 12,5% du revenu cadastral indexé[8].

Par exemple, si un contribuable possède une habitation ayant un revenu cadastral de 900 EUR, le revenu cadastral indexé vaut

$$900 \times 1.4276 = 1284,84.$$

Ce montant doit être arrondi à l'unité ce qui nous donne la somme de 1285 EUR. Pour obtenir le précompte immobilier, nous devons limiter cette somme à 12,5% :

$$\text{Précompte immobilier} = 1285 \times 12,5\% = 160.625 \text{ EUR} .$$

Modélisation

Le précompte immobilier pourra être calculé avec le prédicat `precompte_immobilier` à partir du revenu cadasral. On commence par calculer le revenu cadastral indexé (l'index pour l'exercice d'imposition 2007 est de 1,4276). Ensuite, on calcule le précompte avec un taux de 12,5%.

```
735   precompte_immobilier(Revenu_cadastral, Precompte):-
736           Revenu_indexe #= Revenu_cadastral // 1000 * 14276 // 10,
737           Precompte #= Revenu_indexe * 125 // 1000.
```

4.8.2 Précomptes remboursables

On définit comme réduction d'impôt, précomptes et crédits d'impôt **remboursables** :

- le précompte professionel sur traitements, salaires, pensions et revenus de remplacement (cadres IV et V) ;
- les versements anticipés (cadre XI) ;
- la réduction d'impôt "Prêt gagnant-gagnant" pour les habitants de la région flamande ;
- le crédit d'impôt "internet pour tous".

4.8.3 Crédits d'impôt ARKimedes

Les montants qui sont affectés par un contribuable à l'achat d'actions dans des fonds ARKimedes[9] donnent droit à ce contribuable à un crédit d'impôt égal à 8,75% des montants engagés avec un maximum de 2.500,00 EUR.

[8] Pour l'exercice d'imposition 2007, cet index vaut **1,4276**

[9] Capital à risque destiné aux entreprises de la région flamande

Notons que ce crédit d'impôt est "utilisé" afin de diminuer l'impôt dû par le contribuable après le précompte immobilier. Le solde éventuel n'est pas remboursé.

Modélisation

Les crédits d'impôt ARKimedes seront calculés à l'aide du prédicat suivant :

```
749  credit_impot_arkimedes(Montant_engage,Reduction):-
750          Reduction1 #= Montant_engage // 10000 * 875,
751          limiter_somme_superieur(Reduction1, 250000, Reduction).
```

De nouveau, nous devons effectuer la division en premier lieu (ligne 750) afin d'éviter un overflow. Cela a pour conséquence de ramener à 0 le crédit d'impôt pour tous les montants affectés de moins de 100,00 EUR. En effet, pour tout montant inférieur, le résultat de la division par 10.000 sera nul car il s'agit d'une division entière.

4.8.4 Crédit d'impôt "internet pour tous"

Afin d'inciter la population belge à investir dans du matériel informatique leur permettant d'accéder à internet, un crédit d'impôt est accordé à l'achat d'un paquet agréé "internet pour tous". Ce crédit est de 21% de la valeur d'achat (hors TVA) avec cependant un maximum de :

- 147,50 EUR pour un ordinateur fixe et
- 172,00 EUR pour un ordinateur portable.

Ce crédit d'impôt est remboursable.

Modélisation

Etant donné qu'il nous est impossible de savoir si il s'agit d'un ordinateur fixe ou portable, nous opterons systématiquement pour l'ordinateur portable (limitation à 172,00 EUR). Le crédit d'impôt sera calculé selon le même principe qu'à la section 4.8.3.

4.9 Taxes additionnelles

4.9.1 Versements anticipés

Chaque fois qu'un employeur paye ses salariés, il retient un précompte professionnel qui peut être vu comme un acompte sur l'impôt qu'ils devront payer plus tard. Les indépendants, quant à eux ne sont pas sous l'effet de cette retenue de salaire. Pour remédier à cette différence, les indépendants doivent, en général, payer une majoration sur leurs impôts. Afin d'éviter cette majoration, on leur permet, 4 fois par an, d'effectuer un versement anticipé sur l'impôt dû.

Dans le cadre de ce mémoire, nous nous occupons **uniquement** de la partie 1 de la déclaration fiscale c'est-à-dire que nous ne traitons pas le cas des indépendants. Nous ne devrons donc calculer aucune majoration d'impôt lors de l'établissement de l'impôt dû.

Malgré cela, il existe pour *tous* les contribuables un système de *bonification d'impôt* qui est lié aux éventuels versements anticipés que le contribuable aurait réalisés. En effet, il est possible qu'un contribuable, même après soustraction des précomptes, crédits d'impôts,..., doivent encore payer un supplément important. Si ils font des versements anticipés sur ce supplément, ces contribuables ont droit à une bonification d'impôt.

Les bonifications suivantes sont accordées sur les montants payés au plus tard le :

- le 10 avril 2006 $\Rightarrow$ 4,5% ;
- le 10 juillet 2006 $\Rightarrow$ 3,75% ;
- le 10 octobre 2006 $\Rightarrow$ 3% ;
- le 10 décembre 2006 $\Rightarrow$ 2,25% ;

Notons que la bonification n'est accordée que sur le supplément réellement dû. Si le contribuable paye par anticipation plus qu'il ne lui est nécessaire, l'excédent lui sera simplement restitué.

Afin de calculer les bonifications, nous devons augmenter de 6% l'impôt des personnes physiques dû. Nous soustrayons ensuite cette somme avec les précomptes autres que les versements anticipés et nous obtenons l'impôt maximum donnant droit à une bonification.

Par exemple, si l'impôt des personnes physiques dû par un employé est de 4.000 EUR et qu'il a effectué les versements anticipés suivants :

- le 2 avril 2006 $\Rightarrow$ 250 EUR ;
- le 8 juillet 2006 $\Rightarrow$ 250 EUR ;
- le 5 octobre 2006 $\Rightarrow$ 250 EUR ;
- le 7 décembre 2006 $\Rightarrow$ 250 EUR ;

Nous obtenons les calculs suivants :

Impôt maximum donnant droit à une bonification	$= 4000 \times 106\% =$	4.240 EUR
Précompte professionnel		−3.200 EUR
Précompte immobilier		−125 EUR
Montant pris en considération pour la bonification		915 EUR

Nous calculons ensuite la bonification sur le montant de 915 EUR :

1^{er} versement anticipé	$=$	250 EUR $\times 4,5\%$	$=$	$11,25$ EUR
$2^{\text{ème}}$ versement anticipé	$=$	250 EUR $\times 3,75\%$	$=$	$9,38$ EUR
$3^{\text{ème}}$ versement anticipé	$=$	250 EUR $\times 3\%$	$=$	$7,50$ EUR
$4^{\text{ème}}$ versement anticipé	$=$	125 EUR $\times 2,25\%$	$=$	$3,71$ EUR
				$31,84$ EUR

La bonification totale est donc obtenue en additionnant le montant excédentaire 325 EUR[10] avec la bonification calculée ce qui nous donne 356,84 EUR.

Dans le cadre de ce mémoire, il nous est impossible de savoir quand les versements anticipés ont été réalisés. En effet, le contribuable doit simplement renseigner le montant total des versements anticipés qu'il a effectué. Nous ferons donc l'hypothèse de répartir ce montant total dans les 4 périodes autorisées. Nous utiliserons donc le taux 3,375%[11] sur le total des versements anticipés.

Modélisation

La bonification accordée pour des versements anticipés peut être calculée à l'aide des prédicats suivants :

```
774  bonification(Impot_finalement_du,
775               Precompte_pro,
776               Precompte_immobilier,
777               Versement,
778               Bonification):-
779          Impot_max #= Impot_finalement_du * 106 // 100,
780          Impot_restant #= Impot_max - Precompte_pro - Precompte_immobilier,
781          Impot_restant #> 0,
782          Impot_restant #=< Versement,
783          Bonification #= Impot_restant // 100 * 3375 // 1000.
784  bonification(Impot_finalement_du,
785               Precompte_pro,
786               Precompte_immobilier,
787               Versement,
788               Bonification):-
789          Impot_max #= Impot_finalement_du * 106 // 100,
790          Impot_restant #= Impot_max - Precompte_pro - Precompte_immobilier,
791          Impot_restant #> 0,
792          Impot_restant #> Versement,
793          Bonification #= Versement // 100 * 3375 // 1000.
794  bonification(Impot_finalement_du,
795               Precompte_pro,
796               Precompte_immobilier,
797               _,
798               Bonification):-
799          Impot_max #= Impot_finalement_du * 106 // 100,
800          Impot_max #< Precompte_immobilier + Precompte_pro,
```

[10] 1000 EUR - 675 EUR

[11] La moyenne arithmétique des taux 4,5%, 3,75% ; 3% et 2,25%

```
801             Bonification #= 0.
```

Le prédicat de la ligne 774 à 783 permet de gérer le cas où il existe encore une somme dûe par le contribuable après soustraction des précomptes immobiliers et professionnels et où les versements effectués sont suffisants. Le deuxième prédicat gère un cas identique sauf que les versements sont insuffisants alors que le troisième prédicat s'occupe du cas où il n'y a pas d'impôt dû. Dans ce cas, la bonification est nulle.

4.9.2 Centimes additionnels d'agglomérations ou communaux

Tout contribuable doit également payer des *centimes additionnels communaux*. Ce sont les communes qui décident des taux applicables. Cette somme doit être calculée sur l'impôt des personnes physiques dû *après* déduction de :

- la réduction de l'épargne à long terme ;
- la réduction pour l'épargne logement ;
- la réduction pour les allocations de chômage, revenus de remplacement, pensions et prépensions ;
- la réduction pour indemnités légales d'assurance maladie-invalidité ;

mais *avant* l'imputation des versements anticipés.

Modélisation

Le taux à utiliser sera demandé à l'utilisateur avant de lancer le calcul de l'impôt. Le pourcentage retenu sera directement utilisé dans le programme C pour calculer les centimes additionnels d'agglomérations.

4.10 Quotient conjugal

Dans le cas d'une déclaration commune, si un seul des conjoints ou cohabitants légaux a bénéficié de revenus professionnels en 2006, une partie de ces revenus (appelée le **quotient conjugal**) peut être attribuée au conjoint qui ne perçoit pas de revenu professionnel.

Un quotient conjugal est également octroyé si les 2 conjoints bénificient de revenus professionnels mais que le **revenu professionnel net** d'un des deux ne dépasse pas 30% du total des revenus professionnels nets du ménage.

Les revenus professionnels nets sont calculés à partir des revenus professionnels déclarés diminués des frais professionnels, des éléments immunisés, des pertes professionnelles propres de l'année et des pertes professionnelles propres pas encore imputées des années antérieures. Les dépenses déductibles mentionnées au cadre VII et les pertes professionnelles de l'autre conjoint ne peuvent donc pas être déduites des revenus professionnels pour le calcul du quotient conjugal.

4.10.1 Calcul du quotient conjugal

Dans les ménages à revenu professionnel unique, 30% des revenus professionnels nets (avec un maximum de 8.560 EUR) peuvent être attribués au conjoint qui n'exerce pas d'activité professionnelle.

Dans les ménages à double revenus professionnels, le quotient conjugal est égal à la valeur absolue de la différence entre :

- 30% du total des revenus professionnels nets du ménage avec une limite de 8.560 EUR et
- les revenus professionnels nets du conjoint auquel on attribuera le quotien conjugal.

Par exemple dans un ménage, si les revenus professionnels nets du mari s'élèvent à 22.000 EUR et ceux de la femme à 7.200 EUR, on obtient :

$$\begin{aligned} \text{Quotient conjugal} &= |\underbrace{((22.000 + 7.200) \times 30\%)}_{=8.760 \text{ à limiter à } 8.560 \text{ EUR}} - 7.200| \\ &= |8.560 - 7.200| \\ &= 1.360 \text{ EUR.} \end{aligned}$$

Modélisation

Le quotient conjugal peut être calculé à l'aide des prédicats suivants :

```
814  quotient_conjugal(Revenu_fort, Revenu_faible, Quotient_conjugal):-
815          Total #= (Revenu_fort + Revenu_faible) * 3 // 10,
816          limiter_somme_superieur(Total, 856000, Total_correction),
817          Quotient_conjugal #= Total_correction - Revenu_faible.
818  quotient_conjugal(Revenu_fort, Revenu_faible, Quotient_conjugal):-
819          Total #= (Revenu_fort + Revenu_faible) * 3 // 10,
820          limiter_somme_superieur(Total, 856000, Total_correction),
821          Quotient_conjugal #= Revenu_faible - Total_correction .
```

Nous devons utiliser 2 prédicats car le solveur de contraintes de GNU Prolog ne gère pas les nombres négatifs et par conséquent les valeurs absolues.

4.11 Cotisation spéciale de sécurité sociale

Chaque contribuable est redevable d'une cotisation spéciale pour la sécurité sociale. Le montant total de la cotisation est déterminé à l'aide du tableau suivant :

Revenu net imposable du ménage	*Montant de la cotisation*
jusqu'à 18.592,02 EUR inclus	0 EUR
de 18.592,02 EUR à 21.070,96 EUR inclus	9% de la tranche de revenu à partir de 18.592,02 EUR
de 21.070,96 EUR à 60.161,85 EUR inclus	223,10 EUR + 1,3% de la tranche de revenu à partir de 21.070,96
à partir de 60.161,85 EUR	731,28 EUR

TAB. 4.4 – Calcul du montant de la cotisation spéciale pour la sécurité sociale

Remarquons qu'un accompte sur cette cotisation est déjà retenu par les employeurs sur le salaire des contribuables.

Modélisation

Le montant dû par un contribuable pour sa cotisation spéciale pour la sécurité sociale peut être calculé par le prédicat `calcul_cotisation_speciale`.

```
827  calcul_cotisation_speciale(Revenus_menage, 0):-
828          Revenus_menage #=< 1859202.
829  calcul_cotisation_speciale(Revenus_menage, Montant_cotisation):-
830          Revenus_menage #> 1859202,
831          Revenus_menage #=< 2107096,
832          Montant_cotisation #= 1859202 * 9 // 100.
833  calcul_cotisation_speciale(Revenus_menage, Montant_cotisation):-
834          Revenus_menage #> 2107096,
835          Revenus_menage #=< 6016185,
836          Tranche #= Revenus_menage - 2107096,
837          Montant_cotisation #= 22310 + (Tranche * 13 // 1000).
838  calcul_cotisation_speciale(Revenus_menage, 73128):-
839          Revenus_menage #> 6016185.
```

4.12 Calcul de l'impôt

Dans cette section, nous allons détailler les principales étapes que nous allons suivre afin de déterminer l'impôt d'un contribuable qui a rempli sa déclaration. Nous pouvons diviser le travail en 3 parties :

1. La détermination du revenu imposable
2. Le calcul de l'impôt
3. Le calcul final

4.12.1 Détermination du revenu imposable

Revenus des biens immobiliers

A partir des revenus cadastraux que le contribuable mentionne dans sa déclaration, nous pouvons calculer ses revenus de biens immobiliers ainsi que le précompte immobilier qu'il doit verser en fonction des ces revenus.

Pour déterminer le revenu immobilier, le revenu cadastral doit être indexé avec un coéfficient de 1.4276. Nous calculons également le précompte immobilier comme nous l'avons expliqué dans la section 4.8.1.

Si le contribuable a payé en 2006 des intérêts liés à un emprunt hypothécaire nous devons diminuer le revenu immobilier de ces intérêts. Dans le cas d'une déclaration commune, nous devons effectuer ce calcul pour chacun des conjoints.

Traitements et salaires

Le salaire imposable du déclarant est composé de son traitement, des pécules de vacances, des frais de déplacement, ... non exonérés d'impôt. Nous devons déduire de cette somme les frais professionnels forfaitaires et, éventuellement, le forfait spécial long déplacement. Nous obtenons alors le salaire net auquel nous devons ajouter les éventuelles allocations de chômage, indemnités légales de maladie, prépensions, ... ce qui nous donne le montant imposable au taux plein.

Dans le cas d'une déclaration commune, nous appliquons la règle du quotient conjugal. Nous devons ensuite enlever les éléments déductibles tels que les libéralités,...

4.12.2 Calcul de l'impôt

Nous devons ensuite calculer le montant exempté d'impôt auquel chaque contribuable a droit (voir section 4.2) ainsi que les différentes réductions d'impôt accordées pour des épargnes logements, épargnes à long terme,....

L'impôt est égal à l'impôt sur le revenu imposable calculé dans la section précédente auquel nous enlevons l'impôt sur le montant exempté d'impôt ainsi que les différentes réductions.

4.12.3 Calcul final

Une fois que nous avons calculé l'impôt, nous devons appliquer les règles de bonifications et soustraire le montant restant avec les précomptes immobiliers, professionnels. Nous devons ajouter les centimes additionnels d'agglomérations ou communaux ainsi que

le montant de la cotisation spéciale pour la sécurité sociale qui n'a pas encore été retenu.

Dans le cas d'un résultat positif, nous obtenons le montant que le contribuable doit payer. Si pas contre nous obtenons un solde négatif, il s'agit de la somme que le contribuable va récupérer.

Remarquons également que les soldes positifs de moins de 2,5 EUR ne doivent pas être payés et que les soldes négatifs de moins de 2,5 EUR ne seront pas remboursés.

4.12.4 Exemple

Afin d'illustrer les différentes étapes que nous venons de définir, prenons un exemple complet. Un salarié a 3 enfants à charge dont 1 qui, le premier janvier 2007, n'avait pas atteint l'âge de 3 ans. Il perçoit 27.264,27 EUR de rémunérations imposables et sa femme n'exerce pas d'activité professionnelle. Le précompte professionnel retenu pour l'homme s'élève à 4.074,29 EUR et 260,73 EUR ont déjà été retenus pour la cotisation spéciale de la sécurité sociale. Le salarié a également perçu 3.521,10 EUR de pécule de vacances sur lequel un précompte professionnel de 817,60 EUR a été retenu. Il a touché 619,73 EUR de frais de déplacement exonérés d'impôts. La distance entre son domicile et son lieu de travail est de 130 kilomètres. Il bénéficie donc d'un forfait long déplacement de 175 EUR. L'homme et la femme ont droit respectivement à une réduction d'impôt de 596,32 EUR et de 189,32 EUR pour l'épargne logement et de 186 EUR et de 191,75 EUR pour l'épargne à long terme. Ils ont effectué une libéralité de 60 EUR et sont les propriétaires d'une habitation dont le revenu cadastral est de 902 EUR. La moitié de ce revenu sera attribué à chacun des conjoints.

A présent, calculons l'impôt dû par le ménage.

Revenu imposable

Revenus immobilier

Pour l'homme		
Revenu cadastral :	401,00 EUR × 1,4276 =	644 EUR
Intêrets payés :		-2.157,16 EUR
Revenus immobiliers :		-1.513,16 EUR ⇒ 0 EUR
Précompte immobilier :	644,00 × 12,5% =	80,50 EUR
Pour la femme		
Revenu cadastral :	401,00 EUR × 1,4276 =	644 EUR
Intêrets payés :		-2.157,16 EUR
Revenus immobiliers :		-1.513,16 EUR ⇒ 0 EUR
Précompte immobilier :	644,00 × 12,5% =	80,50 EUR

Traitements et salaires		
Salaire du déclarant :		27.264,27 EUR
Pécule de vacances :		3.521,10 EUR
Frais déplacemement :		619,73 EUR
Frais déplacemement exonérés :		-619,73 EUR
Salaire imposable du déclarant :		30.785,37 EUR
Frais professionnels forfaitaires :		-2.487,75 EUR
Forfait longs déplacements :		-175,00 EUR
Imposable au taux plein :		28.122,62 EUR

Répartition quotient conjugal		
Quotient conjugal	28.122,62 EUR × 30% =	8.436,79 EUR

	homme	femme
Répartition des revenus initiaux	28.122,62 EUR	-
Quotient conjugal	-8.436,79 EUR	8.436,79 EUR
Répartition finale :	19.685,83 EUR	8.436,79 EUR

Eléments déductibles		
Pour l'homme :	60,00 EUR $\times \frac{19.685,83}{28.122,62}$ =	-42,00 EUR
Pour la femme :	60,00 EUR $\times \frac{8.426,79}{28.122,62}$ =	-18,00 EUR

Revenus imposables		
Pour l'homme :	19.685,83 EUR - 42,00 EUR =	19.643,83 EUR
Pour la femme :	8.436,79 EUR - 18,00 EUR =	8.418,79 EUR

Calcul de l'impôt

Montant exempté d'impôt	*homme*	*femme*
Montant exempté d'impôt pour chaque contribuable	5.940,00 EUR	5.940,00 EUR
Bonus 3 enfants à charge	7.280,00 EUR	-
Bonus enfants de moins de 3 ans	470,00 EUR	-
Total montant exempté d'impôt :	13.690,00 EUR	5.940,00 EUR
Impôt sur le revenu :	6.572,22 EUR	2.161,13 EUR
Impôt sur le montant exempté d'impôt :	-4.073,50 EUR	-1.485,00 EUR
Réduction épargne logement :	-596,32 EUR	-189,32 EUR
Réduction épargne long terme :	-186,00 EUR	- 191,75 EUR
Impôt :	1.716,40 EUR	295,06 EUR

Calcul final

	homme	*femme*
Impôt :	1.716,40 EUR	295,06 EUR
Précompte immobilier	-80,50 EUR	-80,50 EUR
Précompte professionnel	-4.891,91 EUR	0,00 EUR
	-3.256,01 EUR	214,56 EUR
Impôt à récupérer	-3.256,01 + 214,56 =	**- 3.041,45 EUR**
Taxe communale :		7%
Pour le homme	1.716,40 × 7% =	129,15 EUR
Pour la femme	295,06 × 7% =	20,65 EUR
Cotisation spéciale :		298,11 EUR
pour la sécurité sociale		
Déjà payé		-260,73 EUR
Reste à payer		37,38 EUR
Solde à récupérer :		- 3.041,45 EUR
		+ 129,15 EUR
		+ 20,65 EUR
		+ 37,38 EUR
		-2.854,27 EUR

4.12.5 Modélisation

Commençons par faire remarquer que notre interface Web ne permet qu'à un seul contribuable de rentrer des données. Si il s'agit d'une imposition commune, nous ferons donc l'hypothèse que les revenus de l'autre conjoint sont nuls.

La ligne conductrice de notre application est le programme C. Celui-ci reçoit les données personnelles du contribuable, les transforme pour que Prolog puisse les utiliser et appelle, tour à tour les différents prédicats nécessaires au calcul de l'impôt.

Par exemple, après avoir fait la somme de tous les revenus cadastraux encodés par l'utilisateur, le programme C appellera le prédicat `precompte_immobilier` pour calculer le précompte immobilier relatif à ces revenus.

Plus tard, le programme appellera le prédicat `montant_exempte` pour calculer, sur base des informations collectées, le montant exempté d'impôt auquel le contribuable aura droit. Et ainsi de suite avec les autres prédicats qui ont été présentés dans ce chapitre.

Une fois que le programme C sera en possession de tous les résultats, il effectuera les additions et soustractions nécessaires puis construira une page html à renvoyer au serveur afin que l'utilisateur puisse consulter le résumé du calcul de son impôt.

Chapitre 5

Conclusion

5.1 Intelligence artificielle & Loi

Le fait d'utiliser l'intelligence artificielle (la programmation logique) pour modéliser des lois et des règles juridiques n'est pas une nouveauté. En effet, de nombreux travaux ont déjà été réalisés dans cette optique appelée par les spécialistes "*IA and Law*".

On pourrait pourtant croire que les 2 disciplines, la loi et l'intelligence artificielle, n'ont rien en commun. En effet, la première est une discipline basée sur la culture humaine et existe depuis des millénaires alors que la seconde est une science basée sur la technologie qui a fait son apparition dans les années 1950. Néanmoins, les 2 disciplines nécessitent d'organiser de grandes quantités d'informations et ont pour buts de résoudre des problèmes dans des domaines d'activités très complexes c'est pourquoi, au fil des années, de nombreuses recherches et travaux ont étés effectués dans ce cadre.

Par exemple, le projet TAXMAN[10], dirigé par L.T. McCarty, avait pour but d'utiliser l'intelligence artificielle pour pouvoir évaluer, sur une base de faits pertinents, si une entreprise américaine pouvait être affectée à une des 3 catégories exemptées de taxe selon la législation en vigueur aux Etats Unis. Pour ce faire, McCarty a adopté une approche $bottom - up$ c'est-à-dire qu'il a, en premier lieu, modélisé les concepts les plus simples pour ensuite les assembler en concepts beaucoup plus complexes.

Un autre grand projet visant à appliquer la programmation logique à la législation est celui développé au département informatique de l'Imperial College de Londres. Le but était de formaliser l'acte de nationalité britannique dans un style déclaratif. Voici un exemple de cette modélisation tiré de [10] :

x acquires British citizenship on date y by sect. 1.1
if x was born in the UK.
and x was born on date y
and x has a parent who qualifies under 1.1 on date y

x has a parent who qualifies under 1.1 on date y
if z is a parent of x
and z is a British citizen on date y.

FIG. 5.1 – Exemple de modélisation du *British Nationality Act*

Contrairement au projet TAXMAN, l'approche utilisée ici est une approche $top-down$. En effet, la formalisation de la législation débute par le concept le plus haut niveau (celui de la citoyenneté britannique) et est réduit, via le mécanisme d'exécution de Prolog, a des règles modélisant des concepts de plus en plus primaires.

On remarque bien dans ce dernier exemple que les lois peuvent souvent être modélisées simplement en traduisant les textes statutaires en règles formelles interprétables par un ordinateur.

Le système expert que nous avons construit dans le cadre de ce mémoire utilise à la fois une approche $top-down$ et une approche $bottom-up$. En effet, certains concepts importants comme le calcul du montant exempté d'impôt ont été construits à partir de concepts plus primaires (celui d'isolé par exemple). Dans ce cas, c'est l'approche $tpo-down$ qui a été utilisée. Par contre, lors du calcul final de l'impôt dû, le système expert combine l'ensemble des lois et des règles modélisées afin de trouver la solution. On retrouve bien ici l'approche utilisée dans le cadre du projet TAXMAN à savoir de construire la solution à partir d'éléments plus petits.

5.2 Réalisation

Après avoir rappelé les principes de la programmation logique avec contraintes, nous avons expliqué les différentes étapes de la conception du système expert fiscal, de l'interface utilisateur jusqu'à la modélisation des lois fiscales en passant par le stockage des données et l'appel du programme. L'application ainsi conçue permet à un contribuable de calculer la somme d'argent qu'il doit payer ou récupérer pour la déclaration de ses revenus.

Comme nous l'avons déjà signalé, intelligence artificielle et lois sont des concepts qui, au fil des années, sont devenus très liés. Nous devons cependant nous poser une question importante :

"Le paradigme de programmation utilisé est-il bien
adapté au traitement de lois ***fiscales*** ?"

Il est vrai que le paradigme de programmation logique avec contraintes permet de modéliser de manière presque systématique la plupart des lois grâce au mécanisme très puissant des contraintes. Nous avons cependant rencontré plusieurs difficultés liées au caractère fini du solveur de contraintes utilisé. En effet, si certaines règles peuvent être modélisées parfaitement (le concept de contribuable isolé par exemple), d'autres le sont de manière relativement imprécises. Cela provient du fait que le domaine des contraintes utilisées dans le moteur d'inférence de GNU Prolog est fini. Il s'agit donc, pour des sommes d'argent, du domaine des entiers naturels. Or la plupart des montants utilisés dans le calcul de l'impôt sont des nombres réels avec 2 décimales ou plus. Nous avons donc dû multiplier tous les montants par 100 pour conserver une précision relative mais néanmoins pas totale. Le système expert construit ne convient donc pas à un utilisateur qui désire un résultat précis au centime près ! Ce qui est certainement le cas du fisc. L'application développée dans ce mémoire doit donc être utilisée uniquement dans le cadre de prévisions et de tentatives d'optimisation de l'impôt dû.

Outre le fait que le solveur de contraintes finies de GNU Prolog n'est pas spécialement adapté aux calculs fiscaux, d'autres difficultés ont été rencontrées dans le développement du système expert. Dans [6] et dans [3], les auteurs recommandent, lors de l'élaboration d'un système expert, l'implication de 2 types de personnes : les experts et les concepteurs. Les experts sont des personnes qui connaissent parfaitement bien le domaine étudié et qui aident les concepteurs à comprendre correctement les lois et les règles à suivre afin que ceux-ci les modélisent correctement. Dans le cadre de ce mémoire, nous n'avions pas d'expert "à notre disposition". Une des plus grandes difficultés a donc été de plonger dans les textes statutaires des lois fiscales pour en extraire le sens. N'étant pas habitué à ce type de texte, la compréhension s'est parfois faite avec difficulté.

Signalons enfin que l'ensemble des lois qui ont été modélisées n'est en aucun cas un ensemble exhaustif. En effet, bien que les règles d'imposition les plus courantes et les plus élémentaires aient été modélisées dans notre programme logique, plusieurs autres lois et cas d'exception ne s'y trouvent pas. De plus, le système expert se limite à la partie 1 de la déclaration d'impôt. La partie 2, consacrée aux dirigeants d'entreprise, aux commerçants et aux titulaires de professions libérales, n'a pas été modélisée. L'application ne peut donc pas être utilisée par ce type de contribuable.

Une amélioration possible serait donc de compléter la modélisation effectuée avec d'autres lois fiscales relatives à la partie 1 et par la suite, de rendre possible l'utilisation du système par les contribuables qui doivent également remplir la partie 2 de la déclaration.

5.3 Expérience personnelle

N'ayant moi-même jamais rempli de déclaration d'impôt, je n'avais aucune idée du travail qui m'attendait avant de commencer ce mémoire. Au fur et à mesure des lectures

des lois fiscales, je me suis fait une idée relativement précise de la façon de fonctionner du calcul de l'impôt.

Ce mémoire m'a appris à déployer un serveur Apache et à développer une application complète tournant sur ce serveur. Pour ce faire, j'ai dû apprendre les notions élémentaires des langages HTML et PHP ainsi que le fonctionnement de la *Common Gateway Interface*. J'ai étudié, de manière plus approfondie que ce qui a été vu dans mon cursus, le paradigme de la programmation logique avec contraintes et j'ai compris que ce paradigme pouvait être utilisé dans de nombreux cas pour résoudre des problèmes qui nécessiteraient plus de temps si nous utilisions un paradigme de programmation impératif. Pour la première fois, j'ai utilisé ce paradigme dans un projet d'une certaine ampleur via le moteur d'inférence GNU Prolog qui s'est révélé être un logiciel d'une grande qualité. En effet, il permet de créer des exécutables utilisant la puissance de la programmation logique sans pour autant devoir rentrer des requêtes "à la main" dans un shell ce qui permet d'utiliser Prolog dans beaucoup d'applications.

Annexe A

Procédure de déploiement sous Ubuntu

Dans cette section, nous présentons la procédure de déploiement de l'application sous Ubuntu 6.06 Dapper Drake. Cette procédure peut être décomposée en plusieurs étapes :

- installation d'utilitaires de base ;
- installation du serveur **Apache** ;
- installation du gestionnaire de base de données **MySQL** ;
- installation de **PHP** ;
- sécurisation du gestionnaire de base de données **MySQL** ;
- installation de **phpMyAdmin** ;
- installation de **GNU Prolog** ;
- chargement de la base de données MySQL ;
- préparation de l'interface Web et
- compilation du programme.

Installation d'utilitaires de base

Contrairement à beaucoup de distributions Linux, Ubuntu ne fournit pas dès l'installation le compilateur C et l'outil de compilation make. Or nous aurons besoin de ces programmes pour déployer notre application. Il est néanmoins très facile d'installer ces logiciels. Pour ce faire il suffit de taper en console la commande suivante :

```
sudo apt-get install build-essential
```

Installation du serveur Apache

La version utilisée sera **Apache2**. Afin d'installer le serveur Apache, nous devons entrer en console la ligne :

```
sudo apt-get install apache2 apache2-doc
```

Pour vérifier que l'installation s'est bien effectuée, nous pouvons entrer les URL `http://localhost` ou `http://127.0.0.1`. Si le serveur Apache est opérationnel, nous devons arriver sur une page d'accueil par défaut du serveur Apache.

Les pages web que nous voudrons rendre disponibles sur le serveur devront être ajoutées dans le répertoire `/var/www/`. Remarquons que ce répertoire est un répertoire système. Nous devrons donc utiliser les droits de super-utilisateur (via la commande `sudo`) pour écrire dans ce répertoire. Nous devons maintenant configurer le serveur pour qu'il puisse exécuter des scripts CGI[1].

Le fichier texte `/etc/apache2/sites-available/default` est le fichier de configuration du serveur Apache. Si nous désirons placer nos script CGI dans le répertoire `/var/www/cgi-bin`, nous devons remplacer les lignes :

```
ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/
     <Directory "/usr/lib/cgi-bin">
          AllowOverride None
          Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
          Order allow,deny
          Allow from all
     </Directory>
```

par les lignes

```
ScriptAlias /cgi-bin/ /var/www/cgi-bin/
     <Directory "/var/www/cgi-bin">
          AllowOverride None
          Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
          Order allow,deny
          Allow from all
     </Directory>
```

Il faut ensuite sauvegarder le fichier et redémarrer le serveur via la commande :

```
/etc/init.d/apache2 reload.
```

Afin de pouvoir utiliser les scripts CGI, nous devons également installer le programme **uncgi** qui permet de récupérer les données entrées dans un formulaire Web et de les utiliser dans un script CGI.

[1] Common Gateway Interface

Le programme **uncgi** peut être téléchargé à l'adresse `http://www.midwinter.com/~koreth/uncgi.html`. Avant de l'installer, il faut éditer le fichier `Makefile` et modifier les paramètres suivants :

- `CC` : indique le compilateur C que nous voulons utiliser (gcc,...) ;
- `SCRIPT_BIN` : indique le répertoire où nous allons placer nos scripts CGI : dans notre cas, il s'agit de `/var/www/cgi-bin` ;
- `DESTDIR` : indique le repertoire où nous voulons installer le programme **uncgi** : nous l'avons installé dans le répertoire cgi-bin de notre serveur à savoir `/var/www/cgi-bin`.

Après avoir enregistrer ce fichier, pour installer le programme, il suffit de se placer dans le répertoire où se trouve le fichier `Makefile` et de taper la ligne suivante en console :

`sudo make install`.

Afin de pouvoir appeler directement la fonction `uncgi()` à l'intérieur même d'un script CGI écrit en C, nous devons créer un fichier objet qui fera office de librairie. Pour ce faire, nous déplaçons le fichier `uncgi.c` dans le répertoire `/var/www/cgi-bin` et nous exécutons la commande `sudo gcc -DLIBRARY -c uncgi.c`. Nous obtenons alors le fichier `uncgi.o` qui sera utilisé lors de la compilation du programme C :

```
gcc monprogrammeC.c uncgi.o
```

Installation de MySQL

Pour installer le gestionnaire de base de données **MySQL**, il suffit d'entrer la ligne suivante dans une console :

`sudo apt-get install mysql-server`.

Installation de Php 5

Nous allons maintenant installer **Php 5** ainsi que quelques modules supplémentaires pour Apache 2 et MySQL. Voici la ligne de commande à entrer :

`sudo apt-get install php5 libapache2-mod-php5 php5-mysql`.

Ensuite, nous devons créer le fichier `/var/www/phpinfo.php` et y insérer le code suivant :

```
<?php
    phpinfo();
?>
```

Si l'installation s'est bien déroulée, en tapant l'adresse `http://localhost/phpinfo.php` dans notre navigateur internet, nous devons voir apparaitre une page Web contenant plusieurs informations sur le serveur (version de Php utilisée, ...).

Sécurisation du serveur MySQL

Jusqu'à présent, n'importe qui peut accéder à la base de données en entrant le nom d'utilisateur "root" et sans donner de mot de passe. Afin de sécuriser le serveur MySQL, nous devons rentrer la ligne de commande :

```
sudo mysql_secure_installation
```

Le système va alors nous demander d'entrer un nouveau mot de passe, nous demander si nous autorisons l'administration de la base de données à distance, ...

Il est bien sûr possible d'utiliser n'importe quel login et mot de passe. Cependant, si le login est différent de 'root' et que le mot de passe est différent de 'test', avant de pouvoir utiliser le système expert, il faudra modifier les instructions PHP qui permettent de se connecter à la base de données dans chacun des cadres de l'interface Web.

Installation de phpMyAdmin

phpMyAdmin est une interface conviviale réalisée en Php pour le gestionnaire de base de données MySQL afin de faciliter son administration sur des serveurs Web. Pour installer ce programme, il suffit d'entrer la ligne de commande suivante :

```
sudo apt-get install phpmyadmin.
```

Installation de GNU Prolog

Pour installer le moteur **GNU Prolog**, il suffit de taper la ligne suivant en console :

```
sudo apt-get install gprolog gprolog-doc.
```

Chargement de la base de données MySQL

Une fois que tous les logiciels nécessaires sont installés, il faut charger la base de données. Ceci peut être facilement effectué à l'aide de phpMyAdmin. Une fois sur la page d'accueil, il suffit de cliquer sur *Base de données* puis de créer une nouvelle base de données appelée 'sef'. Ensuite, il faut cliquer sur importer, spécifier le chemin du fichier `sef.sql` et exécuter la requête. La base de données est alors chargée et peut être utilisée par le serveur Apache.

Préparation de l'interface Web

Afin de pouvoir consulter les formulaires de déclaration d'impôt sur le serveur web, il faut copier les fichiers *.php dans le répertoire `/var/www/` du système. Il sera peut être également nécessaire d'accorder les privilèges de lecture à n'importe quel utilisateur via la commande `chmod a+r *.php`.

Compilation du programme

Nous pouvons maintenant compiler le programme qui va nous permettre de calculer l'impôt dû.

```
gplc sef.c sef.pl uncgi.o
```

La commande `gplc` permet de créer un exécutable à partir de programme C et Prolog. Nous obtenons alors le fichier `sef` qui doit être placé dans le répertoire `/var/www/cgi-bin`.

Une fois toutes ces opérations effectuées, nous pouvons utiliser le système expert en tapant l'url `http://localhost/cadre1.php` dans notre navigateur internet favori.

Bien entendu, le nom d'hôte `localhost` peut être remplacé par l'adresse ip de la machine sur laquelle tourne le serveur Apache. Le système expert peut donc être utilisé à distance.

Annexe B

Code Source

B.1 Code Prolog

```
/*
  * Le predicat impot_pas_du permet de savoir si le contribubale
  * n'est pas redevable de l'impot
*/
/* Cas ou le contribuable ne touche que des allocations de chomage */
impot_pas_du(Allocation_chomage, Pension, Prepension,Indemnite, Autre_revenu):-
        Autre_revenu #= 0,
        Pension + Prepension + Indemnite #= 0,!,
        Allocation_chomage #=< 1386918.
/* Cas ou le contribuable ne touche que des indemnites maladie */
impot_pas_du(Allocation_chomage, Pension, Prepension,Indemnite, Autre_revenu):-
        Autre_revenu #= 0,
        Pension + Prepension + Allocation_chomage #= 0,!,
        Indemnite #=< 1402007.
impot_pas_du(Allocation_chomage, Pension, Prepension, Indemnite, Autre_revenu):-
        Autre_revenu #= 0,
        Indemnite #= 0,!,
        Allocation_chomage + Pension + Prepension #=< 1261806.


/*
   *  Le predicat impot permet de calculer l'impot a partir d'un montant.
   *  Comme nous devons utiliser des entiers avec prolog, les montants reels
   *  sont multiplie par 100
*/
impot(Montant, Impot):-
        Montant #=< 729000,
        Impot #= (Montant * 25) // 100,
```

```
        fd_labeling(Montant).
impot(Montant, Impot):-
        Montant #> 729000,
        Montant #=< 1038000,
        Impot #= 182250 +(((Montant - 729000)* 30)// 100),
        fd_labeling(Montant).
impot(Montant, Impot):-
        Montant #> 1038000,
        Montant #=< 1730000,
        Impot #= 274950 +(((Montant - 1038000)* 40)// 100),
        fd_labeling(Montant).
impot(Montant, Impot):-
        Montant #> 1730000,
        Montant #=< 3170000,
        Impot #= 551750 +(((Montant - 1730000)* 45)// 100),
        fd_labeling(Montant).
impot(Montant, Impot):-
        Montant #> 3170000,
        Impot #= 1199750 +(((Montant - 3170000)* 50)// 100),
        fd_labeling(Montant).

/*
  * Le predicat frais_professionnel_forfaitaire permet de calculer
  * les frais professionnels forfaitaires auxquels le contribuable a droit.
*/
frais_professionnel_forfaitaire(RevenusBrut, Frais):-
        RevenusBrut #=< 479000,
        Frais #= RevenusBrut * 261 // 1000.
frais_professionnel_forfaitaire(RevenusBrut, Frais):-
        RevenusBrut #> 479000,
        RevenusBrut #=< 952000,
        Frais #= 125019 + ((RevenusBrut - 479000) // 10).
frais_professionnel_forfaitaire(RevenusBrut, Frais):-
        RevenusBrut #> 952000,
        RevenusBrut #=< 1585000,
        Frais #= 172319 + ((RevenusBrut - 952000) * 5 // 100).
frais_professionnel_forfaitaire(RevenusBrut, Frais):-
        RevenusBrut #> 1585000,
        Frais1 #= 203969 + ((RevenusBrut - 1585000) * 3 // 100),
        limiter_somme_superieur(Frais1, 323000, Frais).


/*
```

```
      * Le predicat taux permet d'obtenir le taux correspondant a une somme ainsi
      * que les bornes
    */
    taux(Montant, BorneInf, BorneSup, Taux):-
        Montant #=< 729000,
        BorneInf #= 0,
        BorneSup #= 729000,
        Taux #= 25.
    taux(Montant, BorneInf, BorneSup, Taux):-
        Montant #> 729000,
        Montant #=< 1038000,
        BorneInf #= 729000,
        BorneSup #= 1038000,
        Taux #= 30.
    taux(Montant, BorneInf, BorneSup, Taux):-
        Montant #> 1038000,
        Montant #=< 1730000,
        BorneInf #= 1038000,
        BorneSup #= 1730000,
        Taux #= 40.
    taux(Montant, BorneInf, BorneSup, Taux):-
        Montant #> 1730000,
        Montant #=< 3170000,
        BorneInf #= 1730000,
        BorneSup #= 3170000,
        Taux #= 45.
    taux(Montant, BorneInf, BorneSup, Taux):-
        Montant #> 3170000,
        BorneInf #= 3170000,
        BorneSup #= 0,
        Taux #= 50.


    /*
       *  Le predicat montant_exempte permet de calculer le montant exempte d'impot
       *  auquel le contribuable a droit.
    */
    montant_exempte(Etat_civil,
                    Handicape,
                    Nb_Enfant_Total,
                    Nb_Enfant_Handicape,
                    Nb_Enfant_Moitie,
                    Nb_Enfant_Moitie_Handicape,
```

```
                    Nb_Enfant_3ans,
                    Nb_Enfant_3ans_Handicape,
                    Nb_Pers_Charge,
                    Nb_Pers_Charge_Handicape,
                    Nb_Pers_Charge_65ans,
                    Nb_Pers_Charge_Handicape_65ans,
                    Bonus_handicape,
                    Bonus_isole,
                    Bonus_marie,
                    Bonus_cohabitant,
                    Bonus_enfant,
                    Bonus_enfant_moitie,
                    Bonus_enfant_3ans,
                    Bonus_pers_charge,
                    Bonus_pers_charge_65,
                    MontantExempte):-
    NE #= Nb_Enfant_Total + Nb_Enfant_Moitie,
    montant_exempte_bonus_handicape(Handicape, Bonus_handicape),
    montant_exempte_bonus_isole(Etat_civil,NE,Bonus_isole),
    montant_exempte_bonus_marie(Etat_civil,Bonus_marie),
    montant_exempte_bonus_cohabitant(Etat_civil,Bonus_cohabitant),
    /*
      *calcul du nombre d'enfant/personnes a prendre en compte
      * pour le calcul du montant
      * exempte d'impot (les enfants /personnes handicapes comptent pour
      * 2 enfants/personnes)
    */
    NET #= Nb_Enfant_Total + Nb_Enfant_Handicape,
    /* M1 contient le montant exempte calcule */
    montant_exempte_enfant(NET,Bonus_enfant),
    NEM #= Nb_Enfant_Moitie + Nb_Enfant_Moitie_Handicape,
    montant_exempte_enfant_moitie(NEM,Bonus_enfant_moitie),
    NE3 #= Nb_Enfant_3ans + Nb_Enfant_3ans_Handicape,
    montant_exempte_enfant_3ans(NE3,Bonus_enfant_3ans),
    NP #= Nb_Pers_Charge + Nb_Pers_Charge_Handicape,
    montant_exempte_pers_charge(NP,Bonus_pers_charge),
    NP65 #= Nb_Pers_Charge_65ans + Nb_Pers_Charge_Handicape_65ans,
    montant_exempte_pers_charge_65(NP65,Bonus_pers_charge_65),
    !, /* empeche la recherche d'autres solutions*/
    MontantExempte #= 594000 + Bonus_handicape + Bonus_isole + Bonus_marie
    + Bonus_cohabitant + Bonus_enfant + Bonus_enfant_moitie +
    Bonus_enfant_3ans + Bonus_pers_charge + Bonus_pers_charge_65.
```

```
/*
  * Le predicat montant_exempte_enfant donne le montant exempte supplementaire
  * auquel le contribuable a droit en fonction du nombre d'enfants a sa charge.
*/
montant_exempte_enfant(0,0).
montant_exempte_enfant(1,126000).
montant_exempte_enfant(2,326000).
montant_exempte_enfant(3,728000).
montant_exempte_enfant(4,1177000).
montant_exempte_enfant(Nb,Montant):-
        Nb #> 4,
        Nb_Enfant_Supp #= Nb-4,
        Montant #= 1177000 + Nb_Enfant_Supp*449000.

/*
  * Le predicat montant_exempte_enfant_moitie donne le montant exempte
  * supplementaire auquel le contribuable a droit en fonction du nombre
  * d'enfants pour moitie a sa charge.
*/
montant_exempte_enfant_moitie(0,0).
montant_exempte_enfant_moitie(1,63000).
montant_exempte_enfant_moitie(2,163000).
montant_exempte_enfant_moitie(3,364000).
montant_exempte_enfant_moitie(4,588500).
montant_exempte_enfant_moitie(Nb,Montant):-
        Nb #> 4,
        Nb_Enfant_Supp #= Nb-4,
        Montant #= ( 1177000 + Nb_Enfant_Supp*449000 ) // 2.

/*
  * Le predicat montant_exempte_enfant_3ans donne le montant exempte
  * supplementaire auquel le contribuable a droit en fonction du nombre
  * d'enfants a sa charge de moins  de 3 ans.
*/
montant_exempte_enfant_3ans(N,Montant):-
        Montant #= N * 47000.

/*
  * Le predicat montant_exempte_pers_charge donne le montant exempte
  * supplementaire auquel le contribuable a droit en fonction du nombre
  * de personne a sa charge.
*/
montant_exempte_pers_charge(N,Montant):-
```

```
200         Montant #= N * 126000.

/*
  * Le predicat montant_exempte_pers_charge_65 donne le montant exempte
  * supplementaire auquel le contribuable a droit en fonction du nombre
205   * de personne de plus de 65 ans a sa charge.
*/
montant_exempte_pers_charge_65(N,Montant):-
        Montant #= N * 252000.

210 /*
  * Le predicat montant_exempte_bonus_handicape permet de determiner si le
  * contribuable a droit a un montant exempte d'impot supplementaire
  * parce qu'il est handicape.
*/
215 montant_exempte_bonus_handicape(pashandicape,0).
montant_exempte_bonus_handicape(handicape,126000).


/*
220    * Le predicat montant_exempte_bonus_isole  permet de determiner si le
   * contribuable a droit a un bonus pour le montant exempte d'impot car
   * il est isole et a des enfants a charge.
*/
montant_exempte_bonus_isole([H|T],Nb_enfant,Montant):-
225         Nb_enfant #> 0,
            isole([H|T]),
            Montant #= 126000.
montant_exempte_bonus_isole(_,_,0).
montant_exempte_bonus_marie([H1,H2,H3|_],Montant):-
230         H1 = 1002, /* 1002=code pour marie */
            H2 = 1003, /* 1003=code pour marie depuis 2006*/
            H3 = 1004, /* 1004=code pour le conjoint a des ressources nettes
                          de moins de 2610 euro*/
            Montant #= 126000.
235 montant_exempte_bonus_marie(_,0).
montant_exempte_bonus_cohabitant([H1,H2,H3|_],Montant):-
            H1 = 1006, /* 1006=code pour cohabitant legal */
            H2 = 1007, /* 1007=code pour cohabitant legal depuis 2006*/
            H3 = 1008, /* 1008=code pour le conjoint a des ressources nettes
240                       de moins de 2610 euro*/
            Montant #= 126000.
montant_exempte_bonus_cohabitant(_,0).
```

```
/*
   * Le predicat isole permet de determiner si le contribuable est considere
   * comme isole.
*/
isole([H|_]):- H = 1001. /* 1001=code pour celibataire */
isole([H|_]):- H = 1010. /* 1010=code pour veuf/veuve */
isole([H1,H2|_]):-
        H1 = 1002, /* 1002=code pour marie */
        H2 = 1003. /* 1003=code pour marie depuis 2006*/
isole([H1,H2|_]):-
        H1 = 1006, /* 1006=code pour cohabitant legal */
        H2 = 1007. /* 1007=code pour cohabitant legal depuis 2006*/
isole([H1|T]):-
        H1 = 1018, /* 1018=code pour separe de faits */
        T = []. /* la separation de fait n'a pas eu lieu en 2006*/
isole([H1|_]):-
        H1 = 1014. /* 1014=code pour divorce */
isole([H1|_]):-
        H1 = 1016. /* 1016=code pour separe de corps */
isole([H1,H2|_]):-
        H1 = 1002, /* 1002=code pour marie */
        H2 = 1021. /* 1021=code pour marie a un fonctionnaire d'une
                      institution internationale*/
isole([H1,H2|_]):-
        H1 = 1006, /* 1006=code pour cohabitant legal */
        H2 = 1021. /* 1021=code pour cohabitant legal d'1 fonctionnaire
                      d'une institution internationale*/

/*
  * Le predicat montant_exempte_revenu_plus faible permet de determiner le
  * montant exempte d'impot auquel le contribuable qui a le revenu le
  * moins eleve a droit.
*/
montant_exempte_revenu_plus_faible(handicape,MontantExempte):-
        MontantExempte #= 594000 + 123000.
montant_exempte_revenu_plus_faible(pashandicape,MontantExempte):-
        MontantExempte #= 594000.

/*
  * Le predicat repartition_montant_exempte permet de repartir le montant
  * exempte d'impot dans le cas d'une declaration commune.
*/
```

```
repartition_montant_exempte(Revenus1,Revenus2,
                            MontantExempteInitial1,MontantExempteInitial2,
                            MontantExempteFinal1,MontantExempteFinal2):-
        Revenus1 #>= MontantExempteInitial1,
        Revenus2 #>= MontantExempteInitial2,
        MontantExempteFinal1 #= MontantExempteInitial1,
        MontantExempteFinal2 #= MontantExempteInitial2.
repartition_montant_exempte(Revenus1,Revenus2,
                            MontantExempteInitial1,MontantExempteInitial2,
                            MontantExempteFinal1,MontantExempteFinal2):-
        Revenus1 #>= MontantExempteInitial1,
        Revenus2 #< MontantExempteInitial2,
        MontantExempteFinal1 #= MontantExempteInitial1 +
        MontantExempteInitial2 - Revenus2,
        MontantExempteFinal2 #= Revenus2.
repartition_montant_exempte(Revenus1,Revenus2,
                            MontantExempteInitial1,MontantExempteInitial2,
                            MontantExempteFinal1,MontantExempteFinal2):-
        Revenus1 #< MontantExempteInitial1,
        Revenus2 #>= MontantExempteInitial2,
        MontantExempteFinal1 #=  Revenus1,
        MontantExempteFinal2 #= MontantExempteInitial2 +
        MontantExempteInitial1 - Revenus1.
repartition_montant_exempte(Revenus1,Revenus2,
                            MontantExempteInitial1,MontantExempteInitial2,
                            MontantExempteFinal1,MontantExempteFinal2):-
        Revenus1 #< MontantExempteInitial1,
        Revenus2 #< MontantExempteInitial2,
        MontantExempteFinal1 #= MontantExempteInitial1 +
        MontantExempteInitial2 - Revenus2,
        MontantExempteFinal2 #= Revenus2.


/*
  * Le predicat revenus_plus_faible permet de determiner quel conjoint
  * a le revenu le plus faible.
*/
revenus_plus_faible(Revenus1,Revenus2,NumRevenusfaible):-
        Revenus1 #< Revenus2,
        NumRevenusfaible #= 1.
revenus_plus_faible(Revenus1,Revenus2,NumRevenusfaible):-
        Revenus2 #=< Revenus1,
        NumRevenusfaible #= 2.
```

```
330
    /*************************************************************
      * PREDICATS RELATIFS AUX REDUCTIONS D'IMPOT
      *************************************************************/
    /*
335   * Le predicat reduction_epargne_logement permet de calculer la reduction
      * accordee au contribuable qui a paye pour une epargne-logement.
    */
    /* Cas ou les taux sont identiques. */
    reduction_epargne_logement(Revenu_net, Part_epargne, Reduction):-
340         taux(Revenu_net, _, _, Taux_revenu_net),
            Difference #= Revenu_net - Part_epargne,
            taux(Difference, _, _, Taux_difference),
            Taux_revenu_net = Taux_difference,
            Reduction #= (Part_epargne * Taux_revenu_net) // 100.
345 /* Cas ou les taux sont differents. */
    reduction_epargne_logement(Revenu_net, Part_epargne, Reduction):-
            taux(Revenu_net, Min, _, Taux_revenu_net),
            Difference #= Revenu_net - Part_epargne,
            taux(Difference, _, _, Taux_difference),
350         Taux_revenu_net #\= Taux_difference,
            Difference1 #= Revenu_net - Min,
            Part_epargne1 #= Part_epargne - Difference1,
            Reduction #= ((Difference1 * Taux_revenu_net) // 100) + Red1,
            reduction_epargne_logement(Min, Part_epargne1, Red1).
355
    /*
      * Le predicat limite_epargne_logement permet de calculer le montant limite
      * qui entre en considération pour le calcul de la reduction accordee
      * pour une epargne logement.
360 */
    limite_epargne_logement(Remuneration, Limite):-
            Remuneration #>=  160000,
            Tranche2 #= Remuneration - 160000,
            Limite1 #= 24000 + (Tranche2 *6 // 100),
365         limiter_somme_superieur(Limite1,192000,Limite).
    limite_epargne_logement(Remuneration, Limite):-
            Remuneration #<  160000,
            Limite #= Remuneration * 15 // 100.

370 /*
      * Le predicat reduction_epargne_long_terme permet de calculer la reduction
```

```
      * accordee au contribuable qui a paye pour une epargne_long_terme.
      * Il permet egalement de calculer la reduction accordee pour le
      * travail des ALE.
375   */
      reduction_epargne_long_terme(Revenu,
                                   Montant_exempte_CONJOINT,
                                   Part_epargne,
                                   Reduction):-
380           impot(Revenu,Impot_revenu),
              impot(Montant_exempte_CONJOINT,Impot_montant_exempte_CONJOINT),
              /*Le taux est calcule en pour 1000 pour avoir la precision necessaire*/
              Tmp #= Revenu // 10,
              Taux_imposition_moyen #= ((Impot_revenu - Impot_montant_exempte_CONJOINT)
385                                     * 100) // Tmp,
              limiter_somme_superieur(Taux_imposition_moyen,400,Taux1),
              limiter_somme_inferieur(Taux1,300,Taux),
              Reduction #= (Part_epargne * Taux) // 1000.
      /*Cas ou Impot_revenu - Impot_montant_exempte_CONJOINT est negatif*/
390   reduction_epargne_long_terme(Revenu,
                                   Montant_exempte_CONJOINT,
                                   Part_epargne,
                                   Reduction):-
              impot(Revenu,Impot_revenu),
395           impot(Montant_exempte_CONJOINT,Impot_montant_exempte_CONJOINT),
              Impot_revenu #< Impot_montant_exempte_CONJOINT,
              Reduction #= (Part_epargne * 3) // 10.

      /*
400     * Le predicat reduction_titre_service permet de calculer la reduction
        * accordee pour des montant payes avec des titres-services.
      */
      reduction_titre_service(Montant,Reduction):-
              Reduction #= (Montant * 3)//10.
405


      /*
        * Le predicat reduction_pret_gagnant permet de calculer la reduction accordee
        * pour des prets gagnants gagnants.
410   */
      reduction_pret_gagnant(Somme, Reduction):-
              Reduction1 #= Somme * 25 // 1000,
              limiter_somme_superieur(Reduction1,125000,Reduction).
      /*
```

```
415     * Le predicat reduction_pret_gagnant_perdu permet de calculer la reduction
        * accordee pour des prets gagnants gagnants.
      */
      reduction_pret_gagnant_perdu(Somme, Reduction):-
              Reduction1 #= Somme * 3 // 10,
420           limiter_somme_superieur(Reduction1,1500000,Reduction).

      /*
        * Les predicats limiter_somme_superieur et limiter_somme_inferieur permettent
        * de limiter une somme respectivement superieurement et inferieurement.
425   */
      limiter_somme_superieur(Montant, Limite, Nouveau_montant):-
              Montant #=< Limite,
              Nouveau_montant #= Montant.
      limiter_somme_superieur(Montant, Limite, Nouveau_montant):-
430           Montant #> Limite,
              Nouveau_montant #= Limite.
      limiter_somme_inferieur(Montant, Limite, Nouveau_montant):-
              Montant #>= Limite,
              Nouveau_montant #= Montant.
435   limiter_somme_inferieur(Montant, Limite, Nouveau_montant):-
              Montant #< Limite,
              Nouveau_montant #= Limite.

      /*
440     * Le predicat reduction_economie_energie permet de calculer la reduction
        * accordee pour les sommes depensees en vue d'economiser de l'energie
      */
      reduction_economie_energie(Montant,Reduction):-
              Reduction1 #= (Montant * 40)//100,
445           /*On limite la reduction a 128000 */
              limiter_somme_superieur(Reduction1,128000,Reduction).

      /*
        * Le predicat reduction_depense_renovation permet de calculer la reduction
450     * accordee pour les sommes depensees dans une renovation dans une zone
        * d'action positive des grandes villes.
      */
      reduction_depense_renovation(Montant,Reduction):-
              Reduction1 #= (Montant * 15)//100,
455                   /*On limite la reduction a 64000*/
              limiter_somme_superieur(Reduction1,64000,Reduction).
```

```
/*
  * Le predicat reduction_depense_vehicule_moins_105 permet de calculer la
  * reduction accordee pour les sommes depensees dans l'achat d'un vehicule
  * qui emet moins de 105 grammes de CO2 par kilometre.
*/
reduction_depense_vehicule_moins_105(Montant,Reduction):-
        /*
          * Ici, nous devons effectuer la division en premier lieu
          * sinon nous obtenons un overflow (voiture cher).
        */
        Reduction1 #= (Montant //100)* 15,
                /*On limite la reduction a 4190 EUR*/
        limiter_somme_superieur(Reduction1,419000,Reduction).
reduction_depense_vehicule_105_115(Montant, Reduction):-
        /*
          * Ici, nous devons effectuer la division en premier lieu
          * sinon nous obtenons un overflow (voiture cher).
        */
        Reduction1 #= (Montant //100)* 3,
                /*On limite la reduction a 790 EUR*/
        limiter_somme_superieur(Reduction1,79000,Reduction).


/*
  * Le predicat reduction_impot_chomage permet de calculer la reduction
  * accordee a un contribuable qui a percu des allocations de chomage.
*/
/*
  * Cas ou le revenu ne depasse pas 19050 EUR, est compose uniquement de
  * chomage et ou le contribuable est un isole.
*/
reduction_impot_chomage(Etat_civil,
                        _,
                        Revenus_imposable,
                        Part_chomage,
                        Reduction):-
        Revenus_imposable #< 1905000,
        Revenus_imposable = Part_chomage,
        isole(Etat_civil),
        Reduction #= 171876.
/*
  * Cas ou le revenu ne depasse pas 19050 EUR, est compose uniquement
```

```
      * de chomage et ou le contribuable n'est pas isole.
    */
    reduction_impot_chomage(Etat_civil,
                            _,
505                         Revenus_imposable,
                            Part_chomage,
                            Reduction):-
            Revenus_imposable #< 1905000,
            Revenus_imposable = Part_chomage,
510         \+isole(Etat_civil),
            Reduction #= 200688.
    /*
      * Cas ou le revenu ne depasse pas 19050 EUR et le contribuable est un isole.
    */
515 reduction_impot_chomage(Etat_civil,
                            Revenus,
                            Revenus_imposable,
                            Part_chomage,
                            Reduction):-
520         Revenus_imposable #< 1905000,
            Revenus_imposable #\= Part_chomage,
            isole(Etat_civil),
            reduction_proportionnelle(171876,Part_chomage,Revenus,Reduction).
    /*
525   * Cas ou le revenu ne depasse pas 19050 EUR et le contribuable n'est pas
      * isole.
    */
    reduction_impot_chomage(Etat_civil,
                            Revenus,
530                         Revenus_imposable,
                            Part_chomage,
                            Reduction):-
            Revenus_imposable #< 1905000,
            Revenus_imposable #\= Part_chomage,
535         \+isole(Etat_civil),
            reduction_proportionnelle(200688,Part_chomage,Revenus,Reduction).
    /*
      * Cas ou le revenu depasse 19050 EUR mais est inferieur a 23780 EUR et
      * le contribuable est un isole.
540 */
    reduction_impot_chomage(Etat_civil,
                            Revenus,
                            Revenus_imposable,
```

```
                            Part_chomage,
                            Reduction):-
        Revenus_imposable #>= 1905000,
        Revenus_imposable #< 2378000,
        isole(Etat_civil),
        reduction_proportionnelle(171876,Part_chomage,Revenus,Reduction1),
        Num #= 2378000 - Revenus_imposable,
        Den #= 2378000 - 1905000,
        reduction_proportionnelle(Reduction1, Num, Den, Reduction).
/*
  * Cas ou le revenu depasse 19050 EUR mais est inferieur a 23780 EUR et
  * le contribuable n'est pas isole.
*/
reduction_impot_chomage(Etat_civil,
                            Revenus,
                            Revenus_imposable,
                            Part_chomage,
                            Reduction):-
        Revenus_imposable #>= 1905000,
        Revenus_imposable #< 2378000,
        \+isole(Etat_civil),
        reduction_proportionnelle(200688,Part_chomage,Revenus,Reduction1),
        Num #= 2378000 - Revenus_imposable,
        Den #= 2378000 - 1905000,
        reduction_proportionnelle(Reduction1, Num, Den, Reduction).
/*
  * Cas ou le revenu depasse 23780 EUR. Aucune reduction n'est accordée.
*/
reduction_impot_chomage(_,
                            _,
                            Revenus_imposable,
                            _,
                            Reduction):-
        Revenus_imposable #>= 2378000,
        Reduction #= 0.

/*
  * Le predicat reduction_proportionnelle permet de calculer le montant
  * d'une reduction proportionnelle.
*/
reduction_proportionnelle(Base, Num, Den, Reduction):-
        /*
          * Etant donne la grandeur des nombres, nous devons diviser par 100
```

```
              * avant de multiplier Num et Base sinon on obtient un over flow!
             */
             Num1 #= Num // 100,
             Den1 #= Den // 100,
             Base1 #= Base // 100,
             Reduction #= Base1 * Num1 // Den1 * 100.


/*
  * Le predicat reduction_impot_indemnite permet de calculer la reduction
  * accordee sur des primes d'indemnites legales d'assurance maladie-sante.
*/
/*
  *Cas ou tous les revenus viennent de ces indemnites et ne depassent
  * pas 19050 EUR.
*/
reduction_impot_indemnite(_,
                          Revenus_imposable,
                          Part_indemnite,
                          Reduction):-
        Revenus_imposable #=< 1905000,
        Revenus_imposable #= Part_indemnite,
        Reduction #= 220632.
/*
  * Cas ou tous les revenus ne viennent pas de ces indemnites et ne
  * depassent pas 19050 EUR.
*/
reduction_impot_indemnite(Revenus,
                          Revenus_imposable,
                          Part_indemnite,
                          Reduction):-
        Revenus_imposable #=< 1905000,
        Revenus_imposable #\= Part_indemnite,
        reduction_proportionnelle(220632, Part_indemnite, Revenus, Reduction).
/*
  * Cas ou le revenu est compris entre 19050 EUR et 38090 EUR.
*/
reduction_impot_indemnite(Revenus,
                          Revenus_imposable,
                          Part_indemnite,
                          Reduction):-
        Revenus_imposable #> 1905000,
        Revenus_imposable #< 3809000 ,
```

```
630           reduction_proportionnelle(220632, Part_indemnite, Revenus, Reduction1),
              Terme1 #= Reduction1 // 3,
              /*
                * Par rapport a la formule, on divise facteur1 par 100 et on
                * multiplie facteur2 par 100 ce qui revient au meme pour le terme2.
635             * On est oblige de proceder de cette manière sinon on obtient
                * des nombres trop grands.
              */
              Facteur1 #= Reduction1 // 300,
              Facteur2 #= (3809000 - Revenus_imposable) // 9520,
640           Terme2 #= Facteur1 * Facteur2,
              Reduction #= Terme1 + Terme2.
      /*
        * Cas ou le revenu est superieur a 38090 EUR.
      */
645   reduction_impot_indemnite(Revenus,
                                Revenus_imposable,
                                Part_indemnite,
                                Reduction):-
              Revenus_imposable #>= 3809000 ,
650           reduction_proportionnelle(220632,Part_indemnite, Revenus, Reduction1),
              Reduction #= Reduction1 //3.

      /*
        * Le predicat reduction_impot_pension permet de calculer la reduction
655     * accordee sur des pensions, prepensions et autres revenus de remplacement.
      */
      /*
        *Cas ou tous les revenus viennent de pensions,... et ne depassent
        * pas 19050 EUR.
660   */
      reduction_impot_pensions(_,
                               Revenus_imposable,
                               Part_pensions,
                               Reduction):-
665           Revenus_imposable =< 1905000,
              Revenus_imposable = Part_pensions,
              Reduction #= 171876.
      /*
        * Cas ou tous les revenus ne viennent pas de pensions,...  et ne depassent
670     * pas 19050 EUR.
      */
      reduction_impot_pensions(Revenus,
```

```
                            Revenus_imposable,
                            Part_pensions,
                            Reduction):-
        Revenus_imposable #=< 1905000,
        Revenus_imposable #\= Part_pensions,
        reduction_proportionnelle(171876, Part_pensions, Revenus, Reduction).

/*
  * Cas ou le revenu est compris entre 19050 EUR et 38090 EUR.
*/
reduction_impot_pensions(Revenus,
                            Revenus_imposable,
                            Part_pensions,
                            Reduction):-
        Revenus_imposable #> 1905000,
        Revenus_imposable < 3809000 ,
        reduction_proportionnelle(171876, Part_pensions, Revenus, Reduction1),
        Terme1 #= Reduction1 // 3,
        /*
          * Par rapport a la formule, on divise facteur1 par 100 et on
          * multiplie facteur2 par 100 ce qui revient donc au meme pour terme2.
          * On est oblige de faire comme cela sinon on obtient des nombres
          * trop grands.
        */
        Facteur1 #= Reduction1 // 300,
        Facteur2 #= (3809000 - Revenus_imposable) // 9520,
        Terme2 #= Facteur1 * Facteur2,
        Reduction #= Terme1 + Terme2.
/*
  * Cas ou le revenu est superieur a 38090 EUR.
*/
reduction_impot_pensions(Revenus,
                            Revenus_imposable,
                            Part_pensions,
                            Reduction):-
        Revenus_imposable #>= 3809000 ,
        reduction_proportionnelle(171876,Part_pensions, Revenus, Reduction1),
        Reduction #= Reduction1 //3.

/*
  * Le predicat reduction_impot_sursalaire permet de calculer les reductions
  * accordees pour une remuneration d'heures supplementaire.
*/
```

```
/*
  * Cas ou il y a moins de 65 heures.
*/
reduction_impot_sursalaire(Nombre_heure,Base_calcul, Reduction):-
        Nombre_heure #=< 65,
        Reduction #= ((Base_calcul // 100)  * 2475) // 100.
/*
  * Cas ou il y a plus de 65 heures.
*/
reduction_impot_sursalaire(Nombre_heure,Base_calcul, Reduction):-
        Nombre_heure #> 65,
        Reduction1 #= ((Base_calcul // 100)  * 2475) // 100,
        Reduction #= Reduction1 * 65 // Nombre_heure.


/*
  * Le predicat precompte_immobilier permet de calculer le précompte immobilier
  * a partir du revenu cadastral.
*/
precompte_immobilier(Revenu_cadastral, Precompte):-
        /* index 2007 = 1.4276*/
        /* On doit utiliser cet ordre dans les operateurs pour eviter
           un overflow!*/
        Revenu_indexe #= Revenu_cadastral // 1000 * 14276 // 10,
                /*Reduction = 12,5% du revenus cadastral indexe*/
        Precompte #= Revenu_indexe * 125 // 1000.



/*
  * Le predicat credit_impot_arkimedes permet de calculer le credit d'impot
  * accorde a un contribuable qui a achete des actions dans des fonds
  * arkimedes.
*/
credit_impot_arkimedes(Montant_engage,Reduction):-
        /*Reduction = 8,75% des montants engagés*/
        Reduction1 #= Montant_engage // 10000 * 875,
        /*On limite la somme a 2500,00 EUR*/
        limiter_somme_superieur(Reduction1, 250000, Reduction).

/*
  * Le predicat credit_impot_internet permet de calculer le credit d'impot
  * accorde a un contribuable qui a achete un kit agree "internet pour tous".
*/
```

```
credit_impot_internet(Montant_achat,Reduction):-
        /*Reduction = 21% du montant de l'achat*/
        Reduction1 #= Montant_achat *21 // 100,
        /*On limite la somme a 172,00 EUR => ordi portable*/
        limiter_somme_superieur(Reduction1, 17200, Reduction).

/*
  ******************FIN REDUCTION************************
*/

/*
  * Le predicat bonification permet de calculer les bonifications auxquels a
  * droit un contribuable qui a effectue des versements anticipes.
*/
/*Cas ou les versements effectues sont suffisants.*/
bonification(Impot_finalement_du,
             Precompte_pro,
             Precompte_immobilier,
             Versement,
             Bonification):-
        Impot_max #= Impot_finalement_du * 106 // 100,
        Impot_restant #= Impot_max - Precompte_pro - Precompte_immobilier,
        Impot_restant #> 0,
        Impot_restant #=< Versement,
        Bonification #= Impot_restant // 100 * 3375 // 1000.
/*Cas ou les versements effectues ne sont pas suffisants.*/
bonification(Impot_finalement_du,
             Precompte_pro,
             Precompte_immobilier,
             Versement,
             Bonification):-
        Impot_max #= Impot_finalement_du * 106 // 100,
        Impot_restant #= Impot_max - Precompte_pro - Precompte_immobilier,
        Impot_restant #> 0,
        Impot_restant #> Versement,
        Bonification #= Versement // 100 * 3375 // 1000.
/*
  * Cas ou, apres deduction des precomptes, le contribuable ne doit plus
  * rien payer.
  * On doit alors simplement lui rendre ses versements anticipes.
*/
bonification(Impot_finalement_du,
             Precompte_pro,
```

```
                Precompte_immobilier,
                _,
                Bonification):-
        Impot_max #= Impot_finalement_du * 106 // 100,
        Impot_max #< Precompte_immobilier + Precompte_pro,
        Bonification #= 0.


/*
  * Le predicat quotient conjugal permet de calculer le quotient conjugal
  * pour un ménage.
*/
quotient_conjugal(Revenu_fort, Revenu_faible, Quotient_conjugal):-
        Total #= (Revenu_fort + Revenu_faible) * 3 // 10,
        limiter_somme_superieur(Total, 856000, Total_correction),
        Quotient_conjugal #= Total_correction - Revenu_faible.
quotient_conjugal(Revenu_fort, Revenu_faible, Quotient_conjugal):-
        Total #= (Revenu_fort + Revenu_faible) * 3 // 10,
        limiter_somme_superieur(Total, 856000, Total_correction),
        Quotient_conjugal #= Revenu_faible - Total_correction .

/*
  * Le predicat calcul_cotisation_speciale permet de calculer le montant
  * de la cotisation speciale pour la securite sociale.
*/
calcul_cotisation_speciale(Revenus_menage, 0):-
        Revenus_menage #=< 1859202.
calcul_cotisation_speciale(Revenus_menage, Montant_cotisation):-
        Revenus_menage #> 1859202,
        Revenus_menage #=< 2107096,
        Montant_cotisation #= 1859202 * 9 // 100.
calcul_cotisation_speciale(Revenus_menage, Montant_cotisation):-
        Revenus_menage #> 2107096,
        Revenus_menage #=< 6016185,
        Tranche #= Revenus_menage - 2107096,
        Montant_cotisation #= 22310 + (Tranche * 13 // 1000).
calcul_cotisation_speciale(Revenus_menage, 73128):-
        Revenus_menage #> 6016185.
```

B.2 Code du fichier calcul.php

```
<? session_start();

mysql_connect(localhost, root, test) or die("erreur connexion");
mysql_select_db("sef");

$num_nat=$_SESSION['num_nat'];

if(isset($_POST['save']))
        {
                $v1075=$_POST['v1075'];

                mysql_query("UPDATE CADRE12
                                        SET v1075 = '$v1075'
                                        WHERE NumeroNational = '$num_nat'");
        }

        $reponse1 = mysql_query("SELECT *
                                FROM CADRE1
                                WHERE NumeroNational=$num_nat");
        if ($donnees1 = mysql_fetch_array($reponse1))
                {
                $num_tel=$donnees1['NumTel'];
                $type_titulaire=$donnees1['TypeTitulaire'];
                $num_compte=$donnees1['NumCompte'];
                }

        $reponse2 = mysql_query("SELECT *
                                FROM CADRE2
                                WHERE NumeroNational='$num_nat'");

if ($donnees2 = mysql_fetch_array($reponse2))
        {
                $v1001=$donnees2['v1001'];
                $v1002=$donnees2['v1002'];
                $v1003=$donnees2['v1003'];
                $v1004=$donnees2['v1004'];
                $v1005=$donnees2['v1005'];
                $v1006=$donnees2['v1006'];
                $v1007=$donnees2['v1007'];
                $v1008=$donnees2['v1008'];
                $v1009=$donnees2['v1009'];
```

```
                $v1010=$donnees2['v1010'];
                $v1011=$donnees2['v1011'];
                $v1012=$donnees2['v1012'];
                $v1013=$donnees2['v1013'];
                $v1014=$donnees2['v1014'];
                $v1015=$donnees2['v1015'];
                $v1016=$donnees2['v1016'];
                $v1017=$donnees2['v1017'];
                $v1018=$donnees2['v1018'];
                $v1019=$donnees2['v1019'];
                $v1020=$donnees2['v1020'];
                $v1021=$donnees2['v1021'];
                $v1022=$donnees2['v1022'];
                $v1023=$donnees2['v1023'];
                $v1024=$donnees2['v1024'];
                $v1025=$donnees2['v1025'];
                $v1026=$donnees2['v1026'];
                $v1027=$donnees2['v1027'];
                $v1028=$donnees2['v1028'];
                $v1029=$donnees2['v1029'];
                $v1030=$donnees2['v1030'];
                $v1031=$donnees2['v1031'];
                $v1032=$donnees2['v1032'];
                $v1033=$donnees2['v1033'];
                $v1034=$donnees2['v1034'];
                $v1035=$donnees2['v1035'];
                $v1036=$donnees2['v1036'];
                $v1037=$donnees2['v1037'];
                $v1038=$donnees2['v1038'];
                $v1039=$donnees2['v1039'];
                $v1043=$donnees2['v1043'];
                $v1044=$donnees2['v1044'];
        }

        $reponse3 = mysql_query("SELECT *
                                FROM CADRE3
                                WHERE NumeroNational='$num_nat'");

if ($donnees3 = mysql_fetch_array($reponse3))
        {
                $v1100=$donnees3['v1100'];
                $v2100=$donnees3['v2100'];
                $v1101=$donnees3['v1101'];
```

```
85                  $v2101=$donnees3['v2101'];
                    $v1104=$donnees3['v1104'];
                    $v1105=$donnees3['v1105'];
                    $v2105=$donnees3['v2105'];
                    $v1106=$donnees3['v1106'];
90                  $v2106=$donnees3['v2106'];
                    $v1107=$donnees3['v1107'];
                    $v2107=$donnees3['v2107'];
                    $v1108=$donnees3['v1108'];
                    $v2108=$donnees3['v2108'];
95                  $v1109=$donnees3['v1109'];
                    $v2109=$donnees3['v2109'];
                    $v1110=$donnees3['v1110'];
                    $v2110=$donnees3['v2110'];
                    $v1111=$donnees3['v1111'];
100                 $v2111=$donnees3['v2111'];
                    $v1112=$donnees3['v1112'];
                    $v2112=$donnees3['v2112'];
                    $v1113=$donnees3['v1113'];
                    $v2113=$donnees3['v2113'];
105                 $v1115=$donnees3['v1115'];
                    $v2115=$donnees3['v2115'];
                    $v1116=$donnees3['v1116'];
                    $v2116=$donnees3['v2116'];
                    $v1114=$donnees3['v1114'];
110                 $v2114=$donnees3['v2114'];
                    $v1123=$donnees3['v1123'];
                    $v2123=$donnees3['v2123'];
                    $v1124=$donnees3['v1124'];
                    $v2124=$donnees3['v2124'];
115                 $v1125=$donnees3['v1125'];
                    $v2125=$donnees3['v2125'];
                    $v1130=$donnees3['v1130'];
                    $v2130=$donnees3['v2130'];
                    $v1131=$donnees3['v1131'];
120                 $v2131=$donnees3['v2131'];
                    $v1132=$donnees3['v1132'];
                    $v2132=$donnees3['v2132'];
                    $v1147=$donnees3['v1147'];
                    $v2147=$donnees3['v2147'];
125         }

    $reponse4a = mysql_query("SELECT *
```

```
                                FROM CADRE4A
                                WHERE NumeroNational='$num_nat'");
130
    if ($donnees4a = mysql_fetch_array($reponse4a))
            {
                    $v1250=$donnees4a['v1250'];
                    $v1249=$donnees4a['v1249'];
135                 $v1248=$donnees4a['v1248'];
                    $v1251=$donnees4a['v1251'];
                    $v1252=$donnees4a['v1252'];
                    $v1253=$donnees4a['v1253'];
                    $v1245=$donnees4a['v1245'];
140                 $v1291=$donnees4a['v1291'];
                    $v1254=$donnees4a['v1254'];
                    $v1255=$donnees4a['v1255'];
                    $v1256=$donnees4a['v1256'];
                    $v1257=$donnees4a['v1257'];
145                 $v1258=$donnees4a['v1258'];
                    $v1260=$donnees4a['v1260'];
                    $v1261=$donnees4a['v1261'];
                    $v1262=$donnees4a['v1262'];
                    $v1263=$donnees4a['v1263'];
150                 $v1264=$donnees4a['v1264'];
                    $v1265=$donnees4a['v1265'];
                    $v1266=$donnees4a['v1266'];
                    $v1268=$donnees4a['v1268'];
                    $v1292=$donnees4a['v1292'];
155                 $v1293=$donnees4a['v1293'];
                    $v1296=$donnees4a['v1296'];
                    $v1294=$donnees4a['v1294'];
                    $v1295=$donnees4a['v1295'];
                    $v1297=$donnees4a['v1297'];
160                 $v1269=$donnees4a['v1269'];
                    $v1270=$donnees4a['v1270'];
                    $v1271=$donnees4a['v1271'];
                    $v1272=$donnees4a['v1272'];
            }
165
    $reponse4b = mysql_query("SELECT *
                                FROM CADRE4B
                                WHERE NumeroNational='$num_nat'");

170 if ($donnees4b = mysql_fetch_array($reponse4b))
```

```
        {
                $v1281=$donnees4b['v1281'];
                $v1282=$donnees4b['v1282'];
                $v1285=$donnees4b['v1285'];
                $v1283=$donnees4b['v1283'];
                $v1246=$donnees4b['v1246'];
                $v1247=$donnees4b['v1247'];
                $v1286=$donnees4b['v1286'];
                $v1287=$donnees4b['v1287'];
                $v1290=$donnees4b['v1290'];
        }

$reponse5 = mysql_query("SELECT *
                        FROM CADRE5
                        WHERE NumeroNational='$num_nat'");

if ($donnees5 = mysql_fetch_array($reponse5))
        {
                $v1211=$donnees5['v1211'];
                $v1212=$donnees5['v1212'];
                $v1213=$donnees5['v1213'];
                $v1214=$donnees5['v1214'];
                $v1215=$donnees5['v1215'];
                $v1216=$donnees5['v1216'];
                $v1218=$donnees5['v1218'];
                $v1217=$donnees5['v1217'];
                $v1224=$donnees5['v1224'];
                $v1226=$donnees5['v1226'];
                $v1227=$donnees5['v1227'];
                $v1219=$donnees5['v1219'];
                $v1220=$donnees5['v1220'];
                $v1221=$donnees5['v1221'];
                $v1222=$donnees5['v1222'];
                $v1223=$donnees5['v1223'];
                $v1225=$donnees5['v1225'];
        }

$reponse6 = mysql_query("SELECT *
                        FROM CADRE6
                        WHERE NumeroNational='$num_nat'");

if ($donnees6 = mysql_fetch_array($reponse6))
        {
```

```
                $v1192=$donnees6['v1192'];
                $v1193=$donnees6['v1193'];
                $v1194=$donnees6['v1194'];
                $v1195=$donnees6['v1195'];
                $v1196=$donnees6['v1196'];
        }

$reponse7 = mysql_query("SELECT *
                        FROM CADRE7
                        WHERE NumeroNational='$num_nat'");

if ($donnees7 = mysql_fetch_array($reponse7))
        {
                $v1350=$donnees7['v1350'];
                $v1349=$donnees7['v1349'];
                $v1390=$donnees7['v1390'];
                $v1392=$donnees7['v1392'];
                $v1393=$donnees7['v1393'];
                $v1394=$donnees7['v1394'];
                $v1384=$donnees7['v1384'];
                $v1385=$donnees7['v1385'];
                $v1387=$donnees7['v1387'];
                $v1388=$donnees7['v1388'];
                $v1389=$donnees7['v1389'];
        }

$reponse8 = mysql_query("SELECT *
                        FROM CADRE8
                        WHERE NumeroNational='$num_nat'");

if ($donnees8 = mysql_fetch_array($reponse8))
        {
                $v1370=$donnees8['v1370'];
                $v1372=$donnees8['v1372'];
                $v1373=$donnees8['v1373'];
                $v1371=$donnees8['v1371'];
                $v1138=$donnees8['v1138'];
                $v1139=$donnees8['v1139'];
                $v1140=$donnees8['v1140'];
                $v1141=$donnees8['v1141'];
                $v1142=$donnees8['v1142'];
                $v1144=$donnees8['v1144'];
                $v1145=$donnees8['v1145'];
```

```
                $v1148=$donnees8['v1148'];
                $v1149=$donnees8['v1149'];
                $v1150=$donnees8['v1150'];
                $v1146=$donnees8['v1146'];
                $v1355=$donnees8['v1355'];
                $v1356=$donnees8['v1356'];
                $v1357=$donnees8['v1357'];
                $v1358=$donnees8['v1358'];
                $v1359=$donnees8['v1359'];
                $v1360=$donnees8['v1360'];
                $v1351=$donnees8['v1351'];
                $v1352=$donnees8['v1352'];
                $v1353=$donnees8['v1353'];
                $v1354=$donnees8['v1354'];
        }

$reponse9 = mysql_query("SELECT *
                        FROM CADRE9
                        WHERE NumeroNational='$num_nat'");

if ($donnees9 = mysql_fetch_array($reponse9))
        {
                $v1361=$donnees9['v1361'];
                $v1362=$donnees9['v1362'];
                $v1365=$donnees9['v1365'];
                $v1364=$donnees9['v1364'];
                $v1378=$donnees9['v1378'];
                $v1379=$donnees9['v1379'];
                $v1363=$donnees9['v1363'];
                $v1369=$donnees9['v1369'];
                $v1396=$donnees9['v1396'];
                $v1380=$donnees9['v1380'];
                $v1381=$donnees9['v1381'];
        }

$reponse10 = mysql_query("SELECT *
                         FROM CADRE10
                         WHERE NumeroNational='$num_nat'");

if ($donnees10 = mysql_fetch_array($reponse10))
        {
                $v1397=$donnees10['v1397'];
                $v1398=$donnees10['v1398'];
```

```
300          }

     $reponse11 = mysql_query("SELECT *
                              FROM CADRE11
                              WHERE NumeroNational='$num_nat'");
305
     if ($donnees11 = mysql_fetch_array($reponse11))
             {
                     $v1570=$donnees11['v1570'];
             }
310
     $reponse12 = mysql_query("SELECT *
                              FROM CADRE12
                              WHERE NumeroNational='$num_nat'");

315  if ($donnees12 = mysql_fetch_array($reponse12))
             {
                     $v1075=$donnees12['v1075'];
             }
     ?>
320
     <html>
     <body>
     <h3 align="center">Toutes vos donn&eacute;es ont bien &eacute;t&eacute;
     enregistr&eacute;es</h3>
325
     <form name="calcul" method="post" action="/cgi-bin/sef">
     <CENTER>
     Taux communal &agrave; utiliser :
     <input type="text" style="width:45px" name="taux_commune" value="7">%<BR> <BR>
330  </CENTER>

     <input type="hidden" name="num_nat" value="<?php echo $num_nat ?>">
     <input type="hidden" name="type_titulaire" value="<?php echo $type_titulaire ?>">
     <input type="hidden" name="num_tel" value="<?php echo $num_tel ?>">
335  <input type="hidden" name="num_compte" value="<?php echo $num_compte ?>">

     <input type="hidden" name="v1001" value="<?php echo $v1001 ?>">
     <input type="hidden" name="v1002" value="<?php echo $v1002 ?>">
     <input type="hidden" name="v1003" value="<?php echo $v1003 ?>">
340  <input type="hidden" name="v1004" value="<?php echo $v1004 ?>">
     <input type="hidden" name="v1005" value="<?php echo $v1005 ?>">
     <input type="hidden" name="v1006" value="<?php echo $v1006 ?>">
```

```
    <input type="hidden" name="v1007" value="<?php echo $v1007 ?>">
    <input type="hidden" name="v1008" value="<?php echo $v1008 ?>">
345 <input type="hidden" name="v1009" value="<?php echo $v1009 ?>">
    <input type="hidden" name="v1010" value="<?php echo $v1010 ?>">
    <input type="hidden" name="v1011" value="<?php echo $v1011 ?>">
    <input type="hidden" name="v1012" value="<?php echo $v1012 ?>">
    <input type="hidden" name="v1013" value="<?php echo $v1013 ?>">
350 <input type="hidden" name="v1014" value="<?php echo $v1014 ?>">
    <input type="hidden" name="v1015" value="<?php echo $v1015 ?>">
    <input type="hidden" name="v1016" value="<?php echo $v1016 ?>">
    <input type="hidden" name="v1017" value="<?php echo $v1017 ?>">
    <input type="hidden" name="v1018" value="<?php echo $v1018 ?>">
355 <input type="hidden" name="v1019" value="<?php echo $v1019 ?>">
    <input type="hidden" name="v1020" value="<?php echo $v1020 ?>">
    <input type="hidden" name="v1021" value="<?php echo $v1021 ?>">
    <input type="hidden" name="v1022" value="<?php echo $v1022 ?>">
    <input type="hidden" name="v1023" value="<?php echo $v1023 ?>">
360 <input type="hidden" name="v1024" value="<?php echo $v1024 ?>">
    <input type="hidden" name="v1025" value="<?php echo $v1025 ?>">
    <input type="hidden" name="v1026" value="<?php echo $v1026 ?>">
    <input type="hidden" name="v1027" value="<?php echo $v1027 ?>">
    <input type="hidden" name="v1028" value="<?php echo $v1028 ?>">
365 <input type="hidden" name="v1029" value="<?php echo $v1029 ?>">
    <input type="hidden" name="v1030" value="<?php echo $v1030 ?>">
    <input type="hidden" name="v1031" value="<?php echo $v1031 ?>">
    <input type="hidden" name="v1034" value="<?php echo $v1034 ?>">
    <input type="hidden" name="v1035" value="<?php echo $v1035 ?>">
370 <input type="hidden" name="v1036" value="<?php echo $v1036 ?>">
    <input type="hidden" name="v1037" value="<?php echo $v1037 ?>">
    <input type="hidden" name="v1038" value="<?php echo $v1038 ?>">
    <input type="hidden" name="v1039" value="<?php echo $v1039 ?>">
    <input type="hidden" name="v1043" value="<?php echo $v1043 ?>">
375 <input type="hidden" name="v1044" value="<?php echo $v1044 ?>">
    <input type="hidden" name="v1032" value="<?php echo $v1032 ?>">
    <input type="hidden" name="v1033" value="<?php echo $v1033 ?>">

    <input type="hidden" name="v1100" value="<?php echo $v1100 ?>">
380 <input type="hidden" name="v2100" value="<?php echo $v2100 ?>">
    <input type="hidden" name="v1101" value="<?php echo $v1101 ?>">
    <input type="hidden" name="v2101" value="<?php echo $v2101 ?>">
    <input type="hidden" name="v1104" value="<?php echo $v1104 ?>">
    <input type="hidden" name="v1105" value="<?php echo $v1105 ?>">
385 <input type="hidden" name="v2105" value="<?php echo $v2105 ?>">
```

```
    <input type="hidden" name="v1106" value="<?php echo $v1106 ?>">
    <input type="hidden" name="v2106" value="<?php echo $v2106 ?>">
    <input type="hidden" name="v1107" value="<?php echo $v1107 ?>">
    <input type="hidden" name="v2107" value="<?php echo $v2107 ?>">
390 <input type="hidden" name="v1108" value="<?php echo $v1108 ?>">
    <input type="hidden" name="v2108" value="<?php echo $v2108 ?>">
    <input type="hidden" name="v1109" value="<?php echo $v1109 ?>">
    <input type="hidden" name="v2109" value="<?php echo $v2109 ?>">
    <input type="hidden" name="v1110" value="<?php echo $v1110 ?>">
395 <input type="hidden" name="v2110" value="<?php echo $v2110 ?>">
    <input type="hidden" name="v1111" value="<?php echo $v1111 ?>">
    <input type="hidden" name="v2111" value="<?php echo $v2111 ?>">
    <input type="hidden" name="v1112" value="<?php echo $v1112 ?>">
    <input type="hidden" name="v2112" value="<?php echo $v2112 ?>">
400 <input type="hidden" name="v1113" value="<?php echo $v1113 ?>">
    <input type="hidden" name="v2113" value="<?php echo $v2113 ?>">
    <input type="hidden" name="v1115" value="<?php echo $v1115 ?>">
    <input type="hidden" name="v2115" value="<?php echo $v2115 ?>">
    <input type="hidden" name="v1116" value="<?php echo $v1116 ?>">
405 <input type="hidden" name="v2116" value="<?php echo $v2116 ?>">
    <input type="hidden" name="v1114" value="<?php echo $v1114 ?>">
    <input type="hidden" name="v2114" value="<?php echo $v2114 ?>">
    <input type="hidden" name="v1123" value="<?php echo $v1123 ?>">
    <input type="hidden" name="v2123" value="<?php echo $v2123 ?>">
410 <input type="hidden" name="v1124" value="<?php echo $v1124 ?>">
    <input type="hidden" name="v2124" value="<?php echo $v2124 ?>">
    <input type="hidden" name="v1125" value="<?php echo $v1125 ?>">
    <input type="hidden" name="v2125" value="<?php echo $v2125 ?>">
    <input type="hidden" name="v1130" value="<?php echo $v1130 ?>">
415 <input type="hidden" name="v2130" value="<?php echo $v2130 ?>">
    <input type="hidden" name="v1131" value="<?php echo $v1131 ?>">
    <input type="hidden" name="v2131" value="<?php echo $v2131 ?>">
    <input type="hidden" name="v1132" value="<?php echo $v1132 ?>">
    <input type="hidden" name="v2132" value="<?php echo $v2132 ?>">
420 <input type="hidden" name="v1147" value="<?php echo $v1147 ?>">
    <input type="hidden" name="v2147" value="<?php echo $v2147 ?>">

    <input type="hidden" name="v1250" value="<?php echo $v1250 ?>">
    <input type="hidden" name="v1249" value="<?php echo $v1249 ?>">
425 <input type="hidden" name="v1248" value="<?php echo $v1248 ?>">
    <input type="hidden" name="v1251" value="<?php echo $v1251 ?>">
    <input type="hidden" name="v1252" value="<?php echo $v1252 ?>">
    <input type="hidden" name="v1253" value="<?php echo $v1253 ?>">
```

```
    <input type="hidden" name="v1245" value="<?php echo $v1245 ?>">
    <input type="hidden" name="v1291" value="<?php echo $v1291 ?>">
    <input type="hidden" name="v1254" value="<?php echo $v1254 ?>">
    <input type="hidden" name="v1255" value="<?php echo $v1255 ?>">
    <input type="hidden" name="v1256" value="<?php echo $v1256 ?>">
    <input type="hidden" name="v1257" value="<?php echo $v1257 ?>">
    <input type="hidden" name="v1258" value="<?php echo $v1258 ?>">
    <input type="hidden" name="v1260" value="<?php echo $v1260 ?>">
    <input type="hidden" name="v1261" value="<?php echo $v1261 ?>">
    <input type="hidden" name="v1262" value="<?php echo $v1262 ?>">
    <input type="hidden" name="v1263" value="<?php echo $v1263 ?>">
    <input type="hidden" name="v1264" value="<?php echo $v1264 ?>">
    <input type="hidden" name="v1265" value="<?php echo $v1265 ?>">
    <input type="hidden" name="v1266" value="<?php echo $v1266 ?>">
    <input type="hidden" name="v1268" value="<?php echo $v1268 ?>">
    <input type="hidden" name="v1292" value="<?php echo $v1292 ?>">
    <input type="hidden" name="v1293" value="<?php echo $v1293 ?>">
    <input type="hidden" name="v1296" value="<?php echo $v1296 ?>">
    <input type="hidden" name="v1294" value="<?php echo $v1294 ?>">
    <input type="hidden" name="v1295" value="<?php echo $v1295 ?>">
    <input type="hidden" name="v1297" value="<?php echo $v1297 ?>">
    <input type="hidden" name="v1269" value="<?php echo $v1269 ?>">
    <input type="hidden" name="v1270" value="<?php echo $v1270 ?>">
    <input type="hidden" name="v1271" value="<?php echo $v1271 ?>">
    <input type="hidden" name="v1272" value="<?php echo $v1272 ?>">

    <input type="hidden" name="v1281" value="<?php echo $v1281 ?>">
    <input type="hidden" name="v1282" value="<?php echo $v1282 ?>">
    <input type="hidden" name="v1285" value="<?php echo $v1285 ?>">
    <input type="hidden" name="v1283" value="<?php echo $v1283 ?>">
    <input type="hidden" name="v1246" value="<?php echo $v1246 ?>">
    <input type="hidden" name="v1247" value="<?php echo $v1247 ?>">
    <input type="hidden" name="v1286" value="<?php echo $v1286 ?>">
    <input type="hidden" name="v1287" value="<?php echo $v1287 ?>">
    <input type="hidden" name="v1290" value="<?php echo $v1290 ?>">

    <input type="hidden" name="v1211" value="<?php echo $v1211 ?>">
    <input type="hidden" name="v1212" value="<?php echo $v1212 ?>">
    <input type="hidden" name="v1213" value="<?php echo $v1213 ?>">
    <input type="hidden" name="v1214" value="<?php echo $v1214 ?>">
    <input type="hidden" name="v1215" value="<?php echo $v1215 ?>">
    <input type="hidden" name="v1216" value="<?php echo $v1216 ?>">
    <input type="hidden" name="v1218" value="<?php echo $v1218 ?>">
```

```
    <input type="hidden" name="v1217" value="<?php echo $v1217 ?>">
    <input type="hidden" name="v1224" value="<?php echo $v1224 ?>">
    <input type="hidden" name="v1226" value="<?php echo $v1226 ?>">
475 <input type="hidden" name="v1227" value="<?php echo $v1227 ?>">
    <input type="hidden" name="v1219" value="<?php echo $v1219 ?>">
    <input type="hidden" name="v1220" value="<?php echo $v1220 ?>">
    <input type="hidden" name="v1221" value="<?php echo $v1221 ?>">
    <input type="hidden" name="v1222" value="<?php echo $v1222 ?>">
480 <input type="hidden" name="v1223" value="<?php echo $v1223 ?>">
    <input type="hidden" name="v1225" value="<?php echo $v1225 ?>">

    <input type="hidden" name="v1192" value="<?php echo $v1192 ?>">
    <input type="hidden" name="v1193" value="<?php echo $v1193 ?>">
485 <input type="hidden" name="v1194" value="<?php echo $v1194 ?>">
    <input type="hidden" name="v1195" value="<?php echo $v1195 ?>">
    <input type="hidden" name="v1196" value="<?php echo $v1196 ?>">

    <input type="hidden" name="v1350" value="<?php echo $v1350 ?>">
490 <input type="hidden" name="v1349" value="<?php echo $v1349 ?>">
    <input type="hidden" name="v1390" value="<?php echo $v1390 ?>">
    <input type="hidden" name="v1392" value="<?php echo $v1392 ?>">
    <input type="hidden" name="v1393" value="<?php echo $v1393 ?>">
    <input type="hidden" name="v1394" value="<?php echo $v1394 ?>">
495 <input type="hidden" name="v1384" value="<?php echo $v1384 ?>">
    <input type="hidden" name="v1385" value="<?php echo $v1385 ?>">
    <input type="hidden" name="v1387" value="<?php echo $v1387 ?>">
    <input type="hidden" name="v1388" value="<?php echo $v1388 ?>">
    <input type="hidden" name="v1389" value="<?php echo $v1389 ?>">
500
    <input type="hidden" name="v1370" value="<?php echo $v1370 ?>">
    <input type="hidden" name="v1372" value="<?php echo $v1372 ?>">
    <input type="hidden" name="v1373" value="<?php echo $v1373 ?>">
    <input type="hidden" name="v1371" value="<?php echo $v1371 ?>">
505 <input type="hidden" name="v1138" value="<?php echo $v1138 ?>">
    <input type="hidden" name="v1139" value="<?php echo $v1139 ?>">
    <input type="hidden" name="v1140" value="<?php echo $v1140 ?>">
    <input type="hidden" name="v1141" value="<?php echo $v1141 ?>">
    <input type="hidden" name="v1142" value="<?php echo $v1142 ?>">
510 <input type="hidden" name="v1144" value="<?php echo $v1144 ?>">
    <input type="hidden" name="v1145" value="<?php echo $v1145 ?>">
    <input type="hidden" name="v1148" value="<?php echo $v1148 ?>">
    <input type="hidden" name="v1149" value="<?php echo $v1149 ?>">
    <input type="hidden" name="v1150" value="<?php echo $v1150 ?>">
```

```
<input type="hidden" name="v1146" value="<?php echo $v1146 ?>">
<input type="hidden" name="v1355" value="<?php echo $v1355 ?>">
<input type="hidden" name="v1356" value="<?php echo $v1356 ?>">
<input type="hidden" name="v1357" value="<?php echo $v1357 ?>">
<input type="hidden" name="v1358" value="<?php echo $v1358 ?>">
<input type="hidden" name="v1359" value="<?php echo $v1359 ?>">
<input type="hidden" name="v1360" value="<?php echo $v1360 ?>">
<input type="hidden" name="v1351" value="<?php echo $v1351 ?>">
<input type="hidden" name="v1352" value="<?php echo $v1352 ?>">
<input type="hidden" name="v1353" value="<?php echo $v1353 ?>">
<input type="hidden" name="v1354" value="<?php echo $v1354 ?>">

<input type="hidden" name="v1361" value="<?php echo $v1361 ?>">
<input type="hidden" name="v1362" value="<?php echo $v1362 ?>">
<input type="hidden" name="v1365" value="<?php echo $v1365 ?>">
<input type="hidden" name="v1364" value="<?php echo $v1364 ?>">
<input type="hidden" name="v1378" value="<?php echo $v1378 ?>">
<input type="hidden" name="v1379" value="<?php echo $v1379 ?>">
<input type="hidden" name="v1363" value="<?php echo $v1363 ?>">
<input type="hidden" name="v1369" value="<?php echo $v1369 ?>">
<input type="hidden" name="v1396" value="<?php echo $v1396 ?>">
<input type="hidden" name="v1380" value="<?php echo $v1380 ?>">
<input type="hidden" name="v1381" value="<?php echo $v1381 ?>">

<input type="hidden" name="v1397" value="<?php echo $v1397 ?>">
<input type="hidden" name="v1398" value="<?php echo $v1398 ?>">

<input type="hidden" name="v1570" value="<?php echo $v1570 ?>">

<input type="hidden" name="v1075" value="<?php echo $v1075 ?>">

<center><input type="submit" value="Lancer le calcul de l'imp&ocirc;t">
</center>
</body>
</html>
```

B.3 Code C

```
#include<stdio.h>
#include<stdlib.h>
#include"/usr/include/gprolog/gprolog.h"

int Main_Wrapper(){

  /**********************DECLARATION DES VARIABLES*********************/
  int predicat; //cle interne du predicat prolog
  int pas_imposition_commune; // vaut 1 si il ne s'agit pas d'une imposition
                              // commune,
                              // 0 sinon.
  float taux_commune; // taux centimes additionnels d'agglomeration.

  // variables cadre 1
  char *numero_national; // numero national du contribuable
  int type_titulaire; // type de titulaire de la declaration
  char *compte; // compte contribuable
  char *num_tel; // numero de telephone du contribuable


  // Variable Prolog
  PlTerm liste_etat_civil; // liste d'etat civil;
  PlTerm tableau_etat_civil[4]; // tableau d'etat civil
  PlTerm handicape; // statut d'handicape
  // Variable Prolog charge de famille
  PlTerm Nb_Enfant_Total, Nb_Enfant_Handicape;
  PlTerm Nb_Enfant_Moitie, Nb_Enfant_Moitie_Handicape;
  PlTerm Nb_Enfant_3ans, Nb_Enfant_3ans_Handicape;
  PlTerm Nb_Pers_Charge, Nb_Pers_Charge_Handicape;
  PlTerm Nb_Pers_Charge_65ans,  Nb_Pers_Charge_Handicape_65ans;

  // Variable revenus de biens immobiliers
  double revenus_cadastral_homme; // RC homme
  double revenus_cadastral_sans_precompte_h;// RC homme non soumis au precompte
  PlTerm RC_homme; // RC homme

  // Variables cadre IV
  double remunerations_ordinaires;
  double frais_deplacement;
  double frais_deplacement_exonere;
  double forfait_long_deplacement;
```

```
        double allocation_chomage;
        double indemnite_maladie;
        double revenus_remplacement;
45      double prepension;
        PlTerm Pl_nb_heure_sup;
        double heure_sup;
        PlTerm Pl_heure_sup;
        double precompte_prof;
50      double retenue_sec_sociale;

        // Variables cadre V
        double pension;
        double precompte_prof_pension;
55
        // Le total des precomptes professionnels payes par le contribuable
        double precompte_prof_total;

        double pension_prepension_autre;
60
        // Salaire a prendre en compte pour le calcul des frais professionnels.
        double salaire_imposable_pour_frais_prof;
        // Salaire total (y compris les allocations de chomage,  ...)
        double salaire_imposable;
65      // Salaire ou on a deja deduit les frais professionnels.
        double imposable_taux_plein;

        // Dans le cas d'une declaration commune -> repartition quotient conjugal
        double revenu_homme_hors_deduction;
70      double revenu_femme_hors_deduction;

        // revenu imposable une fois que les deduction ont ete faites
        double revenu;
        // idem mais pour une imposition commune
75      double revenu_homme;
        double revenu_femme;

        // Variables cadre VI

80      double rente1;
        double rente2;
        double rente3;

        // Variables cadre VII
```

```
85      double elements_deductibles;

        // Variables cadre VIII
        double epargne_logement;
        double epargne_long_terme;
90
        // Variables cadre IX
        double versement_ALE;
        double versement_titre_service;
        double pret_gagnant_gagnant;
95      double pret_gagnant_gagnant_perdu;
        double economie_energie;
        double renovation_zone_active;
        double voiture_moins_105;
        double voiture_plus_105;
100
        // Variables cadre X
        double arkimedes;
        double internet;

105     // Variable cadre XI
        double versement_anticipe;

        // Demarrage du moteur d'inference GNU Prolog
        Start_Prolog(0,NULL);
110
        /*************RECUPERATION ET TRAITEMENT DES VARIABLES****************/

        // Recuperation des donnees du formulaire html avec la fonction uncgi()
        uncgi();
115     // Maintenant, les variables sont placees dans des variables d'environnement

        // On extrait les valeurs des variables d'environnement
        // et on traite les variables pour que Prolog puisse les utiliser

120     taux_commune = atof(getenv("WWW_taux_commune"));

        // CADRE 1
        numero_national = getenv("WWW_num_nat");
        type_titulaire = atoi(getenv("WWW_type_titulaire"));
125     compte = getenv("WWW_num_compte");
        num_tel = getenv("WWW_num_tel");
```

```
        // CADRE 2
        int nb_etat_civil = 0; // compte le nombre de case cochee pour etat civil
130     if(strcmp(getenv("WWW_v1001"),"checked") == 0){
          tableau_etat_civil[nb_etat_civil] = Mk_Integer(1001);
          nb_etat_civil++;
        }
        if(strcmp(getenv("WWW_v1002"),"checked") == 0){
135       tableau_etat_civil[nb_etat_civil] = Mk_Integer(1002);
          nb_etat_civil++;
        }
        if(strcmp(getenv("WWW_v1003"),"checked") == 0){
          tableau_etat_civil[nb_etat_civil] = Mk_Integer(1003);
140       nb_etat_civil++;
        }
        if(strcmp(getenv("WWW_v1004"),"checked") == 0){
          tableau_etat_civil[nb_etat_civil] = Mk_Integer(1004);
          nb_etat_civil++;
145     }
        if(strcmp(getenv("WWW_v1005"),"checked") == 0){
          tableau_etat_civil[nb_etat_civil] = Mk_Integer(1005);
          nb_etat_civil++;
        }
150     if(strcmp(getenv("WWW_v1006"),"checked") == 0){
          tableau_etat_civil[nb_etat_civil] = Mk_Integer(1006);
          nb_etat_civil++;
        }
        if(strcmp(getenv("WWW_v1007"),"checked") == 0){
155       tableau_etat_civil[nb_etat_civil] = Mk_Integer(1007);
          nb_etat_civil++;
        }
        if(strcmp(getenv("WWW_v1008"),"checked") == 0){
          tableau_etat_civil[nb_etat_civil] = Mk_Integer(1008);
160       nb_etat_civil++;
        }
        if(strcmp(getenv("WWW_v1009"),"checked") == 0){
          tableau_etat_civil[nb_etat_civil] = Mk_Integer(1009);
          nb_etat_civil++;
165     }
        if(strcmp(getenv("WWW_v1010"),"checked") == 0){
          tableau_etat_civil[nb_etat_civil] = Mk_Integer(1010);
          nb_etat_civil++;
        }
170     if(strcmp(getenv("WWW_v1011"),"checked") == 0){
```

```
      tableau_etat_civil[nb_etat_civil] = Mk_Integer(1011);
      nb_etat_civil++;
    }
    if(strcmp(getenv("WWW_v1012"),"checked") == 0){
      tableau_etat_civil[nb_etat_civil] = Mk_Integer(1012);
      nb_etat_civil++;
    }
    if(strcmp(getenv("WWW_v1013"),"checked") == 0){
      tableau_etat_civil[nb_etat_civil] = Mk_Integer(1013);
      nb_etat_civil++;
    }
    if(strcmp(getenv("WWW_v1014"),"checked") == 0){
      tableau_etat_civil[nb_etat_civil] = Mk_Integer(1014);
      nb_etat_civil++;
    }
    if(strcmp(getenv("WWW_v1015"),"checked") == 0){
      tableau_etat_civil[nb_etat_civil] = Mk_Integer(1015);
      nb_etat_civil++;
    }
    if(strcmp(getenv("WWW_v1016"),"checked") == 0){
      tableau_etat_civil[nb_etat_civil] = Mk_Integer(1016);
      nb_etat_civil++;
    }
    if(strcmp(getenv("WWW_v1017"),"checked") == 0){
      tableau_etat_civil[nb_etat_civil] = Mk_Integer(1017);
      nb_etat_civil++;
    }
    if(strcmp(getenv("WWW_v1018"),"checked") == 0){
      tableau_etat_civil[nb_etat_civil] = Mk_Integer(1018);
      nb_etat_civil++;
    }
    if(strcmp(getenv("WWW_v1019"),"checked") == 0){
      tableau_etat_civil[nb_etat_civil] = Mk_Integer(1019);
      nb_etat_civil++;
    }
    if(strcmp(getenv("WWW_v1020"),"checked") == 0){
      tableau_etat_civil[nb_etat_civil] = Mk_Integer(1020);
      nb_etat_civil++;
    }
    if(strcmp(getenv("WWW_v1021"),"checked") == 0){
      tableau_etat_civil[nb_etat_civil] = Mk_Integer(1021);
      nb_etat_civil++;
    }
```

```
        // Creation de la liste qui contient l'etat civil du contribuable
215     liste_etat_civil = Mk_Proper_List(nb_etat_civil,tableau_etat_civil);



        // Statut d'handicape
220     // Si les case v1028 ou v1029 sont cochees
        if(strcmp(getenv("WWW_v1028"),"checked") == 0 ||
           strcmp(getenv("WWW_v1029"),"checked") == 0){
          handicape = Mk_String("handicape");
        }
225     else{
          handicape = Mk_String("pashandicape");
        }

        // Charge de famille
230     Nb_Enfant_Total = Mk_Integer(atoi(getenv("WWW_v1030")));
        Nb_Enfant_Handicape = Mk_Integer(atoi(getenv("WWW_v1031")));
        Nb_Enfant_Moitie = Mk_Integer(atoi(getenv("WWW_v1034")) +
                                      atoi(getenv("WWW_v1036")));
        Nb_Enfant_Moitie_Handicape = Mk_Integer(atoi(getenv("WWW_v1035")) +
235                                             atoi(getenv("WWW_v1037")));
        Nb_Enfant_3ans = Mk_Integer(atoi(getenv("WWW_v1038")));
        Nb_Enfant_3ans_Handicape  = Mk_Integer(atoi(getenv("WWW_v1039")));
        Nb_Pers_Charge = Mk_Integer(atoi(getenv("WWW_v1032")));
        Nb_Pers_Charge_Handicape = Mk_Integer(atoi(getenv("WWW_v1033")));
240     Nb_Pers_Charge_65ans = Mk_Integer(atoi(getenv("WWW_v1043")));
        Nb_Pers_Charge_Handicape_65ans = Mk_Integer(atoi(getenv("WWW_v1044")));


        // CADRE3
245     // Calcul du revenu cadastral soumis au precompte immobilier
        revenus_cadastral_homme = (atof(getenv("WWW_v1100")) +
                                   atof(getenv("WWW_v1106")) +
                                   atof(getenv("WWW_v1107")) +
                                   atof(getenv("WWW_v1108")) +
250                                atof(getenv("WWW_v1109")) +
                                   atof(getenv("WWW_v1112")) +
                                   atof(getenv("WWW_v1115")));
        // Calcul du revenu cadastral non soumis au precompte immobilier
        revenus_cadastral_sans_precompte_h = atof(getenv("WWW_v1101"));
255
        // Multiplication par 100 et arrondi pour que prolog puisse traiter la somme
```

```
        RC_homme = Mk_Integer((int)(revenus_cadastral_homme*100));

        // CADRE4
        remunerations_ordinaires = (atof(getenv("WWW_v1250")) +
                                    atof(getenv("WWW_v1249")) +
                                    atof(getenv("WWW_v1248")) +
                                    atof(getenv("WWW_v1251")) +
                                    atof(getenv("WWW_v1252")) +
                                    atof(getenv("WWW_v1253")) +
                                    atof(getenv("WWW_v1245")) +
                                    atof(getenv("WWW_v1291")));
        frais_deplacement = atof(getenv("WWW_v1254"));
        frais_deplacement_exonere = atof(getenv("WWW_v1255"));
        forfait_long_deplacement  = atof(getenv("WWW_v1256"));
        allocation_chomage = (atof(getenv("WWW_v1260")) +
                              atof(getenv("WWW_v1261")) +
                              atof(getenv("WWW_v1262")) +
                              atof(getenv("WWW_v1263")) +
                              atof(getenv("WWW_v1264")) +
                              atof(getenv("WWW_v1265")));
        indemnite_maladie = (atof(getenv("WWW_v1266")) +
                             atof(getenv("WWW_v1268")));
        revenus_remplacement = (atof(getenv("WWW_v1292")) +
                                atof(getenv("WWW_v1293")) +
                                atof(getenv("WWW_v1296")) +
                                atof(getenv("WWW_v1294")) +
                                atof(getenv("WWW_v1295")) +
                                atof(getenv("WWW_v1269")) +
                                atof(getenv("WWW_v1270")) +
                                atof(getenv("WWW_v1271")) +
                                atof(getenv("WWW_v1272")));
        prepension = (atof(getenv("WWW_v1281")) +
                      atof(getenv("WWW_v1282")));
        Pl_nb_heure_sup = Mk_Integer(atoi(getenv("WWW_v1246")));
        heure_sup = atof(getenv("WWW_v1247"));
        Pl_heure_sup = Mk_Integer((int)(heure_sup*100));
        precompte_prof = atof(getenv("WWW_v1286"));
        retenue_sec_sociale = atof(getenv("WWW_v1287"));

        // CADRE 5
        pension = (atof(getenv("WWW_v1211")) +
                   atof(getenv("WWW_v1212")));
        precompte_prof_pension = atof(getenv("WWW_v1225"));
```

```
300
        precompte_prof_total = precompte_prof + precompte_prof_pension;

        salaire_imposable_pour_frais_prof =
          remunerations_ordinaires +
305       frais_deplacement -
          frais_deplacement_exonere;

        salaire_imposable =
          remunerations_ordinaires +
310       frais_deplacement -
          frais_deplacement_exonere +
          allocation_chomage +
          indemnite_maladie +
          revenus_remplacement +
315       prepension +
          heure_sup +
          pension;

        pension_prepension_autre = revenus_remplacement + prepension + pension;
320
        // CADRE6

        rente1 = atof(getenv("WWW_v1192"));
        rente2 = atof(getenv("WWW_v1193"));
325     rente3 = atof(getenv("WWW_v1194"));

        // CADRE7
        elements_deductibles = (atof(getenv("WWW_v1350")) +
                                atof(getenv("WWW_v1349")) +
330                             atof(getenv("WWW_v1390")) +
                                atof(getenv("WWW_v1392")) +
                                atof(getenv("WWW_v1393")) +
                                atof(getenv("WWW_v1385")) +
                                atof(getenv("WWW_v1387")) +
335                             atof(getenv("WWW_v1388")) +
                                atof(getenv("WWW_v1389")));

        // CADRE8
        epargne_logement = (atof(getenv("WWW_v1355")) +
340                         atof(getenv("WWW_v1356")) +
                            atof(getenv("WWW_v1357")) +
                            atof(getenv("WWW_v1351")) +
```

```
                        atof(getenv("WWW_v1352")));
    epargne_long_terme = (atof(getenv("WWW_v1358")) +
345                         atof(getenv("WWW_v1359")) +
                            atof(getenv("WWW_v1360")) +
                            atof(getenv("WWW_v1353")) +
                            atof(getenv("WWW_v1354")));

350 // CADRE9
    versement_ALE = atof(getenv("WWW_v1365"));
    versement_titre_service = atof(getenv("WWW_v1364"));
    pret_gagnant_gagnant = atof(getenv("WWW_v1378"));
    pret_gagnant_gagnant_perdu = atof(getenv("WWW_v1379"));
355 economie_energie = (atof(getenv("WWW_v1363")) +
                        atof(getenv("WWW_v1369")));
    renovation_zone_active = atof(getenv("WWW_v1396"));
    voiture_moins_105 = atof(getenv("WWW_v1380"));
    voiture_plus_105 = atof(getenv("WWW_v1381"));
360

    // CADRE 10
    arkimedes = atof(getenv("WWW_v1397"));
    internet = atof(getenv("WWW_v1398"));
365
    // CADRE 11
    versement_anticipe = atof(getenv("WWW_v1570"));

    /*******************EXECUTION DU PROGRAMME PROLOG*******************/
370

    // Calcul du precompte immmobilier
    // Recherche de la cle interne du predicat a utiliser
    predicat = Find_Atom("precompte_immobilier");
375 // Debut d'une requete
    Pl_Query_Begin(TRUE);
    // Preparation des arguments de la requête
    PlTerm arg_PI[2];
    arg_PI[0] = RC_homme;
380 arg_PI[1] = Mk_Variable();
    // Appel de la requete
    Pl_Query_Call(predicat,2,arg_PI);
    // Recuperation du resultat
    int PI = Rd_Integer(arg_PI[1]);
385 // Fin de la requete et liberation de la memoire de GNU Prolog
```

```
        Pl_Query_End(PL_RECOVER);


        // Calcul des frais professionnels forfaitaires
390     // Recherche de la cle interne du predicat a utiliser
        predicat = Find_Atom("frais_professionnel_forfaitaire");
        // Debut d'une requete
        Pl_Query_Begin(TRUE);
        // Preparation des arguments de la requête
395     PlTerm arg_FP[2];
        arg_FP[0] = Mk_Integer((int)(salaire_imposable_pour_frais_prof*100));
        arg_FP[1] = Mk_Variable();
        // Appel de la requete
        Pl_Query_Call(predicat,2,arg_FP);
400     // Recuperation du resultat
        int FP = Rd_Integer(arg_FP[1]);
        // Fin de la requete et liberation de la memoire de GNU Prolog
        Pl_Query_End(PL_RECOVER);

405
        imposable_taux_plein =
          salaire_imposable -
          (float)FP/100 -
          forfait_long_deplacement;
410
        // Verification imposition commune
        // Recherche de la cle interne du predicat a utiliser
        predicat = Find_Atom("isole");
        // Debut d'une requete
415     Pl_Query_Begin(TRUE);
        // Preparation des arguments de la requête
        PlTerm arg_IC[1];
        arg_IC[0] = liste_etat_civil;
        // Appel de la requete
420     pas_imposition_commune = Pl_Query_Call(predicat,1,arg_IC);
        // Fin de la requete et liberation de la memoire de GNU Prolog
        Pl_Query_End(PL_RECOVER);

        // Calcul du quotient conjugal si imposition commune
425     int QC = 0; // quotient conjugal
        if(pas_imposition_commune != 1){
```

```
        // Calcul du quotient conjugal
430     // Recherche de la cle interne du predicat a utiliser
        predicat = Find_Atom("quotient_conjugal");
        // Debut d'une requete
        Pl_Query_Begin(TRUE);
        // Preparation des arguments de la requête
435     PlTerm arg_QC[3];
        arg_QC[0] = Mk_Integer((int)(imposable_taux_plein*100));
        arg_QC[1] = Mk_Integer(0);
        arg_QC[2] = Mk_Variable();
        // Appel de la requete
440     Pl_Query_Call(predicat,3,arg_QC);
        // Recuperation du resultat
        QC = Rd_Integer(arg_QC[2]);

        revenu_homme_hors_deduction = imposable_taux_plein - ((float)QC/100);
445     revenu_femme_hors_deduction = (float)QC/100;

        revenu_homme = revenu_homme_hors_deduction - (elements_deductibles*0.7);
        revenu_femme = revenu_femme_hors_deduction - (elements_deductibles*0.3);
        // Fin de la requete et liberation de la memoire de GNU Prolog
450     Pl_Query_End(PL_RECOVER);
      }
      else{ // pas d'imposition commune
        revenu = imposable_taux_plein - elements_deductibles;
      }
455

      // Calcul du montant exempte d'impôt
      // Recherche de la cle interne du predicat a utiliser
      predicat = Find_Atom("montant_exempte");
460   // Debut d'une requete
      Pl_Query_Begin(TRUE);
      // Preparation des arguments de la requête
      PlTerm arg_ME[22];
      arg_ME[0] = liste_etat_civil;
465   arg_ME[1] = handicape;
      arg_ME[2] = Nb_Enfant_Total;
      arg_ME[3] = Nb_Enfant_Handicape;
      arg_ME[4] = Nb_Enfant_Moitie;
      arg_ME[5] = Nb_Enfant_Moitie_Handicape;
470   arg_ME[6] = Nb_Enfant_3ans;
      arg_ME[7] = Nb_Enfant_3ans_Handicape;
```

```
        arg_ME[8] = Nb_Pers_Charge;
        arg_ME[9] = Nb_Pers_Charge_Handicape;
        arg_ME[10] = Nb_Pers_Charge_65ans;
475     arg_ME[11] = Nb_Pers_Charge_Handicape_65ans;
        arg_ME[12] = Mk_Variable();
        arg_ME[13] = Mk_Variable();
        arg_ME[14] = Mk_Variable();
        arg_ME[15] = Mk_Variable();
480     arg_ME[16] = Mk_Variable();
        arg_ME[17] = Mk_Variable();
        arg_ME[18] = Mk_Variable();
        arg_ME[19] = Mk_Variable();
        arg_ME[20] = Mk_Variable();
485     arg_ME[21] = Mk_Variable();
        // Appel de la requete
        Pl_Query_Call(predicat,22,arg_ME);
        // Recuperation des resultats
        int bonus_handicape = Rd_Integer(arg_ME[12]);
490     int bonus_isole = Rd_Integer(arg_ME[13]);
        int bonus_marie = Rd_Integer(arg_ME[14]);
        int bonus_cohabitant = Rd_Integer(arg_ME[15]);
        int bonus_enfant = Rd_Integer(arg_ME[16]);
        int bonus_enfant_moitie = Rd_Integer(arg_ME[17]);
495     int bonus_enfant_3ans = Rd_Integer(arg_ME[18]);
        int bonus_pers_charge = Rd_Integer(arg_ME[19]);
        int bonus_pers_charge_65 = Rd_Integer(arg_ME[20]);
        int ME = Rd_Integer(arg_ME[21]);
        // Fin de la requete et liberation de la memoire de GNU Prolog
500     Pl_Query_End(PL_RECOVER);

        // En cas de declaration commune, on effectue la repartition des montants
        // exemptes d'impot.
        int repartition_ME_homme;
505     int repartition_ME_femme;
        if(pas_imposition_commune != 1){
          // Repartition des montants exemptes
          // Recherche de la cle interne du predicat a utiliser
          predicat = Find_Atom("repartition_montant_exempte");
510       // Debut d'une requete
          Pl_Query_Begin(TRUE);
          // Preparation des arguments de la requête
          PlTerm arg_repartition_ME[6];
          arg_repartition_ME[0] = Mk_Integer((int)(revenu_homme*100));
```

```
515       arg_repartition_ME[1] = Mk_Integer((int)(revenu_femme*100));
          arg_repartition_ME[2] = arg_ME[21];
          arg_repartition_ME[3] = Mk_Integer(594000);
          arg_repartition_ME[4] = Mk_Variable();
          arg_repartition_ME[5] = Mk_Variable();
520       // Appel de la requete
          Pl_Query_Call(predicat,6,arg_repartition_ME);
          // Recuperation des resultats
          repartition_ME_homme = Rd_Integer(arg_repartition_ME[4]);
          repartition_ME_femme = Rd_Integer(arg_repartition_ME[5]);
525       // Fin de la requete et liberation de la memoire de GNU Prolog
          Pl_Query_End(PL_RECOVER);
        }

        // On calcul l'impot sur les revenus et sur les montants exemptes ainsi
530     // que sur les rentes alimentaires.
        int impot_revenus_homme;
        int impot_revenus_femme;
        int impot_revenus;
        int impot_ME_homme;
535     int impot_ME_femme;
        int impot_ME;
        int impot_rente1 = 0;
        int impot_rente2 = 0;
        int impot_rente3 = 0;
540     // Cas d'une imposition commune
        if(pas_imposition_commune != 1){
          // Recherche de la cle interne du predicat a utiliser
          predicat = Find_Atom("impot");

545
          // Calcul de l'impot sur les REVENUS de l'homme.
          // Debut d'une requete
          Pl_Query_Begin(TRUE);
          // Preparation des arguments de la requête
550       PlTerm arg_impot_revenu_homme[2];
          arg_impot_revenu_homme[0] = Mk_Integer((int)(revenu_homme*100));
          arg_impot_revenu_homme[1] = Mk_Variable();
          // Appel de la requete
          Pl_Query_Call(predicat,2,arg_impot_revenu_homme);
555       // Recuperation des resultats
          impot_revenus_homme = Rd_Integer(arg_impot_revenu_homme[1]);
          // Fin de la requete et liberation de la memoire de GNU Prolog
```

```
        Pl_Query_End(PL_RECOVER);

560
        // Calcul de l'impot sur les REVENUS de la femme.
        // Debut d'une requete
        Pl_Query_Begin(TRUE);
        // Preparation des arguments de la requête
565     PlTerm arg_impot_revenu_femme[2];
        arg_impot_revenu_femme[0] = Mk_Integer((int)(revenu_femme*100));
        arg_impot_revenu_femme[1] = Mk_Variable();
        // Appel de la requete
        Pl_Query_Call(predicat,2,arg_impot_revenu_femme);
570     // Recuperation des resultats
        impot_revenus_femme = Rd_Integer(arg_impot_revenu_femme[1]);
        // Fin de la requete et liberation de la memoire de GNU Prolog
        Pl_Query_End(PL_RECOVER);

575
        // Calcul de l'impot sur le montant exempte de l'homme.
        // Debut d'une requete
        Pl_Query_Begin(TRUE);
        // Preparation des arguments de la requête
580     PlTerm arg_impot_ME_homme[2];
        arg_impot_ME_homme[0] = Mk_Integer(repartition_ME_homme);
        arg_impot_ME_homme[1] = Mk_Variable();
        // Appel de la requete
        Pl_Query_Call(predicat,2,arg_impot_ME_homme);
585     // Recuperation des resultats
        impot_ME_homme = Rd_Integer(arg_impot_ME_homme[1]);
        // Fin de la requete et liberation de la memoire de GNU Prolog
        Pl_Query_End(PL_RECOVER);

590
        // Calcul de l'impot sur le montant exempte de la femme.
        // Debut d'une requete
        Pl_Query_Begin(TRUE);
        // Preparation des arguments de la requête
595     PlTerm arg_impot_ME_femme[2];
        arg_impot_ME_femme[0] = Mk_Integer(repartition_ME_femme);
        arg_impot_ME_femme[1] = Mk_Variable();
        // Appel de la requete
        Pl_Query_Call(predicat,2,arg_impot_ME_femme);
600     // Recuperation des resultats
```

```
        impot_ME_femme = Rd_Integer(arg_impot_ME_femme[1]);
        // Fin de la requete et liberation de la memoire de GNU Prolog
        Pl_Query_End(PL_RECOVER);
      }
      else{ // Cas d'une imposition non commune
        // Recherche de la cle interne du predicat a utiliser
        predicat = Find_Atom("impot");


        // Calcul de l'impot sur les REVENUS.
        // Debut d'une requete
        Pl_Query_Begin(TRUE);
        // Preparation des arguments de la requête
        PlTerm arg_impot_revenu[2];
        arg_impot_revenu[0] = Mk_Integer((int)(revenu*100));
        arg_impot_revenu[1] = Mk_Variable();
        // Appel de la requete
        Pl_Query_Call(predicat,2,arg_impot_revenu);
        // Recuperation des resultats
        impot_revenus = Rd_Integer(arg_impot_revenu[1]);
        // Fin de la requete et liberation de la memoire de GNU Prolog
        Pl_Query_End(PL_RECOVER);


        // Calcul de l'impot sur le montant exempte de l'homme.
        // Debut d'une requete
        Pl_Query_Begin(TRUE);
        // Preparation des arguments de la requête
        PlTerm arg_impot_ME[2];
        arg_impot_ME[0] = Mk_Integer(ME);
        arg_impot_ME[1] = Mk_Variable();
        // Appel de la requete
        Pl_Query_Call(predicat,2,arg_impot_ME);
        // Recuperation des resultats
        impot_ME = Rd_Integer(arg_impot_ME[1]);
        // Fin de la requete et liberation de la memoire de GNU Prolog
        Pl_Query_End(PL_RECOVER);
      }


      // Calcul de l'impot sur les rentes alimentaires.
      // Recherche de la cle interne du predicat a utiliser
      predicat = Find_Atom("impot");
```

```
645     // Calcul de l'impot sur la rente alimentaire 1.
        // Debut d'une requete
        Pl_Query_Begin(TRUE);
        // Preparation des arguments de la requête
        PlTerm arg_impot_rente1[2];
650     // On multiplie seulement par 80 car ce montant n'est impose qu'a
        // concurrence de 80%
        arg_impot_rente1[0] = Mk_Integer((int)(rente1*80));
        arg_impot_rente1[1] = Mk_Variable();
        // Appel de la requete
655     Pl_Query_Call(predicat,2,arg_impot_rente1);
        // Recuperation des resultats
        impot_rente1 = Rd_Integer(arg_impot_rente1[1]);
        // Fin de la requete et liberation de la memoire de GNU Prolog
        Pl_Query_End(PL_RECOVER);
660

        // Calcul de l'impot sur la rente alimentaire 2.
        // Debut d'une requete
        Pl_Query_Begin(TRUE);
665     // Preparation des arguments de la requête
        PlTerm arg_impot_rente2[2];
        // On multiplie seulement par 80 car ce montant n'est impose qu'a
        // concurrence de 80%
        arg_impot_rente2[0] = Mk_Integer((int)(rente2*80));
670     arg_impot_rente2[1] = Mk_Variable();
        // Appel de la requete
        Pl_Query_Call(predicat,2,arg_impot_rente2);
        // Recuperation des resultats
        impot_rente2 = Rd_Integer(arg_impot_rente2[1]);
675     // Fin de la requete et liberation de la memoire de GNU Prolog
        Pl_Query_End(PL_RECOVER);


        // Calcul de l'impot sur la rente alimentaire 3.
680     // Debut d'une requete
        Pl_Query_Begin(TRUE);
        // Preparation des arguments de la requête
        PlTerm arg_impot_rente3[2];
        // On multiplie seulement par 80 car ce montant n'est impose qu'a
685     // concurrence de 80%
        arg_impot_rente3[0] = Mk_Integer((int)(rente3*80));
```

```
        arg_impot_rente3[1] = Mk_Variable();
        // Appel de la requete
        Pl_Query_Call(predicat,2,arg_impot_rente3);
690     // Recuperation des resultats
        impot_rente3 = Rd_Integer(arg_impot_rente3[1]);
        // Fin de la requete et liberation de la memoire de GNU Prolog
        Pl_Query_End(PL_RECOVER);

695
        // Calcul des reductions d'impot

        // Reduction epargne-logement
        int limite_epargne_logement = 0;
700     int min_epargne_logement = 0;
        int reduction_epargne_logement = 0;

        // Calcul de la limitation de la part entrant en compte pour la reduction
        // Recherche de la cle interne du predicat a utiliser
705     predicat = Find_Atom("limite_epargne_logement");
        // Debut d'une requete
        Pl_Query_Begin(TRUE);
        // Preparation des arguments de la requête
        PlTerm arg_limite_epargne_logement[2];
710     if(pas_imposition_commune == 1){ // Cas d'une declaration non commune
          arg_limite_epargne_logement[0] = Mk_Integer((int)
                                                (imposable_taux_plein*100));
        }
        else{ // Cas d'une declaration commune
715       arg_limite_epargne_logement[0] = Mk_Integer((int)
                                                (revenu_homme_hors_deduction
                                                *100));
        }
        arg_limite_epargne_logement[1] = Mk_Variable();
720     // Appel de la requete
        Pl_Query_Call(predicat,2,arg_limite_epargne_logement);
        // Recuperation des resultats
        limite_epargne_logement = Rd_Integer(arg_limite_epargne_logement[1]);
        // On prend le minimum entre la limite et le montant obtenu.
725     if(limite_epargne_logement < (int)(epargne_logement*100)){
          min_epargne_logement = limite_epargne_logement;
        }
        else{
          min_epargne_logement = (int)(epargne_logement*100);
```

```
730     }
        // Fin de la requete et liberation de la memoire de GNU Prolog
        Pl_Query_End(PL_RECOVER);


735     // Calcul de la reduction accordee pour l'epargne-logement
        // Recherche de la cle interne du predicat a utiliser
        predicat = Find_Atom("reduction_epargne_logement");
        // Debut d'une requete
        Pl_Query_Begin(TRUE);
740     // Preparation des arguments de la requête
        PlTerm arg_epargne_logement[3];
        if(pas_imposition_commune == 1){ // Cas d'une declaration non commune
          arg_epargne_logement[0] = Mk_Integer((int)(imposable_taux_plein*100));
        }
745     else{ // Cas d'une declaration commune
          arg_epargne_logement[0] = Mk_Integer((int)
                                              (revenu_homme_hors_deduction*100));
        }
        arg_epargne_logement[1] = Mk_Integer(min_epargne_logement);
750     arg_epargne_logement[2] = Mk_Variable();
        // Appel de la requete
        Pl_Query_Call(predicat,3,arg_epargne_logement);
        // Recuperation des resultats
        reduction_epargne_logement = Rd_Integer(arg_epargne_logement[2]);
755     // Fin de la requete et liberation de la memoire de GNU Prolog
        Pl_Query_End(PL_RECOVER);


        // Reduction epargne long-terme
760     int reduction_epargne_long_terme;
        // Recherche de la cle interne du predicat a utiliser
        predicat = Find_Atom("reduction_epargne_long_terme");
        // Debut d'une requete
        Pl_Query_Begin(TRUE);
765     // Preparation des arguments de la requête
        PlTerm arg_epargne_long_terme[4];
        if(pas_imposition_commune == 1){ // Cas d'une declaration non commune
          arg_epargne_long_terme[0] = Mk_Integer((int)(revenu*100));
          arg_epargne_long_terme[1] = Mk_Integer(ME);
770     }
        else{ // Cas d'une declaration commune
          arg_epargne_long_terme[0] = Mk_Integer((int)(revenu_homme*100));
```

```
      arg_epargne_long_terme[1] = Mk_Integer(repartition_ME_femme);
    }
    arg_epargne_long_terme[2] = Mk_Integer((int)(epargne_long_terme*100));
    arg_epargne_long_terme[3] = Mk_Variable();
    // Appel de la requete
    Pl_Query_Call(predicat,4,arg_epargne_long_terme);
    // Recuperation des resultats
    reduction_epargne_long_terme = Rd_Integer(arg_epargne_long_terme[3]);
    // Fin de la requete et liberation de la memoire de GNU Prolog
    Pl_Query_End(PL_RECOVER);


    // Reduction depense ALE
    int reduction_ALE;
    // Recherche de la cle interne du predicat a utiliser
    // On utilise le meme predicat que pour l'epargne long terme
    predicat = Find_Atom("reduction_epargne_long_terme");
    // Debut d'une requete
    Pl_Query_Begin(TRUE);
    // Preparation des arguments de la requête
    PlTerm arg_ALE[4];
    if(pas_imposition_commune == 1){ // Cas d'une declaration non commune
      arg_ALE[0] = Mk_Integer((int)(revenu*100));
      arg_ALE[1] = Mk_Integer(ME);
    }
    else{ // Cas d'une declaration commune
      arg_ALE[0] = Mk_Integer((int)(revenu_homme*100));
      arg_ALE[1] = Mk_Integer(repartition_ME_femme);
    }
    arg_ALE[2] = Mk_Integer((int)(versement_ALE*100));
    arg_ALE[3] = Mk_Variable();
    // Appel de la requete
    Pl_Query_Call(predicat,4,arg_ALE);
    // Recuperation des resultats
    reduction_ALE = Rd_Integer(arg_ALE[3]);
    // Fin de la requete et liberation de la memoire de GNU Prolog
    Pl_Query_End(PL_RECOVER);


    // Reduction titre-service
    int reduction_titre_service;
    // Recherche de la cle interne du predicat a utiliser
    predicat = Find_Atom("reduction_titre_service");
```

```
        // Debut d'une requete
        Pl_Query_Begin(TRUE);
        // Preparation des arguments de la requête
        PlTerm arg_titre_service[2];
820     arg_titre_service[0] = Mk_Integer((int)(versement_titre_service*100));
        arg_titre_service[1] = Mk_Variable();
        // Appel de la requete
        Pl_Query_Call(predicat,2,arg_titre_service);
        // Recuperation des resultats
825     reduction_titre_service = Rd_Integer(arg_titre_service[1]);
        // Fin de la requete et liberation de la memoire de GNU Prolog
        Pl_Query_End(PL_RECOVER);


830     // Reduction pret gagnant-gagnant
        int reduction_gagnant;
        // Recherche de la cle interne du predicat a utiliser
        predicat = Find_Atom("reduction_pret_gagnant");
        // Debut d'une requete
835     Pl_Query_Begin(TRUE);
        // Preparation des arguments de la requête
        PlTerm arg_gagnant_gagnant[2];
        arg_gagnant_gagnant[0] = Mk_Integer((int)(pret_gagnant_gagnant*100));
        arg_gagnant_gagnant[1] = Mk_Variable();
840     // Appel de la requete
        Pl_Query_Call(predicat,2,arg_gagnant_gagnant);
        // Recuperation des resultats
        reduction_gagnant = Rd_Integer(arg_gagnant_gagnant[1]);
        // Fin de la requete et liberation de la memoire de GNU Prolog
845     Pl_Query_End(PL_RECOVER);


        // Reduction pret gagnant-gagnant perdu
        int reduction_gagnant_perdu;
850     // Recherche de la cle interne du predicat a utiliser
        predicat = Find_Atom("reduction_pret_gagnant_perdu");
        // Debut d'une requete
        Pl_Query_Begin(TRUE);
        // Preparation des arguments de la requête
855     PlTerm arg_gagnant_gagnant_perdu[2];
        arg_gagnant_gagnant_perdu[0] = Mk_Integer((int)
                                                (pret_gagnant_gagnant_perdu*100));
        arg_gagnant_gagnant_perdu[1] = Mk_Variable();
```

```
        // Appel de la requete
860     Pl_Query_Call(predicat,2,arg_gagnant_gagnant_perdu);
        // Recuperation des resultats
        reduction_gagnant_perdu = Rd_Integer(arg_gagnant_gagnant_perdu[1]);
        // Fin de la requete et liberation de la memoire de GNU Prolog
        Pl_Query_End(PL_RECOVER);
865

        // Reduction economie energie
        int reduction_economie_energie;
        // Recherche de la cle interne du predicat a utiliser
870     predicat = Find_Atom("reduction_economie_energie");
        // Debut d'une requete
        Pl_Query_Begin(TRUE);
        // Preparation des arguments de la requête
        PlTerm arg_economie_energie[2];
875     arg_economie_energie[0] = Mk_Integer((int)(economie_energie*100));
        arg_economie_energie[1] = Mk_Variable();
        // Appel de la requete
        Pl_Query_Call(predicat,2,arg_economie_energie);
        // Recuperation des resultats
880     reduction_economie_energie = Rd_Integer(arg_economie_energie[1]);
        // Fin de la requete et liberation de la memoire de GNU Prolog
        Pl_Query_End(PL_RECOVER);


885     // Reduction renovation zone action positive
        int reduction_renovation;
        // Recherche de la cle interne du predicat a utiliser
        predicat = Find_Atom("reduction_depense_renovation");
        // Debut d'une requete
890     Pl_Query_Begin(TRUE);
        // Preparation des arguments de la requête
        PlTerm arg_renovation[2];
        arg_renovation[0] = Mk_Integer((int)(renovation_zone_active*100));
        arg_renovation[1] = Mk_Variable();
895     // Appel de la requete
        Pl_Query_Call(predicat,2,arg_renovation);
        // Recuperation des resultats
        reduction_renovation = Rd_Integer(arg_renovation[1]);
        // Fin de la requete et liberation de la memoire de GNU Prolog
900     Pl_Query_End(PL_RECOVER);
```

```
        // Reduction voiture dont l'emission est de moins de 105 gr de CO2.
        int reduction_voiture_moins_105;
905     // Recherche de la cle interne du predicat a utiliser
        predicat = Find_Atom("reduction_depense_vehicule_moins_105");
        // Debut d'une requete
        Pl_Query_Begin(TRUE);
        // Preparation des arguments de la requête
910     PlTerm arg_voiture_moins_105[2];
        arg_voiture_moins_105[0] = Mk_Integer((int)(voiture_moins_105*100));
        arg_voiture_moins_105[1] = Mk_Variable();
        // Appel de la requete
        Pl_Query_Call(predicat,2,arg_voiture_moins_105);
915     // Recuperation des resultats
        reduction_voiture_moins_105 = Rd_Integer(arg_voiture_moins_105[1]);
        // Fin de la requete et liberation de la memoire de GNU Prolog
        Pl_Query_End(PL_RECOVER);

920
        // Reduction voiture dont l'emission est entre 105gr et 115gr de CO2.
        int reduction_voiture_plus_105;
        // Recherche de la cle interne du predicat a utiliser
        predicat = Find_Atom("reduction_depense_vehicule_105_115");
925     // Debut d'une requete
        Pl_Query_Begin(TRUE);
        // Preparation des arguments de la requête
        PlTerm arg_voiture_plus_105[2];
        arg_voiture_plus_105[0] = Mk_Integer((int)(voiture_plus_105*100));
930     arg_voiture_plus_105[1] = Mk_Variable();
        // Appel de la requete
        Pl_Query_Call(predicat,2,arg_voiture_plus_105);
        // Recuperation des resultats
        reduction_voiture_plus_105 = Rd_Integer(arg_voiture_plus_105[1]);
935     // Fin de la requete et liberation de la memoire de GNU Prolog
        Pl_Query_End(PL_RECOVER);


        // Reduction pour allocation de chomage.
940     int reduction_allocation_chomage;
        // Recherche de la cle interne du predicat a utiliser
        predicat = Find_Atom("reduction_impot_chomage");
        // Debut d'une requete
        Pl_Query_Begin(TRUE);
```

```
945     // Preparation des arguments de la requête
        PlTerm arg_alloc_chomage[5];
        arg_alloc_chomage[0] = liste_etat_civil;
        // Cas d'une declaration non commune.
        if(pas_imposition_commune == 1){
950       arg_alloc_chomage[1] = Mk_Integer((int)(imposable_taux_plein*100));
          arg_alloc_chomage[2] = Mk_Integer((int)(revenu*100));
        }
        else{ // Cas d'une declaration commune
          arg_alloc_chomage[1] = Mk_Integer((int)(revenu_homme_hors_deduction*100));
955       arg_alloc_chomage[2] = Mk_Integer((int)(revenu_homme*100));
        }
        arg_alloc_chomage[3] = Mk_Integer((int)(allocation_chomage*100));
        arg_alloc_chomage[4] = Mk_Variable();
        // Appel de la requete
960     Pl_Query_Call(predicat,5,arg_alloc_chomage);
        // Recuperation des resultats
        reduction_allocation_chomage = Rd_Integer(arg_alloc_chomage[4]);
        // Fin de la requete et liberation de la memoire de GNU Prolog
        Pl_Query_End(PL_RECOVER);
965


        // Reduction pour indemnite maladie-invalidite.
        int reduction_indemnite_maladie;
        // Recherche de la cle interne du predicat a utiliser
970     predicat = Find_Atom("reduction_impot_indemnite");
        // Debut d'une requete
        Pl_Query_Begin(TRUE);
        // Preparation des arguments de la requête
        PlTerm arg_indemnite_maladie[4];
975     // Cas d'une declaration non commune.
        if(pas_imposition_commune == 1){
          arg_indemnite_maladie[0] = Mk_Integer((int)(imposable_taux_plein*100));
          arg_indemnite_maladie[1] = Mk_Integer((int)(revenu*100));
        }
980     else{ // Cas d'une declaration commune
          arg_indemnite_maladie[0] = Mk_Integer((int)
                                              (revenu_homme_hors_deduction*100));
          arg_indemnite_maladie[1] = Mk_Integer((int)(revenu_homme*100));
        }
985     arg_indemnite_maladie[2] = Mk_Integer((int)(indemnite_maladie*100));
        arg_indemnite_maladie[3] = Mk_Variable();
        // Appel de la requete
```

```
        Pl_Query_Call(predicat,4,arg_indemnite_maladie);
        // Recuperation des resultats
990     reduction_indemnite_maladie = Rd_Integer(arg_indemnite_maladie[3]);
        // Fin de la requete et liberation de la memoire de GNU Prolog
        Pl_Query_End(PL_RECOVER);


995     // Reduction pour pensions, prepensions et autres revenus de remplacement.
        int reduction_pension;
        // Recherche de la cle interne du predicat a utiliser
        predicat = Find_Atom("reduction_impot_pensions");
        // Debut d'une requete
1000    Pl_Query_Begin(TRUE);
        // Preparation des arguments de la requête
        PlTerm arg_pension[4];
        // Cas d'une declaration non commune.
        if(pas_imposition_commune == 1){
1005      arg_pension[0] = Mk_Integer((int)(imposable_taux_plein*100));
          arg_pension[1] = Mk_Integer((int)(revenu*100));
        }
        else{ // Cas d'une declaration commune
          arg_pension[0] = Mk_Integer((int)(revenu_homme_hors_deduction*100));
1010      arg_pension[1] = Mk_Integer((int)(revenu_homme*100));
        }
        arg_pension[2] = Mk_Integer((int)(pension_prepension_autre*100));
        arg_pension[3] = Mk_Variable();
        // Appel de la requete
1015    Pl_Query_Call(predicat,4,arg_pension);
        // Recuperation des resultats
        reduction_pension = Rd_Integer(arg_pension[3]);
        // Fin de la requete et liberation de la memoire de GNU Prolog
        Pl_Query_End(PL_RECOVER);
1020


        // Reduction pour sursalaire.
        int reduction_sursalaire;
        // Recherche de la cle interne du predicat a utiliser
1025    predicat = Find_Atom("reduction_impot_sursalaire");
        // Debut d'une requete
        Pl_Query_Begin(TRUE);
        // Preparation des arguments de la requête
        PlTerm arg_sursalaire[3];
1030    arg_sursalaire[0] = Pl_nb_heure_sup;
```

```
        arg_sursalaire[1] = Pl_heure_sup;
        arg_sursalaire[2] = Mk_Variable();
        // Appel de la requete
        Pl_Query_Call(predicat,3,arg_sursalaire);
1035    // Recuperation des resultats
        reduction_sursalaire = Rd_Integer(arg_sursalaire[2]);
        // Fin de la requete et liberation de la memoire de GNU Prolog
        Pl_Query_End(PL_RECOVER);

1040    // Impot du apres toutes les reductions
        int impot =0;
        int impot_homme = 0;
        int impot_femme = 0;
        // Cas d'une declaration non commune
1045    if(pas_imposition_commune == 1){
          impot = impot_revenus + impot_rente1 + impot_rente2 + impot_rente3 -
            impot_ME -
            reduction_epargne_logement -
            reduction_epargne_long_terme -
1050        reduction_ALE -
            reduction_titre_service -
            reduction_gagnant -
            reduction_gagnant_perdu -
            reduction_economie_energie -
1055        reduction_renovation -
            reduction_voiture_moins_105 -
            reduction_voiture_plus_105 -
            reduction_allocation_chomage -
            reduction_indemnite_maladie -
1060        reduction_pension -
            reduction_sursalaire;
        }
        else{
          impot_homme = impot_revenus_homme + impot_rente1 + impot_rente2 +
1065        impot_rente3 -
            impot_ME_homme -
            reduction_epargne_logement -
            reduction_epargne_long_terme -
            reduction_ALE -
1070        reduction_titre_service -
            reduction_gagnant -
            reduction_gagnant_perdu -
            reduction_economie_energie -
```

```
            reduction_renovation -
            reduction_voiture_moins_105 -
            reduction_voiture_plus_105 -
            reduction_allocation_chomage -
            reduction_indemnite_maladie -
            reduction_pension -
            reduction_sursalaire;
          impot_femme = impot_revenus_femme - impot_ME_femme;
        }

        // Calcul credit d'impot Arkimedes
        int credit_impot_arkimedes;
        // Recherche de la cle interne du predicat a utiliser
        predicat = Find_Atom("credit_impot_arkimedes");
        // Debut d'une requete
        Pl_Query_Begin(TRUE);
        // Preparation des arguments de la requête
        PlTerm arg_arkimedes[2];
        arg_arkimedes[0] = Mk_Integer((int)(arkimedes*100));
        arg_arkimedes[1] = Mk_Variable();
        // Appel de la requete
        Pl_Query_Call(predicat,2,arg_arkimedes);
        // Recuperation des resultats
        credit_impot_arkimedes = Rd_Integer(arg_arkimedes[1]);
        // Fin de la requete et liberation de la memoire de GNU Prolog
        Pl_Query_End(PL_RECOVER);


        // Calcul credit internet pour tous
        int credit_impot_internet;
        // Recherche de la cle interne du predicat a utiliser
        predicat = Find_Atom("credit_impot_internet");
        // Debut d'une requete
        Pl_Query_Begin(TRUE);
        // Preparation des arguments de la requête
        PlTerm arg_internet[2];
        arg_internet[0] = Mk_Integer((int)(internet*100));
        arg_internet[1] = Mk_Variable();
        // Appel de la requete
        Pl_Query_Call(predicat,2,arg_internet);
        // Recuperation des resultats
        credit_impot_internet = Rd_Integer(arg_internet[1]);
        // Fin de la requete et liberation de la memoire de GNU Prolog
```

```
        Pl_Query_End(PL_RECOVER);


1120    // Si versement anticipe, calcul bonification
        int bonification = 0;
        if(versement_anticipe > 0){
          // Recherche de la cle interne du predicat a utiliser
          predicat = Find_Atom("bonification");
1125      // Debut d'une requete
          Pl_Query_Begin(TRUE);
          // Preparation des arguments de la requête
          PlTerm arg_bonification[5];
          // Cas d'une imposition non commune
1130      if(pas_imposition_commune == 1){
            arg_bonification[0] = Mk_Integer(impot);
          }
          else{ // Cas d'une imposition commune
            arg_bonification[0] = Mk_Integer(impot_homme);
1135      }
          arg_bonification[1] = Mk_Integer(PI);
          arg_bonification[2] = Mk_Integer((int)(precompte_prof_total*100));
          arg_bonification[3] = Mk_Integer((int)(versement_anticipe*100));
          arg_bonification[4] = Mk_Variable();
1140      // Appel de la requete
          Pl_Query_Call(predicat,5,arg_bonification);
          // Recuperation des resultats
          bonification = Rd_Integer(arg_bonification[4]);
          // Fin de la requete et liberation de la memoire de GNU Prolog
1145      Pl_Query_End(PL_RECOVER);
        }


        // Calcul cotisation securite sociale
1150    int cotisation_securite_sociale;
        // Recherche de la cle interne du predicat a utiliser
        predicat = Find_Atom("calcul_cotisation_speciale");
        // Debut d'une requete
        Pl_Query_Begin(TRUE);
1155    // Preparation des arguments de la requête
        PlTerm arg_sec_soc[2];
        arg_sec_soc[0] = Mk_Integer((int)(imposable_taux_plein*100));
        arg_sec_soc[1] = Mk_Variable();
        // Appel de la requete
```

```
1160    Pl_Query_Call(predicat,2,arg_sec_soc);
        // Recuperation des resultats
        cotisation_securite_sociale = Rd_Integer(arg_sec_soc[1]);
        // Fin de la requete et liberation de la memoire de GNU Prolog
        Pl_Query_End(PL_RECOVER);
1165

        // Arret du moteur d'inference de GNU Prolog
        Stop_Prolog();

1170    /******************LISTING A RENVOYER AU SERVEUR****************/

        // En tete de l'output CGI
        printf("Content-type: text/html\n\n");

1175    // En tete HTML
        printf("<HTML>\n");
        printf("<HEAD>\n");
        printf("<TITLE>\n");
        printf("Système expert fiscal\n");
1180    printf("</TITLE>\n");
        printf("</HEAD>\n");

        printf("<BODY>\n");
        printf("<CENTER> <H2> R&eacute;sum&eacute; du calcul de l'imp&ocirc;t ");
1185    printf("du contribuable n&deg; %s </H2></CENTER>",numero_national);

        printf("<H2> D&eacute;termination du revenu imposable </H2>");

        // Revenus immobiliers
1190    printf("<H3>Revenus immobiliers :</H3>");
        printf("<TABLE>");
        printf("<TR>");
        printf("<TD>");
        printf("Revenus cadastraux soumis au pr&eacute;compte immobilier : ");
1195    printf("</TD>");
        printf("<TD align=\"right\">");
        printf(" %.2lf<BR>",(float)revenus_cadastral_homme);
        printf("</TD>");
        printf("</TR>");
1200
        printf("<TR>");
        printf("<TD>");
```

```
        printf("Autres revenus cadastraux : ");
        printf("</TD>");
1205    printf("<TD align=\"right\">");
        printf(" %.2lf<BR>",(float)revenus_cadastral_sans_precompte_h);
        printf("</TD>");
        printf("</TR>");

1210    printf("<TR>");
        printf("<TD>");
        printf("Revenus cadastraux  index&eacute; : ");
        printf("</TD>");
        printf("<TD align=\"right\">");
1215    printf(" %d<BR>",(int)((revenus_cadastral_homme +
               revenus_cadastral_sans_precompte_h)*1.4276));
        printf("</TD>");
        printf("</TR>");

1220    printf("<TR>");
        printf("<TD>");
        printf("Inter&ecirc;ts pay&eacute;s : ");
        printf("</TD>");
        printf("<TD align=\"right\">");
1225    printf(" - %.2lf<BR>",atof(getenv("WWW_v1370")));
        printf("</TD>");
        printf("</TR>");
        // revenus immobiliers
        float RI = ((int)((revenus_cadastral_homme +
1230                       revenus_cadastral_sans_precompte_h)*1.4276)) -
          atof(getenv("WWW_v1370"));
        printf("<TR>");
        printf("<TD>");
        printf("Revenus immobiliers : ");
1235    printf("</TD>");
        printf("<TD align=\"right\">");
        if(RI < 0){
          printf(" %.2lf => 0.00 <BR>",RI);
        }
1240    else{
          printf(" %.2lf<BR>",RI);
        }
        printf("</TD>");
        printf("</TR>");
1245
```

```
        printf("<TR>");
        printf("<TD>");
        printf("Pr&eacute;compte immobilier : ");
        printf("</TD>");
1250    printf("<TD align=\"right\">");
        printf(" %.2lf<BR>",(float)PI/100);
        printf("</TD>");
        printf("</TR>");
        printf("</TABLE>");
1255
        printf("<hr align=\"center\" width=\"90%\" color=\"black\" size=\"2\">");

        // Traitement et salaires
        printf("<H3> Traitements et salaires </H3>");
1260
        printf("<TABLE>");

        printf("<TR>");
        printf("<TD>");
1265    printf("R&eacute;mun&eacute;rations ordinaires : ");
        printf("</TD>");
        printf("<TD align=\"right\">");
        printf(" %.2lf<BR>",remunerations_ordinaires);
        printf("</TD>");
1270    printf("</TR>");

        printf("<TR>");
        printf("<TD>");
        printf("Frais de d&eacute;placement : ");
1275    printf("</TD>");
        printf("<TD align=\"right\">");
        printf(" %.2lf<BR>",frais_deplacement);
        printf("</TD>");
        printf("</TR>");
1280
        printf("<TR>");
        printf("<TD>");
        printf("Frais de d&eacute;placement exon&eacute;r&eacute;s : ");
        printf("</TD>");
1285    printf("<TD align=\"right\">");
        printf(" -%.2lf<BR>",frais_deplacement_exonere);
        printf("</TD>");
        printf("</TR>");
```

```
1290      printf("<TR>");
          printf("<TD>");
          printf("Allocations de ch&ocirc;mage : ");
          printf("</TD>");
          printf("<TD align=\"right\">");
1295      printf(" %.2lf<BR>",allocation_chomage);
          printf("</TD>");
          printf("</TR>");

          printf("<TR>");
1300      printf("<TD>");
          printf("Indemnit&eacute;s l&eacute;gales de maladie-invalidit&eacute; : ");
          printf("</TD>");
          printf("<TD align=\"right\">");
          printf(" %.2lf<BR>",indemnite_maladie);
1305      printf("</TD>");
          printf("</TR>");

          printf("<TR>");
          printf("<TD>");
1310      printf("Autres revenus de remplacement : ");
          printf("</TD>");
          printf("<TD align=\"right\">");
          printf(" %.2lf<BR>",revenus_remplacement);
          printf("</TD>");
1315      printf("</TR>");

          printf("<TR>");
          printf("<TD>");
          printf("Pr&eacute;pensions : ");
1320      printf("</TD>");
          printf("<TD align=\"right\">");
          printf(" %.2lf<BR>",prepension);
          printf("</TD>");
          printf("</TR>");
1325
          printf("<TR>");
          printf("<TD>");
          printf("Heures suppl&eacute;mentaires : ");
          printf("</TD>");
1330      printf("<TD align=\"right\">");
          printf(" %.2lf<BR>",  heure_sup);
```

```
        printf("</TD>");
        printf("</TR>");

1335    printf("<TR>");
        printf("<TD>");
        printf("Pensions : ");
        printf("</TD>");
        printf("<TD align=\"right\">");
1340    printf(" %.2lf<BR>",  pension);
        printf("</TD>");
        printf("</TR>");

        printf("<TR>");
1345    printf("<TD>");
        printf(" ");
        printf("</TD>");
        printf("<TD>");
        printf("<hr align=\"center\" width=\"90%\" color=\"black\" size=\"2\">");
1350    printf("</TD>");
        printf("</TR>");

        printf("<TR>");
        printf("<TD>");
1355    printf("Salaire imposable du d&eacute;clarant : ");
        printf("</TD>");
        printf("<TD align=\"right\">");
        printf(" %.2lf<BR>", salaire_imposable);
        printf("</TD>");
1360    printf("</TR>");

        printf("<TR>");
        printf("<TD>");
        printf("Frais professionnels forfaitaires : ");
1365    printf("</TD>");
        printf("<TD align=\"right\">");
        printf(" -%.2lf<BR>",  (float)FP/100);
        printf("</TD>");
        printf("</TR>");
1370
        if(forfait_long_deplacement > 0){
          printf("<TR>");
          printf("<TD>");
          printf("Forfait longs d&eacute;placements : ");
```

```
1375        printf("</TD>");
            printf("<TD align=\"right\">");
            printf(" -%.2lf<BR>", forfait_long_deplacement );
            printf("</TD>");
            printf("</TR>");
1380      }

          printf("<TR>");
          printf("<TD>");
          printf(" ");
1385      printf("</TD>");
          printf("<TD>");
          printf("<hr align=\"center\" width=\"90%\" color=\"black\" size=\"2\">");
          printf("</TD>");
          printf("</TR>");
1390
          printf("<TR>");
          printf("<TD>");
          printf("Imposable au taux plein : ");
          printf("</TD>");
1395      printf("<TD align=\"right\">");
          printf(" %.2lf<BR>",  imposable_taux_plein);
          printf("</TD>");
          printf("</TR>");

1400      printf("</TABLE>");

          printf("<hr align=\"center\" width=\"90%\" color=\"black\" size=\"2\">");

          // Cas ou il s'agit d'une imposition commune
1405      if(pas_imposition_commune != 1){
            printf("<H3> R&eacute;partition quotient conjugal </H3>");
            printf("<TABLE>");
            printf("<TR>");
            printf("<TD>");
1410        printf(" ");
            printf("</TD>");
            printf("<TD>");
            printf("<I>Homme</I>");
            printf("</TD>");
1415        printf("<TD>");
            printf("<I>Femme</I>");
            printf("</TD>");
```

```
        printf("</TR>");

        printf("<TR>");
        printf("<TD>");
        printf(" Montant initial : ");
        printf("</TD>");
        printf("<TD align=\"right\">");
        printf(" %.2lf<BR>", imposable_taux_plein);
        printf("</TD>");
        printf("<TD>");
        printf("");
        printf("</TD>");
        printf("</TR>");


        printf("<TR>");
        printf("<TD>");
        printf(" Quotient conjugal : ");
        printf("</TD>");
        printf("<TD align=\"right\">");
        printf(" -%.2lf<BR>", (float)QC/100);
        printf("</TD>");
        printf("<TD>");
        printf("");
        printf("</TD>");
        printf("</TR>");

        printf("<TR>");
        printf("<TD>");
        printf(" ");
        printf("</TD>");
        printf("<TD>");
        printf("<hr align=\"center\" width=\"90%\" color=\"black\" size=\"2\">");
        printf("</TD>");
        printf("<TD>");
        printf("<hr align=\"center\" width=\"90%\" color=\"black\" size=\"2\">");
        printf("</TD>");
        printf("</TR>");

        printf("<TR>");
        printf("<TD>");
        printf(" R&eacute;partition finale : ");
        printf("</TD>");
```

```
        printf("<TD align=\"right\">");
        printf(" %.2lf<BR>", revenu_homme_hors_deduction);
        printf("</TD>");
        printf("<TD right\"right\">");
1465    printf(" %.2lf<BR>", revenu_femme_hors_deduction);
        printf("</TD>");
        printf("</TR>");

        printf("</TABLE");
1470
        printf("<hr align=\"center\" width=\"90%\" color=\"black\" size=\"2\">");

        printf("<H3> El&eacute;ments d&eacute;ductibles </H3>");
        printf("<TABLE>");
1475
        printf("<TR>");
        printf("<TD right\"right\">");
        printf(" ");
        printf("</TD>");
1480    printf("<TD>");
        printf("</TD>");
        printf("<TD>");
        printf("<I>Homme</I>");
        printf("</TD>");
1485    printf("<TD>");
        printf("<I>Femme</I>");
        printf("</TD>");
        printf("</TR>");

1490    printf("<TR>");
        printf("<TD>");
        printf(" Revenus hors d&eacute;duction : ");
        printf("</TD>");
        printf("<TD>");
1495    printf("</TD>");
        printf("<TD right\"right\">");
        printf(" %.2lf<BR>", revenu_homme_hors_deduction);
        printf("</TD>");
        printf("<TD right\"right\">");
1500    printf(" %.2lf<BR>", revenu_femme_hors_deduction);
        printf("</TD>");
        printf("</TR>");
```

```
        printf("<TR>");
1505    printf("<TD>");
        printf("El&eacute;ments d&eacute;ductibles : ");
        printf("</TD>");
        printf("<TD right\"right\">");
        printf(" %.2lf<BR>",  elements_deductibles);
1510    printf("</TD>");
        printf("</TR>");

        printf("<TR>");
        printf("<TD>");
1515    printf("R&eacute;partition &eacute;l&eacute;ments d&eacute;ductibles : ");
        printf("</TD>");
        printf("<TD>");
        printf("</TD>");
        printf("<TD right\"right\">");
1520    printf(" -%.2lf <BR>",  elements_deductibles*0.7);
        printf("</TD>");
        printf("<TD align=\"right\">");
        printf(" -%.2lf <BR>",  elements_deductibles*0.3);
        printf("</TD>");
1525
        printf("<TR>");
        printf("<TD>");
        printf(" ");
        printf("</TD>");
1530    printf("<TD>");
        printf("</TD>");
        printf("<TD>");
        printf("<hr align=\"center\" width=\"90%\" color=\"black\" size=\"2\">");
        printf("</TD>");
1535    printf("<TD>");
        printf("<hr align=\"center\" width=\"90%\" color=\"black\" size=\"2\">");
        printf("</TD>");
        printf("</TR>");

1540    printf("<TR>");
        printf("<TD>");
        printf("Revenus imposables : ");
        printf("</TD>");
        printf("<TD>");
1545    printf("</TD>");
        printf("<TD align=\"right\">");
```

```
        printf(" %.2lf <BR>",  revenu_homme);
        printf("</TD>");
        printf("<TD align=\"right\">");
1550    printf(" %.2lf <BR>",  revenu_femme);
        printf("</TD>");

        printf("</TR>");

1555    printf("</TABLE>");

        printf("<hr align=\"center\" width=\"90%\" color=\"black\" size=\"2\">");

        // Partie calcul de l'impot
1560    printf("<H2> Calcul de l'imp&ocirc;t </H2>");

        //Montant exempté d'impot
        printf("<H3> Montant exempt&eacute; d'imp&ocirc;t </H3>");

1565    printf("<TABLE>");

        printf("<TR>");
        printf("<TD>");
        printf("");
1570    printf("</TD>");
        printf("<TD>");
        printf("<I> Homme </I>");
        printf("</TD>");
        printf("<TD>");
1575    printf("<I> Femme </I>");
        printf("</TD>");
        printf("</TR>");

        printf("<TR>");
1580    printf("<TD>");
        printf("Montant exempt&eacute; d'imp&ocirc;t pour tout contribuable :");
        printf("</TD>");
        printf("<TD align=\"right\">");
        printf("5940.00");
1585    printf("</TD>");
        printf("<TD align=\"right\">");
        printf("5940.00");
        printf("</TD>");
        printf("</TR>");
```

```
1590        if(bonus_handicape > 0){
              printf("<TR>");
              printf("<TD>");
              printf("Bonus personne handicap&eacute;e :");
              printf("</TD>");
1595          printf("<TD align=\"right\">");
              printf("%.2lf" , (float)bonus_handicape/100);
              printf("</TD>");
              printf("<TD align=\"right\">");
              printf("<CENTER> - </CENTER>");
1600          printf("</TD>");
              printf("</TR>");
            }
            if(bonus_marie > 0){
              printf("<TR>");
1605          printf("<TD>");
              printf("Bonus mari&eacute; en 2006 avec conjoint faibles revenus :");
              printf("</TD>");
              printf("<TD align=\"right\">");
              printf("%.2lf" , (float)bonus_marie/100);
1610          printf("</TD>");
              printf("<TD>");
              printf("<CENTER> - </CENTER>");
              printf("</TD>");
              printf("</TR>");
1615        }
            if(bonus_cohabitant > 0){
              printf("<TR>");
              printf("<TD>");
              printf("Bonus cohabitant l&eacute;gal en 2006 avec conjoint");
1620          printf(" faibles revenus :");
              printf("</TD>");
              printf("<TD align=\"right\">");
              printf("%.2lf" , (float)bonus_cohabitant/100);
              printf("</TD>");
1625          printf("<TD>");
              printf("<CENTER> - </CENTER>");
              printf("</TD>");
              printf("</TR>");
            }
1630        if(bonus_enfant > 0){
              printf("<TR>");
              printf("<TD>");
```

```
            printf("Bonus pour enfant(s) &agrave; charge :");
            printf("</TD>");
1635        printf("<TD align=\"right\">");
            printf("%.2lf" , (float)bonus_enfant/100);
            printf("</TD>");
            printf("<TD>");
            printf("<CENTER> - </CENTER>");
1640        printf("</TD>");

            printf("</TR>");
          }
          if(bonus_enfant_moitie > 0){
1645        printf("<TR>");
            printf("<TD>");
            printf("Moiti&eacute; du bonus pour enfant à charge :");
            printf("</TD>");
            printf("<TD align=\"right\">");
1650        printf("%.2lf" , (float)bonus_enfant_moitie/100);
            printf("</TD>");
            printf("<TD>");
            printf("<CENTER> - </CENTER>");
            printf("</TD>");
1655        printf("</TR>");
          }
          if(bonus_enfant_3ans > 0){
            printf("<TR>");
            printf("<TD>");
1660        printf("Bonus pour enfant &agrave; charge de moins de 3 ans :");
            printf("</TD>");
            printf("<TD align=\"right\">");
            printf("%.2lf" , (float)bonus_enfant_3ans/100);
            printf("</TD>");
1665        printf("<TD>");
            printf("<CENTER> - </CENTER>");
            printf("</TD>");
            printf("</TR>");
          }
1670      if(bonus_pers_charge > 0){
            printf("<TR>");
            printf("<TD>");
            printf("Bonus pour personne &agrave; charge de moins de 65 ans :");
            printf("</TD>");
1675        printf("<TD align=\"right\">");
```

```
            printf("%.2lf" , (float)bonus_pers_charge/100);
            printf("</TD>");
            printf("<TD>");
            printf("<CENTER> - </CENTER>");
            printf("</TD>");
            printf("</TR>");
          }
          if(bonus_pers_charge_65 > 0){
            printf("<TR>");
            printf("<TD>");
            printf("Bonus pour personne &agrave; charge de plus de 65 ans :");
            printf("</TD>");
            printf("<TD align=\"right\">");
            printf("%.2lf" , (float)bonus_pers_charge_65/100);
            printf("</TD>");
            printf("<TD>");
            printf("<CENTER> - </CENTER>");
            printf("</TD>");
            printf("</TR>");
          }

          printf("<TR>");
          printf("<TD>");
          printf(" ");
          printf("</TD>");
          printf("<TD>");
          printf("<hr align=\"center\" width=\"90%\" color=\"black\" size=\"2\">");
          printf("</TD>");
          printf("<TD>");
          printf("<hr align=\"center\" width=\"90%\" color=\"black\" size=\"2\">");
          printf("</TD>");
          printf("</TR>");

          printf("<TR>");
          printf("<TD>");
          printf("Montant total exempt&eacute; :");
          printf("</TD>");
          printf("<TD align=\"right\">");
          printf("%.2lf" , (float)ME/100);
          printf("</TD>");
          printf("<TD align=\"right\">");
          printf("5940.00");
          printf("</TD>");
```

```
        printf("</TR>");
1720    // Si la repartition change les montants, on affiche les nouveaux
        if(repartition_ME_homme != ME){
          printf("<TR>");
          printf("<TD>");
          printf("Montant  exempt&eacute; apr&egrave;s r&eacute;partition :");
1725      printf("</TD>");
          printf("<TD align=\"right\">");
          printf("%.2lf",(float)repartition_ME_homme/100);
          printf("</TD>");
          printf("<TD align=\"right\">");
1730      printf("%.2lf",(float)repartition_ME_femme/100);
          printf("</TD>");
          printf("</TR>");
        }
        printf("</TABLE>");
1735
        printf("<hr align=\"center\" width=\"90%\" color=\"black\" size=\"2\">");

        printf("<H3> Calcul de l'imp&ocirc;t </H3>");

1740    // Impots
        printf("<TABLE>");

        printf("<TR>");
        printf("<TD>");
1745    printf("");
        printf("</TD>");
        printf("<TD>");
        printf("<I> Homme </I>");
        printf("</TD>");
1750    printf("<TD>");
        printf("<I> Femme </I>");
        printf("</TD>");
        printf("<TD>");
        printf("</TD>");
1755    printf("</TR>");

        printf("<TR>");
        printf("<TD>");
        printf("Imp&ocirc;t sur les revenus :");
1760    printf("</TD>");
        printf("<TD align=\"right\">");
```

```
        printf("%.2lf",(float)impot_revenus_homme/100);
        printf("</TD>");
        printf("<TD align=\"right\">");
1765    printf("%.2lf",(float)impot_revenus_femme/100);
        printf("</TD>");
        printf("<TD>");
        printf("</TD>");
        printf("</TR>");
1770
        printf("<TR>");
        printf("<TD>");
        printf("Imp&ocirc;t sur les rentes alimentaires :");
        printf("</TD>");
1775    printf("<TD align=\"right\">");
        printf("%.2lf" , (float)(impot_rente1+impot_rente2+impot_rente3)/100);
        printf("</TD>");
        printf("<TD align=\"right\">");
        printf("<CENTER> - </CENTER>");
1780    printf("</TD>");
        printf("<TD>");
        printf("</TD>");
        printf("</TR>");

1785    printf("<TR>");
        printf("<TD>");
        printf("Imp&ocirc;t sur les montants exempt&eacute;s :");
        printf("</TD>");
        printf("<TD align=\"right\">");
1790    printf("-%.2lf",(float)impot_ME_homme/100);
        printf("</TD>");
        printf("<TD align=\"right\">");
        printf("-%.2lf",(float)impot_ME_femme/100);
        printf("</TD>");
1795    printf("<TD>");
        printf("</TD>");
        printf("</TR>");


1800    // Reduction epargne logement
        if(epargne_logement >0){
          printf("<TR>");
          printf("<TD>");
          printf("R&eacute;duction &eacute;pargne logement ");
```

```
1805          printf("</TD>");
              printf("<TD>");
              printf("</TD>");
              printf("<TD>");
              printf("</TD>");
1810          printf("<TD>");
              printf("</TD>");
              printf("</TR>");

              printf("<TR>");
1815          printf("<TD>");
              printf("Part &agrave; prendre en compte pour ");
              printf("l'&eacute;pargne-logement :");
              printf("</TD>");
              printf("<TD>");
1820          printf("</TD>");
              printf("<TD>");
              printf("</TD>");
              printf("<TD>");
              printf("%.2lf" , epargne_logement);
1825          printf("</TD>");
              printf("</TR>");

              printf("<TR>");
              printf("<TD>");
1830          printf("A limiter &agrave; :");
              printf("</TD>");
              printf("<TD>");
              printf("</TD>");
              printf("<TD>");
1835          printf("</TD>");
              printf("<TD>");
              printf("%.2lf => %.2lf" , (float)limite_epargne_logement/100,
                     (float)min_epargne_logement/100);
              printf("</TD>");
1840          printf("</TR>");
              printf("<TR>");
              printf("<TD>");
              printf("R&eacute;duction finale &eacute;pargne logement :");
              printf("</TD>");
1845          printf("<TD align=\"right\">");
              printf("-%.2lf" , (float)reduction_epargne_logement/100);
              printf("</TD>");
```

```
            printf("<TD align=\"right\">");
            printf("<CENTER> - </CENTER>");
            printf("</TD>");
            printf("</TR>");
          }

          // Reduction epargne long-terme
          if(epargne_long_terme > 0){
            printf("<TR>");
            printf("<TD>");
            printf("R&eacute;duction &eacute;pargne long-terme ");
            printf("</TD>");
            printf("<TD>");
            printf("</TD>");
            printf("</TR>");

            printf("<TR>");
            printf("<TD>");
            printf("Part &agrave; prendre en compte pour ");
            printf("l'&eacute;pargne long-terme :");
            printf("</TD>");
            printf("<TD>");
            printf("</TD>");
            printf("<TD>");
            printf("</TD>");
            printf("<TD>");
            printf("%.2lf" , epargne_long_terme);
            printf("</TD>");
            printf("</TR>");

            printf("<TR>");
            printf("<TD>");
            printf("R&eacute;duction finale &eacute;pargne long-terme :");
            printf("</TD>");
            printf("<TD align=\"right\">");
            printf("-%.2lf" , (float)reduction_epargne_long_terme/100);
            printf("</TD>");
            printf("<TD align=\"right\">");
            printf("<CENTER> - </CENTER>");
            printf("</TD>");
            printf("</TR>");
          }
```

```
        // Reduction ALE
        if(versement_ALE >0){
          printf("<TR>");
          printf("<TD>");
1895      printf("Versement ALE :");
          printf("</TD>");
          printf("<TD>");
          printf("</TD>");
          printf("<TD>");
1900      printf("</TD>");
          printf("<TD>");
          printf(" %.2lf", versement_ALE);
          printf("</TD>");
          printf("</TR>");
1905
          printf("<TR>");
          printf("<TD>");
          printf("R&eacute;duction pour versements ALE :");
          printf("</TD>");
1910      printf("<TD align=\"right\">");
          printf("-%.2lf" , (float)reduction_ALE/100);
          printf("</TD>");
          printf("<TD align=\"right\">");
          printf("<CENTER> - </CENTER>");
1915      printf("</TD>");
          printf("</TR>");
        }

        // Reduction titre service
1920    if(versement_titre_service > 0){
          printf("<TR>");
          printf("<TD>");
          printf("Versement titres-services :");
          printf("</TD>");
1925      printf("<TD>");
          printf("</TD>");
          printf("<TD>");
          printf("</TD>");
          printf("<TD>");
1930      printf(" %.2lf", versement_titre_service);
          printf("</TD>");
          printf("</TR>");
```

```
            printf("<TR>");
1935        printf("<TD>");
            printf("R&eacute;duction pour versements titres-services :");
            printf("</TD>");
            printf("<TD align=\"right\">");
            printf("-%.2lf" , (float)reduction_titre_service/100);
1940        printf("</TD>");
            printf("<TD align=\"right\">");
            printf("<CENTER> - </CENTER>");
            printf("</TD>");
            printf("</TR>");
1945      }

          // Reduction pret gagnant-gagnant
          if(pret_gagnant_gagnant > 0){
            printf("<TR>");
1950        printf("<TD>");
            printf("Pr&ecirc;t gagnant-gagnant :");
            printf("</TD>");
            printf("<TD>");
            printf("</TD>");
1955        printf("<TD>");
            printf("</TD>");
            printf("<TD>");
            printf(" %.2lf", pret_gagnant_gagnant);
            printf("</TD>");
1960        printf("</TR>");

            printf("<TR>");
            printf("<TD>");
            printf("R&eacute;duction pour pr&ecirc;t gagnant-gagnant  :");
1965        printf("</TD>");
            printf("<TD align=\"right\">");
            printf("-%.2lf" , (float)reduction_gagnant/100);
            printf("</TD>");
            printf("<TD align=\"right\">");
1970        printf("<CENTER> - </CENTER>");
            printf("</TD>");
            printf("</TR>");
          }

1975      // Reduction pret gagnant-gagnant perdu
          if(pret_gagnant_gagnant_perdu > 0){
```

```
            printf("<TR>");
            printf("<TD>");
            printf("Pr&ecirc;t gagnant-gagnant d&eacute;finitivement perdu :");
1980        printf("</TD>");
            printf("<TD>");
            printf("</TD>");
            printf("<TD>");
            printf("</TD>");
1985        printf("<TD>");
            printf(" %.2lf", pret_gagnant_gagnant_perdu);
            printf("</TD>");
            printf("</TR>");

1990        printf("<TR>");
            printf("<TD>");
            printf("R&eacute;duction pour pr&ecirc;t gagnant-gagnant ");
            printf("d&eacute;finitivement perdu :");
            printf("</TD>");
1995        printf("<TD align=\"right\">");
            printf("-%.2lf" , (float)reduction_gagnant_perdu/100);
            printf("</TD>");
            printf("<TD align=\"right\">");
            printf("<CENTER> - </CENTER>");
2000        printf("</TD>");
            printf("</TR>");
          }

          // Reduction pour travaux d'economie d'energie
2005      if(economie_energie > 0){
            printf("<TR>");
            printf("<TD>");
            printf("D&eacute;penses en vue d'&eacute;conomiser de ");
            printf("l'&eacute;nergie :");
2010        printf("</TD>");
            printf("<TD>");
            printf("</TD>");
            printf("<TD>");
            printf("</TD>");
2015        printf("<TD>");
            printf(" %.2lf", economie_energie);
            printf("</TD>");
            printf("</TR>");
```

```
2020          printf("<TR>");
              printf("<TD>");
              printf("R&eacute;duction pour des d&eacute;penses en vue ");
              printf("d'&eacute;conomiser de l'&eacute;nergie ");
              printf("</TD>");
2025          printf("<TD align=\"right\">");
              printf("-%.2lf" , (float)reduction_economie_energie/100);
              printf("</TD>");
              printf("<TD align=\"right\">");
              printf("<CENTER> - </CENTER>");
2030          printf("</TD>");
              printf("</TR>");
            }

            // Reduction pour des renovations dans une zone d'action positive
2035        if(renovation_zone_active > 0){
              printf("<TR>");
              printf("<TD>");
              printf("D&eacute;penses pour des r&eacute;novations dans une zone ");
              printf("d'action positive :");
2040          printf("</TD>");
              printf("<TD>");
              printf("</TD>");
              printf("<TD>");
              printf("<TD>");
2045          printf("</TD>");
              printf(" %.2lf", renovation_zone_active);
              printf("</TD>");
              printf("</TR>");

2050          printf("<TR>");
              printf("<TD>");
              printf("R&eacute;duction pour des r&eacute;novations dans une zone ");
              printf("d'action positive : ");
              printf("</TD>");
2055          printf("<TD align=\"right\">");
              printf("-%.2lf" , (float)reduction_renovation/100);
              printf("</TD>");
              printf("<TD align=\"right\">");
              printf("<CENTER> - </CENTER>");
2060          printf("</TD>");
              printf("</TR>");
            }
```

```
        // Reduction pour l'achat d'une voiture qui emet mois de 105 gr
2065    if(voiture_moins_105 > 0){
          printf("<TR>");
          printf("<TD>");
          printf("D&eacute;penses pour une voiture qui emet - ");
          printf("de 105 gr. de CO2 : ");
2070      printf("</TD>");
          printf("<TD>");
          printf("</TD>");
          printf("<TD>");
          printf("</TD>");
2075      printf("<TD>");
          printf(" %.2lf", voiture_moins_105);
          printf("</TD>");
          printf("</TR>");

2080      printf("<TR>");
          printf("<TD>");
          printf("R&eacute;duction pour des d&eacute;penses pour une voiture ");
          printf("qui emet - de 105 gr. de CO2 : ");
          printf("</TD>");
2085      printf("<TD align=\"right\">");
          printf("-%.2lf" , (float)reduction_voiture_moins_105/100);
          printf("</TD>");
          printf("<TD align=\"right\">");
          printf("<CENTER> - </CENTER>");
2090      printf("</TD>");
          printf("</TR>");
        }

        // Reduction pour l'achat d'une voiture qui emet mois de 105 gr
2095    if(voiture_plus_105 > 0){
          printf("<TR>");
          printf("<TD>");
          printf("D&eacute;penses pour une voiture qui emet entre 105 et 115 ");
          printf("gr. de CO2 : ");
2100      printf("</TD>");
          printf("<TD>");
          printf("</TD>");
          printf("<TD>");
          printf("</TD>");
2105      printf("<TD>");
```

```
            printf(" %.2lf", voiture_plus_105);
            printf("</TD>");
            printf("</TR>");

            printf("<TR>");
            printf("<TD>");
            printf("R&eacute;duction pour des d&eacute;penses pour une voiture ");
            printf("qui emet entre 105 et 115 gr. de CO2 : ");
            printf("</TD>");
            printf("<TD align=\"right\">");
            printf("-%.2lf" , (float)reduction_voiture_plus_105/100);
            printf("</TD>");
            printf("<TD align=\"right\">");
            printf("<CENTER> - </CENTER>");
            printf("</TD>");
            printf("</TR>");
          }

          // Reduction pour allocations de chomage
          if(allocation_chomage > 0){
            printf("<TR>");
            printf("<TD>");
            printf("Allocation de ch&ocirc;mage : ");
            printf("</TD>");
            printf("<TD>");
            printf("</TD>");
            printf("<TD>");
            printf("</TD>");
            printf("<TD>");
            printf(" %.2lf", allocation_chomage);
            printf("</TD>");
            printf("</TR>");

            printf("<TR>");
            printf("<TD>");
            printf("R&eacute;duction pour allocation de ch&ocirc;mage :  ");
            printf("</TD>");
            printf("<TD align=\"right\">");
            printf("-%.2lf" , (float)reduction_allocation_chomage/100);
            printf("</TD>");
            printf("<TD align=\"right\">");
            printf("<CENTER> - </CENTER>");
            printf("</TD>");
```

```
        printf("</TR>");
2150    }

        // Reduction pour indemnites maladie-invalidite
        if(indemnite_maladie > 0){
          printf("<TR>");
2155      printf("<TD>");
          printf("Indemnit&eacute;s maladie-invalidit&eacute; : ");
          printf("</TD>");
          printf("<TD>");
          printf("</TD>");
2160      printf("<TD>");
          printf("</TD>");
          printf("<TD>");
          printf(" %.2lf", indemnite_maladie);
          printf("</TD>");
2165      printf("</TR>");

          printf("<TR>");
          printf("<TD>");
          printf("R&eacute;duction indemnit&eacute;s maladie-invalidit&eacute; : ");
2170      printf("</TD>");
          printf("<TD align=\"right\">");
          printf("-%.2lf" , (float)reduction_indemnite_maladie/100);
          printf("</TD>");
          printf("<TD align=\"right\">");
2175      printf("<CENTER> - </CENTER>");
          printf("</TD>");
          printf("</TR>");
        }

2180    // Reduction pour pensions, prepensions et autres revenus de remplacement
        if(pension_prepension_autre > 0){
          printf("<TR>");
          printf("<TD>");
          printf("Pensions, pr&eacute;pensions et autres revenus ");
2185      printf("de remplacement : ");
          printf("</TD>");
          printf("<TD>");
          printf("</TD>");
          printf("<TD>");
2190      printf("</TD>");
          printf("<TD>");
```

```
            printf(" %.2lf", pension_prepension_autre);
            printf("</TD>");
            printf("</TR>");

            printf("<TR>");
            printf("<TD>");
            printf("R&eacute;duction pour pensions, pr&eacute;pensions et ");
            printf("autres revenus de remplacement  :  ");
            printf("</TD>");
            printf("<TD align=\"right\">");
            printf("-%.2lf" , (float)reduction_pension/100);
            printf("</TD>");
            printf("<TD align=\"right\">");
            printf("<CENTER> - </CENTER>");
            printf("</TD>");
            printf("</TR>");
          }

          // Reduction pour sursalaire suite a des heures supplementaires de travail
          if(heure_sup > 0){
            printf("<TR>");
            printf("<TD>");
            printf("Sursalaire suite &agrave; des heures suppl&eacute;mentaires ");
            printf("de travail : ");
            printf("</TD>");
            printf("<TD>");
            printf("</TD>");
            printf("<TD>");
            printf("</TD>");
            printf("<TD>");
            printf(" %.2lf", heure_sup);
            printf("</TD>");
            printf("</TR>");

            printf("<TR>");
            printf("<TD>");
            printf("R&eacute;duction pour sursalaire suite &agrave; des  ");
            printf("heures suppl&eacute;mentaires de travail");
            printf("</TD>");
            printf("<TD align=\"right\">");
            printf("-%.2lf" , (float)reduction_sursalaire/100);
            printf("</TD>");
            printf("<TD align=\"right\">");
```

```
2235          printf("<CENTER> - </CENTER>");
              printf("</TD>");
              printf("</TR>");
            }

2240        printf("<TR>");
            printf("<TD>");
            printf(" ");
            printf("</TD>");
            printf("<TD>");
2245        printf("<hr align=\"center\" width=\"90%\" color=\"black\" size=\"2\">");
            printf("</TD>");
            printf("<TD>");
            printf("<hr align=\"center\" width=\"90%\" color=\"black\" size=\"2\">");
            printf("</TD>");
2250        printf("</TR>");

            // Impot final
            printf("<TR>");
            printf("<TD>");
2255        printf("Imp&ocirc;t : ");
            printf("</TD>");
            printf("<TD align=\"right\">");
            printf("%.2lf" , (float)impot_homme/100);
            printf("</TD>");
2260        printf("<TD align=\"right\">");
            printf("%.2lf" , (float)impot_femme/100);
            printf("</TD>");
            printf("</TR>");

2265        printf("</TABLE>");

            printf("<hr align=\"center\" width=\"90%\" color=\"black\" size=\"2\">");

            // Calcul final
2270        printf("<H2> Calcul final </H2>");

            printf("<TABLE>");

            printf("<TR>");
2275        printf("<TD>");
            printf("");
            printf("</TD>");
```

```
        printf("<TD>");
        printf("<I> Homme </I>");
2280    printf("</TD>");
        printf("<TD>");
        printf("<I> Femme </I>");
        printf("</TD>");
        printf("<TD>");
2285    printf("</TD>");
        printf("</TR>");

        printf("<TR>");
        printf("<TD>");
2290    printf("Imp&ocirc;t : ");
        printf("</TD>");
        printf("<TD align=\"right\">");
        printf("%.2lf" , (float)impot_homme/100);
        printf("</TD>");
2295    printf("</TD>");
        printf("<TD align=\"right\">");
        printf("%.2lf" , (float)impot_femme/100);
        printf("</TD>");
        printf("</TR>");
2300
        printf("<TR>");
        printf("<TD>");
        printf("Centimes additionnels d'agglom&eacute;ration (%.2lf%): ",
               taux_commune);
2305    printf("</TD>");
        printf("<TD align=\"right\">");
        printf("%.2lf" , ((float)impot_homme/100)*taux_commune/100);
        printf("</TD>");
        printf("<TD align=\"right\">");
2310    printf("<CENTER> - </CENTER>");
        printf("</TD>");
        printf("</TR>");


2315    printf("<TR>");
        printf("<TD>");
        printf("Pr&eacute;compte immobilier (non remboursable) : ");
        printf("</TD>");
        printf("<TD align=\"right\">");
2320    if(impot_homme > 0){
```

```
        printf("- %.2lf" , (float)PI/100);
      }
      else{ // L'impot est negatif, dans ce cas on ne rembourse pas le prec. imm.
        printf("- 0.00");
        PI = 0;
      }
      printf("</TD>");
      printf("<TD align=\"right\">");
      printf("<CENTER> - </CENTER>");
      printf("</TD>");
      printf("</TR>");


      if(credit_impot_arkimedes > 0){
        printf("<TR>");
        printf("<TD>");
        printf("Cr&eacute;dit d'imp&ocirc;t Arkimedes (non remboursable) : ");
        printf("</TD>");
        printf("<TD align=\"right\">");
        if(impot_homme > 0){
          printf("- %.2lf" , (float)credit_impot_arkimedes/100);
        }
        else{ // L'impot est negatif, dans ce cas on ne rembourse pas ce credit
          printf("- 0.00");
          credit_impot_arkimedes = 0;
        }
        printf("</TD>");
        printf("<TD align=\"right\">");
        printf("<CENTER> - </CENTER>");
        printf("</TD>");
        printf("</TR>");
      }

      printf("<TR>");
      printf("<TD>");
      printf("Pr&eacute;compte professionnel : ");
      printf("</TD>");
      printf("<TD align=\"right\">");
      printf("- %.2lf",precompte_prof_total);
      printf("</TD>");
      printf("<TD align=\"right\">");
      printf("<CENTER> - </CENTER>");
      printf("</TD>");
```

```
        printf("</TR>");
2365
        if(credit_impot_internet > 0){
          printf("<TR>");
          printf("<TD>");
          printf("Cr&eacute;dit d'imp&ocirc;t internet pour tous : ");
2370      printf("</TD>");
          printf("<TD align=\"right\">");
          printf("- %.2lf",(float)credit_impot_internet/100);
          printf("</TD>");
          printf("<TD align=\"right\">");
2375      printf("<CENTER> - </CENTER>");
          printf("</TD>");
          printf("</TR>");
        }

2380    if(versement_anticipe > 0){
          printf("<TR>");
          printf("<TD>");
          printf("Versements anticip&eacute;s : ");
          printf("</TD>");
2385      printf("<TD align=\"right\">");
          printf("- %.2lf",versement_anticipe);
          printf("</TD>");
          printf("<TD align=\"right\">");
          printf("<CENTER> - </CENTER>");
2390      printf("</TD>");
          printf("</TR>");
          if(bonification > 0){
            printf("<TR>");
            printf("<TD>");
2395        printf("Bonification li&eacute;es aux versements anticip&eacute;s : ");
            printf("</TD>");
            printf("<TD align=\"right\">");
            printf("- %.2lf",(float)bonification/100);
            printf("</TD>");
2400        printf("<TD align=\"right\">");
            printf("<CENTER> - </CENTER>");
            printf("</TD>");
            printf("</TR>");
          }
2405    }
```

```
        printf("<TR>");
        printf("<TD>");
        printf("Cotisation sp&eacute;ciale s&eacute;curit&eacute sociale : ");
2410    printf("</TD>");
        printf("<TD align=\"right\">");
        printf(" %.2lf",(float)cotisation_securite_sociale/100);
        printf("</TD>");
        printf("<TD align=\"right\">");
2415    printf("<CENTER> - </CENTER>");
        printf("</TD>");
        printf("</TR>");

        if(retenue_sec_sociale > 0 ){
2420      printf("<TR>");
          printf("<TD>");
          printf("D&eacute;j&agrave; retenu sur la cotisation sp&eacute;ciale ");
          printf("s&eacute;curit&eacute sociale : ");
          printf("</TD>");
2425      printf("<TD align=\"right\">");
          printf(" - %.2lf",retenue_sec_sociale);
          printf("</TD>");
          printf("<TD align=\"right\">");
          printf("<CENTER> - </CENTER>");
2430      printf("</TD>");
          printf("</TR>");
        }

        printf("<TR>");
2435    printf("<TD>");
        printf(" ");
        printf("</TD>");
        printf("<TD>");
        printf("<hr align=\"center\" width=\"90%\" color=\"black\" size=\"2\">");
2440    printf("</TD>");
        printf("<TD>");
        printf("<hr align=\"center\" width=\"90%\" color=\"black\" size=\"2\">");
        printf("</TD>");
        printf("</TR>");
2445
        int impot_final_homme = impot_homme +
          impot_homme*7/100 -  // centimes additionnels
          PI - // Precompte immobiliers
          credit_impot_arkimedes -
```

```
2450          (int)(precompte_prof_total*100) -
              credit_impot_internet -
              (int)(versement_anticipe*100) -
              bonification +
              cotisation_securite_sociale -
2455          (int)(retenue_sec_sociale*100);
            int impot_final_femme = impot_femme;

            printf("<TR>");
            printf("<TD>");
2460        printf("Solde : ");
            printf("</TD>");
            printf("<TD align=\"right\">");
            printf(" %.2lf ",(float)impot_final_homme/100);
            printf("</TD>");
2465        printf("<TD align=\"right\">");
            printf(" %.2lf ",(float)impot_final_femme/100);
            printf("</TD>");
            printf("</TR>");

2470        int impot_final_2_conjoints = impot_final_homme + impot_final_femme;

            // Cas ou l'impot est du
            if(impot_final_2_conjoints > 0){
              printf("<TR>");
2475          printf("<TD>");
              printf("Solde &agrave; <B> payer par les contribuables </B>: ");
              printf("</TD>");
              printf("<TD COLSPAN=\"2\" align=\"center\">");
              printf("<B> %.2lf </B>",(float)impot_final_2_conjoints/100);
2480          printf("</TD>");
              printf("</TR>");
            }
            else{ // Cas ou on doit rembourser de l'argent
              printf("<TR>");
2485          printf("<TD>");
              printf("Solde &agrave; <B> rembourser </B> aux contribuables: ");
              printf("</TD>");
              printf("<TD COLSPAN=\"2\" align=\"center\">");
              printf("<B> %.2lf </B>",-(float)impot_final_2_conjoints/100);
2490          printf("</TD>");
              printf("</TR>");
            }
```

```
        printf("</TABLE>");
2495


      }
      else{ // Cas d'une declaration non commune
2500    printf("<H3> El&eacute;ments d&eacute;ductibles </H3>");
        printf("<TABLE>");
        // Elements deductibles
        printf("<TR>");
        printf("<TD>");
2505    printf("El&eacute;ments d&eacute;ductibles : ");
        printf("</TD>");
        printf("<TD align=\"right\">");
        printf(" %.2lf<BR>",  elements_deductibles);
        printf("</TD>");
2510    printf("</TR>");

        printf("<TR>");
        printf("<TD>");
        printf("Revenu imposable : ");
2515    printf("</TD>");
        printf("<TD align=\"right\">");
        printf(" %.2lf - %.2lf = %.2lf <BR>",  imposable_taux_plein,
               elements_deductibles,
               revenu);
2520    printf("</TD>");
        printf("</TR>");

        printf("</TABLE>");

2525    printf("<hr align=\"center\" width=\"90%\" color=\"black\" size=\"2\">");

        // Partie calcul de l'impot
        printf("<H2> Calcul de l'imp&ocirc;t </H2>");

2530    //Montant exempté d'impot
        printf("<H3> Montant exempt&eacute; d'imp&ocirc;t </H3>");

        printf("<TABLE>");

2535    printf("<TR>");
```

```
        printf("<TD>");
        printf("Montant exempt&eacute; d'imp&ocirc;t pour tout contribuable :");
        printf("</TD>");
        printf("<TD align=\"right\">");
2540    printf("5940.00");
        printf("</TD>");
        printf("</TR>");
        if(bonus_handicape > 0){
          printf("<TR>");
2545      printf("<TD>");
          printf("Bonus personne handicap&eacute;e :");
          printf("</TD>");
          printf("<TD align=\"right\">");
          printf("%.2lf" , (float)bonus_handicape/100);
2550      printf("</TD>");
          printf("</TR>");
        }
        if(bonus_isole > 0){
          printf("<TR>");
2555      printf("<TD>");
          printf("Bonus isol&eacute; ayant un ou plusieurs");
          printf(" enfant(s) &agrave; charge :");
          printf("</TD>");
          printf("<TD align=\"right\">");
2560      printf("%.2lf" , (float)bonus_isole/100);
          printf("</TD>");
          printf("</TR>");
        }
        if(bonus_marie > 0){
2565      printf("<TR>");
          printf("<TD>");
          printf("Bonus mari&eacute; en 2006 avec conjoint faibles revenus :");
          printf("</TD>");
          printf("<TD align=\"right\">");
2570      printf("%.2lf" , (float)bonus_marie/100);
          printf("</TD>");
          printf("</TR>");
        }
        if(bonus_cohabitant > 0){
2575      printf("<TR>");
          printf("<TD>");
          printf("Bonus cohabitant l&eacute;gal en 2006 avec conjoint");
          printf(" faibles revenus :");
```

```
          printf("</TD>");
2580      printf("<TD align=\"right\">");
          printf("%.2lf" , (float)bonus_cohabitant/100);
          printf("</TD>");
          printf("</TR>");
        }
2585    if(bonus_enfant > 0){
          printf("<TR>");
          printf("<TD>");
          printf("Bonus pour enfant(s) &agrave; charge :");
          printf("</TD>");
2590      printf("<TD align=\"right\">");
          printf("%.2lf" , (float)bonus_enfant/100);
          printf("</TD>");
          printf("</TR>");
        }
2595    if(bonus_enfant_moitie > 0){
          printf("<TR>");
          printf("<TD>");
          printf("Moiti&eacute; du bonus pour enfant à charge :");
          printf("</TD>");
2600      printf("<TD align=\"right\">");
          printf("%.2lf" , (float)bonus_enfant_moitie/100);
          printf("</TD>");
          printf("</TR>");
        }
2605    if(bonus_enfant_3ans > 0){
          printf("<TR>");
          printf("<TD>");
          printf("Bonus pour enfant &agrave; charge de moins de 3 ans :");
          printf("</TD>");
2610      printf("<TD align=\"right\">");
          printf("%.2lf" , (float)bonus_enfant_3ans/100);
          printf("</TD>");
          printf("</TR>");
        }
2615    if(bonus_pers_charge > 0){
          printf("<TR>");
          printf("<TD>");
          printf("Bonus pour personne &agrave; charge de moins de 65 ans :");
          printf("</TD>");
2620      printf("<TD align=\"right\">");
          printf("%.2lf" , (float)bonus_pers_charge/100);
```

```
        printf("</TD>");
        printf("</TR>");
      }
2625  if(bonus_pers_charge_65 > 0){
        printf("<TR>");
        printf("<TD>");
        printf("Bonus pour personne &agrave; charge de plus de 65 ans :");
        printf("</TD>");
2630    printf("<TD align=\"right\">");
        printf("%.2lf" , (float)bonus_pers_charge_65/100);
        printf("</TD>");
        printf("</TR>");
      }
2635
      printf("<TR>");
      printf("<TD>");
      printf("");
      printf("</TD>");
2640  printf("<TD>");
      printf("<hr align=\"center\" width=\"90%\" color=\"black\" size=\"2\">");
      printf("</TD>");
      printf("</TR>");

2645  printf("<TR>");
      printf("<TD>");
      printf("Montant total exempt&eacute; :");
      printf("</TD>");
      printf("<TD align=\"right\">");
2650  printf("%.2lf" , (float)ME/100);
      printf("</TD>");
      printf("</TR>");

      printf("</TABLE>");
2655
      printf("<hr align=\"center\" width=\"90%\" color=\"black\" size=\"2\">");

      printf("<H3> Calcul de l'imp&ocirc;t </H3>");

2660  // Impots
      printf("<TABLE>");

      printf("<TR>");
      printf("<TD>");
```

```
2665        printf("Imp&ocirc;t sur les revenus :");
            printf("</TD>");
            printf("<TD align=\"right\">");
            printf("%.2lf" , (float)impot_revenus/100);
            printf("</TD>");
2670        printf("</TR>");

            if(impot_rente1+impot_rente2+impot_rente3>0){
              printf("<TR>");
              printf("<TD>");
2675          printf("Imp&ocirc;t sur les rentes alimentaires :");
              printf("</TD>");
              printf("<TD align=\"right\">");
              printf("%.2lf" , (float)(impot_rente1+impot_rente2+impot_rente3)/100);
              printf("</TD>");
2680          printf("</TR>");
            }
            // Impot sur montant exempte
            printf("<TR>");
            printf("<TD>");
2685        printf("Imp&ocirc;t sur le montant exempt&eacute; :");
            printf("</TD>");
            printf("<TD align=\"right\">");
            printf("-%.2lf" , (float)impot_ME/100);
            printf("</TD>");
2690        printf("</TR>");

            // Reduction epargne logement
            if(epargne_logement >0){
              printf("<TR>");
2695          printf("<TD>");
              printf("R&eacute;duction &eacute;pargne logement ");
              printf("</TD>");
              printf("<TD>");
              printf("</TD>");
2700          printf("</TR>");

              printf("<TR>");
              printf("<TD>");
              printf("Part &agrave; prendre en compte pour ");
2705          printf("l'&eacute;pargne-logement :");
              printf("</TD>");
              printf("<TD>");
```

```
            printf("</TD>");
            printf("<TD align=\"right\">");
            printf("%.2lf" , epargne_logement);
            printf("</TD>");
            printf("</TR>");

            printf("<TR>");
            printf("<TD>");
            printf("A limiter &agrave; :");
            printf("</TD>");
            printf("<TD>");
            printf("</TD>");
            printf("<TD align=\"right\">");
            printf("%.2lf => %.2lf" , (float)limite_epargne_logement/100,
                   (float)min_epargne_logement/100);
            printf("</TD>");
            printf("</TR>");

            printf("<TR>");
            printf("<TD>");
            printf("R&eacute;duction finale &eacute;pargne logement :");
            printf("</TD>");
            printf("<TD align=\"right\">");
            printf("-%.2lf" , (float)reduction_epargne_logement/100);
            printf("</TD>");
            printf("</TR>");
          }

          // Reduction epargne long-terme
          if(epargne_long_terme > 0){
            printf("<TR>");
            printf("<TD>");
            printf("R&eacute;duction &eacute;pargne long-terme ");
            printf("</TD>");
            printf("<TD>");
            printf("</TD>");
            printf("</TR>");

            printf("<TR>");
            printf("<TD>");
            printf("Part &agrave; prendre en compte pour ");
            printf("l'&eacute;pargne long-terme :");
            printf("</TD>");
```

```
            printf("<TD>");
            printf("</TD>");
            printf("<TD align=\"right\">");
            printf("%.2lf" , epargne_long_terme);
            printf("</TD>");
            printf("</TR>");

            printf("<TR>");
            printf("<TD>");
            printf("R&eacute;duction finale &eacute;pargne long-terme :");
            printf("</TD>");
            printf("<TD align=\"right\">");
            printf("-%.2lf" , (float)reduction_epargne_long_terme/100);
            printf("</TD>");
            printf("</TR>");
          }

          // Reduction ALE
          if(versement_ALE >0){
            printf("<TR>");
            printf("<TD>");
            printf("Versement ALE :");
            printf("</TD>");
            printf("<TD>");
            printf("</TD>");
            printf("<TD align=\"right\">");
            printf(" %.2lf", versement_ALE);
            printf("</TD>");
            printf("</TR>");

            printf("<TR>");
            printf("<TD>");
            printf("R&eacute;duction pour versements ALE :");
            printf("</TD>");
            printf("<TD align=\"right\">");
            printf("-%.2lf" , (float)reduction_ALE/100);
            printf("</TD>");
            printf("</TR>");
          }

          // Reduction titre service
          if(versement_titre_service > 0){
            printf("<TR>");
```

```
          printf("<TD>");
2795      printf("Versement titres-services :");
          printf("</TD>");
          printf("<TD>");
          printf("</TD>");
          printf("<TD align=\"right\">");
2800      printf(" %.2lf", versement_titre_service);
          printf("</TD>");
          printf("</TR>");

          printf("<TR>");
2805      printf("<TD>");
          printf("R&eacute;duction pour versements titres-services :");
          printf("</TD>");
          printf("<TD align=\"right\">");
          printf("-%.2lf" , (float)reduction_titre_service/100);
2810      printf("</TD>");
          printf("</TR>");
        }

        // Reduction pret gagnant-gagnant
2815    if(pret_gagnant_gagnant > 0){
          printf("<TR>");
          printf("<TD>");
          printf("Pr&ecirc;t gagnant-gagnant :");
          printf("</TD>");
2820      printf("<TD>");
          printf("</TD>");
          printf("<TD align=\"right\">");
          printf(" %.2lf", pret_gagnant_gagnant);
          printf("</TD>");
2825      printf("</TR>");

          printf("<TR>");
          printf("<TD>");
          printf("R&eacute;duction pour pr&ecirc;t gagnant-gagnant  :");
2830      printf("</TD>");
          printf("<TD align=\"right\">");
          printf("-%.2lf" , (float)reduction_gagnant/100);
          printf("</TD>");
          printf("</TR>");
2835    }
```

```
        // Reduction pret gagnant-gagnant perdu
        if(pret_gagnant_gagnant_perdu > 0){
          printf("<TR>");
2840      printf("<TD>");
          printf("Pr&ecirc;t gagnant-gagnant d&eacute;finitivement perdu :");
          printf("</TD>");
          printf("<TD>");
          printf("</TD>");
2845      printf("<TD align=\"right\">");
          printf(" %.2lf", pret_gagnant_gagnant_perdu);
          printf("</TD>");
          printf("</TR>");

2850      printf("<TR>");
          printf("<TD>");
          printf("R&eacute;duction pour pr&ecirc;t gagnant-gagnant ");
          printf("d&eacute;finitivement perdu :");
          printf("</TD>");
2855      printf("<TD align=\"right\">");
          printf("-%.2lf" , (float)reduction_gagnant_perdu/100);
          printf("</TD>");
          printf("</TR>");
        }
2860
        // Reduction pour travaux d'economie d'energie
        if(economie_energie > 0){
          printf("<TR>");
          printf("<TD>");
2865      printf("D&eacute;penses en vue d'&eacute;conomiser de ");
          printf("l'&eacute;nergie :");
          printf("</TD>");
          printf("<TD>");
          printf("</TD>");
2870      printf("<TD align=\"right\">");
          printf(" %.2lf", economie_energie);
          printf("</TD>");
          printf("</TR>");

2875      printf("<TR>");
          printf("<TD>");
          printf("R&eacute;duction pour des d&eacute;penses en vue ");
          printf("d'&eacute;conomiser de l'&eacute;nergie ");
          printf("</TD>");
```

```
2880          printf("<TD align=\"right\">");
              printf("-%.2lf" , (float)reduction_economie_energie/100);
              printf("</TD>");
              printf("</TR>");
            }
2885
            // Reduction pour des renovations dans une zone d'action positive
            if(renovation_zone_active > 0){
              printf("<TR>");
              printf("<TD>");
2890          printf("D&eacute;penses pour des r&eacute;novations dans une zone ");
              printf("d'action positive :");
              printf("</TD>");
              printf("<TD>");
              printf("</TD>");
2895          printf("<TD align=\"right\">");
              printf(" %.2lf", renovation_zone_active);
              printf("</TD>");
              printf("</TR>");

2900          printf("<TR>");
              printf("<TD>");
              printf("R&eacute;duction pour des r&eacute;novations dans une zone ");
              printf("d'action positive : ");
              printf("</TD>");
2905          printf("<TD align=\"right\">");
              printf("-%.2lf" , (float)reduction_renovation/100);
              printf("</TD>");
              printf("</TR>");
            }
2910
            // Reduction pour l'achat d'une voiture qui emet mois de 105 gr
            if(voiture_moins_105 > 0){
              printf("<TR>");
              printf("<TD>");
2915          printf("D&eacute;penses pour une voiture qui emet - ");
              printf("de 105 gr. de CO2 : ");
              printf("</TD>");
              printf("<TD>");
              printf("</TD>");
2920          printf("<TD align=\"right\">");
              printf(" %.2lf", voiture_moins_105);
              printf("</TD>");
```

```
            printf("</TR>");

2925        printf("<TR>");
            printf("<TD>");
            printf("R&eacute;duction pour des d&eacute;penses pour une voiture ");
            printf("qui emet - de 105 gr. de CO2 : ");
            printf("</TD>");
2930        printf("<TD align=\"right\">");
            printf("-%.2lf" , (float)reduction_voiture_moins_105/100);
            printf("</TD>");
            printf("</TR>");
          }
2935
          // Reduction pour l'achat d'une voiture qui emet mois de 105 gr
          if(voiture_plus_105 > 0){
            printf("<TR>");
            printf("<TD>");
2940        printf("D&eacute;penses pour une voiture qui emet entre 105 et 115 ");
            printf("gr. de CO2 : ");
            printf("</TD>");
            printf("<TD>");
            printf("</TD>");
2945        printf("<TD align=\"right\">");
            printf(" %.2lf", voiture_plus_105);
            printf("</TD>");
            printf("</TR>");

2950        printf("<TR>");
            printf("<TD>");
            printf("R&eacute;duction pour des d&eacute;penses pour une voiture ");
            printf("qui emet entre 105 et 115 gr. de CO2 : ");
            printf("</TD>");
2955        printf("<TD align=\"right\">");
            printf("-%.2lf" , (float)reduction_voiture_plus_105/100);
            printf("</TD>");
            printf("</TR>");
          }
2960
          // Reduction pour allocations de chomage
          if(allocation_chomage > 0){
            printf("<TR>");
            printf("<TD>");
2965        printf("Allocation de ch&ocirc;mage : ");
```

```
            printf("</TD>");
            printf("<TD>");
            printf("</TD>");
            printf("<TD align=\"right\">");
            printf(" %.2lf", allocation_chomage);
            printf("</TD>");
            printf("</TR>");

            printf("<TR>");
            printf("<TD>");
            printf("R&eacute;duction pour allocation de ch&ocirc;mage :  ");
            printf("</TD>");
            printf("<TD align=\"right\">");
            printf("-%.2lf" , (float)reduction_allocation_chomage/100);
            printf("</TD>");
            printf("</TR>");
          }

          // Reduction pour indemnites maladie-invalidite
          if(indemnite_maladie > 0){
            printf("<TR>");
            printf("<TD>");
            printf("Indemnit&eacute;s maladie-invalidit&eacute; : ");
            printf("</TD>");
            printf("<TD>");
            printf("</TD>");
            printf("<TD align=\"right\">");
            printf(" %.2lf", indemnite_maladie);
            printf("</TD>");
            printf("</TR>");

            printf("<TR>");
            printf("<TD>");
            printf("R&eacute;duction indemnit&eacute;s maladie-invalidit&eacute; : ");
            printf("</TD>");
            printf("<TD align=\"right\">");
            printf("-%.2lf" , (float)reduction_indemnite_maladie/100);
            printf("</TD>");
            printf("</TR>");
          }

          // Reduction pour pensions, prepensions et autres revenus de remplacement
          if(pension_prepension_autre > 0){
```

```
            printf("<TR>");
            printf("<TD>");
            printf("Pensions, pr&eacute;pensions et autres revenus ");
            printf("de remplacement : ");
            printf("</TD>");
            printf("<TD>");
            printf("</TD>");
            printf("<TD align=\"right\">");
            printf(" %.2lf", pension_prepension_autre);
            printf("</TD>");
            printf("</TR>");

            printf("<TR>");
            printf("<TD>");
            printf("R&eacute;duction pour pensions, pr&eacute;pensions et ");
            printf("autres revenus de remplacement  :  ");
            printf("</TD>");
            printf("<TD align=\"right\">");
            printf("-%.2lf" , (float)reduction_pension/100);
            printf("</TD>");
            printf("</TR>");
        }

        // Reduction pour sursalaire suite a des heures supplementaires de travail
        if(heure_sup > 0){
            printf("<TR>");
            printf("<TD>");
            printf("Sursalaire suite &agrave; des heures suppl&eacute;mentaires ");
            printf("de travail : ");
            printf("</TD>");
            printf("<TD>");
            printf("</TD>");
            printf("<TD align=\"right\">");
            printf(" %.2lf", heure_sup);
            printf("</TD>");
            printf("</TR>");

            printf("<TR>");
            printf("<TD>");
            printf("R&eacute;duction pour sursalaire suite &agrave; des  ");
            printf("heures suppl&eacute;mentaires de travail");
            printf("</TD>");
            printf("<TD align=\"right\">");
```

```
            printf("-%.2lf" , (float)reduction_sursalaire/100);
            printf("</TD>");
            printf("</TR>");
3055      }

          printf("<TR>");
          printf("<TD>");
          printf(" ");
3060      printf("</TD>");
          printf("<TD>");
          printf("<hr align=\"center\" width=\"90%\" color=\"black\" size=\"2\">");
          printf("</TD>");
          printf("</TR>");
3065
          // Impot final
          printf("<TR>");
          printf("<TD>");
          printf("Imp&ocirc;t : ");
3070      printf("</TD>");
          printf("<TD align=\"right\">");
          printf("%.2lf" , (float)impot/100);
          printf("</TD>");
          printf("</TR>");
3075
          printf("</TABLE>");

          printf("<hr align=\"center\" width=\"90%\" color=\"black\" size=\"2\">");

3080      // Calcul final
          printf("<H2> Calcul final </H2>");

          printf("<TABLE>");
          printf("<TR>");
3085      printf("<TD>");
          printf("Imp&ocirc;t : ");
          printf("</TD>");
          printf("<TD align=\"right\">");
          printf("%.2lf" , (float)impot/100);
3090      printf("</TD>");
          printf("</TR>");

          printf("<TR>");
          printf("<TD>");
```

```
3095        printf("Centimes additionnels d'agglom&eacute;ration (%.2lf%) : "
                   ,taux_commune);
            printf("</TD>");
            printf("<TD align=\"right\">");
            printf("%.2lf" , ((float)impot/100)*taux_commune/100);
3100        printf("</TD>");
            printf("</TR>");


            printf("<TR>");
3105        printf("<TD>");
            printf("Pr&eacute;compte immobilier (non remboursable) : ");
            printf("</TD>");
            printf("<TD align=\"right\">");
            if(impot > 0){
3110          printf("- %.2lf" , (float)PI/100);
            }
            else{ // L'impot est negatif, dans ce cas on ne rembourse pas le prec. imm.
              printf("- 0.00");
              PI = 0;
3115        }
            printf("</TD>");
            printf("</TR>");


3120        if(credit_impot_arkimedes > 0){
              printf("<TR>");
              printf("<TD>");
              printf("Cr&eacute;dit d'imp&ocirc;t Arkimedes (non remboursable) : ");
              printf("</TD>");
3125          printf("<TD align=\"right\">");
              if(impot > 0){
                printf("- %.2lf" , (float)credit_impot_arkimedes/100);
              }
              else{ // L'impot est negatif, dans ce cas on ne rembourse pas ce credit
3130            printf("- 0.00");
                credit_impot_arkimedes = 0;
              }
              printf("</TD>");
              printf("</TR>");
3135        }

            printf("<TR>");
```

```
        printf("<TD>");
        printf("Pr&eacute;compte professionnel : ");
3140    printf("</TD>");
        printf("<TD align=\"right\">");
        printf("- %.2lf",precompte_prof_total);
        printf("</TD>");
        printf("</TR>");
3145
        if(credit_impot_internet > 0){
          printf("<TR>");
          printf("<TD>");
          printf("Cr&eacute;dit d'imp&ocirc;t internet pour tous : ");
3150      printf("</TD>");
          printf("<TD align=\"right\">");
          printf("- %.2lf",(float)credit_impot_internet/100);
          printf("</TD>");
          printf("</TR>");
3155    }

        if(versement_anticipe > 0){
          printf("<TR>");
          printf("<TD>");
3160      printf("Versements anticip&eacute;s : ");
          printf("</TD>");
          printf("<TD align=\"right\">");
          printf("- %.2lf",versement_anticipe);
          printf("</TD>");
3165      printf("</TR>");
          if(bonification > 0){
            printf("<TR>");
            printf("<TD>");
            printf("Bonification li&eacute;es aux versements anticip&eacute;s : ");
3170        printf("</TD>");
            printf("<TD align=\"right\">");
            printf("- %.2lf",(float)bonification/100);
            printf("</TD>");
            printf("</TR>");
3175      }
        }

        printf("<TR>");
        printf("<TD>");
3180    printf("Cotisation sp&eacute;ciale s&eacute;curit&eacute sociale : ");
```

```
        printf("</TD>");
        printf("<TD align=\"right\">");
        printf(" %.2lf",(float)cotisation_securite_sociale/100);
        printf("</TD>");
3185    printf("</TR>");

        if(retenue_sec_sociale > 0 ){
          printf("<TR>");
          printf("<TD>");
3190      printf("D&eacute;j&agrave; retenu sur la cotisation sp&eacute;ciale ");
          printf("s&eacute;curit&eacute sociale : ");
          printf("</TD>");
          printf("<TD align=\"right\">");
          printf(" - %.2lf",retenue_sec_sociale);
3195      printf("</TD>");
          printf("</TR>");
        }

        printf("<TR>");
3200    printf("<TD>");
        printf(" ");
        printf("</TD>");
        printf("<TD>");
        printf("<hr align=\"center\" width=\"90%\" color=\"black\" size=\"2\">");
3205    printf("</TD>");
        printf("</TR>");

        int impot_final = impot +
          impot*7/100 -  // centimes additionnels
3210      PI - // Precompte immobiliers
          credit_impot_arkimedes -
          (int)(precompte_prof_total*100) -
          credit_impot_internet -
          (int)(versement_anticipe*100) -
3215      bonification +
          cotisation_securite_sociale -
          (int)(retenue_sec_sociale*100);


3220    // Cas ou l'impot est du
        if(impot_final > 0){
          printf("<TR>");
          printf("<TD>");
```

```
          printf("Solde &agrave; <B> payer par le contribuable </B>: ");
          printf("</TD>");
          printf("<TD align=\"right\">");
          printf("<B> %.2lf </B>",(float)impot_final/100);
          printf("</TD>");
          printf("</TR>");
        }
        else{ // Cas ou on doit rembourser de l'argent
          printf("<TR>");
          printf("<TD>");
          printf("Solde &agrave; <B> rembourser </B> au contribuable: ");
          printf("</TD>");
          printf("<TD align=\"right\">");
          printf("<B> %.2lf </B>",-(float)impot_final/100);
          printf("</TD>");
          printf("</TR>");
        }

        printf("</TABLE>");

      }// Fin imposition non commune

      printf("<hr align=\"center\" width=\"90%\" color=\"black\" size=\"2\">");


      // Creation formulaire pour revenir au cadre1
      printf("<FORM action = \"../cadre1.php\">");
      printf("<P align=\"right\">");
      printf("<INPUT type=\"submit\" align=\"center\" value=\"Retourner ");
      printf("au cadre I\">");
      printf("</FORM>");

      printf("</BODY>\n");
      // Fin du document HTML
      printf("</HTML>\n");

      return 0;
    }

    int main(){
      return (Main_Wrapper());
    }
```

B.4 Code de la base de données

Le code présent dans cette section à été généré par *PhpMyAdmin*.

```
1   -- phpMyAdmin SQL Dump
    -- version 2.8.0.3-Debian-1
    -- http://www.phpmyadmin.net
    --
5   -- Serveur: localhost
    -- Généré le : Vendredi 17 Août 2007 à 15:51
    -- Version du serveur: 5.0.22
    -- Version de PHP: 5.1.2
    --
10  -- Base de données: `sef`
    --

    -- --------------------------------------------------------

15  --
    -- Structure de la table `CADRE1`
    --

    CREATE TABLE `CADRE1` (
20    `NumeroNational` varchar(25) character set utf8 NOT NULL,
      `TypeTitulaire` smallint(1) default NULL,
      `NumTel` varchar(25) character set utf8 default NULL,
      `NumCompte` varchar(14) character set utf8 default NULL,
      PRIMARY KEY  (`NumeroNational`)
25  ) ENGINE=MyISAM DEFAULT CHARSET=latin1;

    --
    -- Contenu de la table `CADRE1`
    --
30

    -- --------------------------------------------------------

    --
35  -- Structure de la table `CADRE10`
    --

    CREATE TABLE `CADRE10` (
      `v1397` double NOT NULL default '0',
```

```
40    `v2397` double NOT NULL default '0',
      `v1398` double NOT NULL default '0',
      `v2398` double NOT NULL default '0',
      `NumeroNational` varchar(25) character set utf8 NOT NULL,
      PRIMARY KEY  (`NumeroNational`)
45  ) ENGINE=MyISAM DEFAULT CHARSET=latin1;

    --
    -- Contenu de la table `CADRE10`
    --
50

    -- --------------------------------------------------------

    --
55  -- Structure de la table `CADRE11`
    --

    CREATE TABLE `CADRE11` (
      `v1570` double NOT NULL default '0',
60    `v2570` double NOT NULL default '0',
      `NumeroNational` varchar(25) character set utf8 NOT NULL,
      PRIMARY KEY  (`NumeroNational`)
    ) ENGINE=MyISAM DEFAULT CHARSET=latin1;

65  --
    -- Contenu de la table `CADRE11`
    --


70  -- --------------------------------------------------------

    --
    -- Structure de la table `CADRE12`
    --
75
    CREATE TABLE `CADRE12` (
      `v1075` varchar(10) character set utf8 default NULL,
      `NumeroNational` varchar(25) character set utf8 NOT NULL,
      PRIMARY KEY  (`NumeroNational`)
80  ) ENGINE=MyISAM DEFAULT CHARSET=latin1;

    --
```

```
-- Contenu de la table ‘CADRE12‘
--

-- --------------------------------------------------------

--
-- Structure de la table ‘CADRE2‘
--

CREATE TABLE ‘CADRE2‘ (
  ‘v1001‘ varchar(10) character set utf8 default NULL,
  ‘v1002‘ varchar(10) character set utf8 default NULL,
  ‘v1003‘ varchar(10) character set utf8 default NULL,
  ‘v1004‘ varchar(10) character set utf8 default NULL,
  ‘v1005‘ varchar(10) character set utf8 default NULL,
  ‘v1006‘ varchar(10) character set utf8 default NULL,
  ‘v1007‘ varchar(10) character set utf8 default NULL,
  ‘v1008‘ varchar(10) character set utf8 default NULL,
  ‘v1009‘ varchar(10) character set utf8 default NULL,
  ‘v1010‘ varchar(10) character set utf8 default NULL,
  ‘v1011‘ varchar(10) character set utf8 default NULL,
  ‘v1012‘ varchar(10) character set utf8 default NULL,
  ‘v1013‘ varchar(10) character set utf8 default NULL,
  ‘v1014‘ varchar(10) character set utf8 default NULL,
  ‘v1015‘ varchar(10) character set utf8 default NULL,
  ‘v1016‘ varchar(10) character set utf8 default NULL,
  ‘v1017‘ varchar(10) character set utf8 default NULL,
  ‘v1018‘ varchar(10) character set utf8 default NULL,
  ‘v1019‘ varchar(10) character set utf8 default NULL,
  ‘v1020‘ varchar(10) character set utf8 default NULL,
  ‘v1021‘ varchar(10) character set utf8 default NULL,
  ‘v1022‘ varchar(10) character set utf8 default NULL,
  ‘v1023‘ varchar(10) character set utf8 default NULL,
  ‘v1024‘ varchar(10) character set utf8 default NULL,
  ‘v1025‘ varchar(10) character set utf8 default NULL,
  ‘v1026‘ varchar(10) character set utf8 default NULL,
  ‘v1027‘ varchar(10) character set utf8 default NULL,
  ‘v1028‘ varchar(10) character set utf8 default NULL,
  ‘v1029‘ varchar(10) character set utf8 default NULL,
  ‘v1030‘ int(2) NOT NULL default ’0’,
  ‘v1031‘ int(2) NOT NULL default ’0’,
  ‘v1034‘ int(2) NOT NULL default ’0’,
```

```
  `v1035` int(2) NOT NULL default '0',
  `v1036` int(2) NOT NULL default '0',
  `v1037` int(2) NOT NULL default '0',
  `v1038` int(2) NOT NULL default '0',
  `v1039` int(2) NOT NULL default '0',
  `v1043` int(2) NOT NULL default '0',
  `v1044` int(2) NOT NULL default '0',
  `v1032` int(2) NOT NULL default '0',
  `v1033` int(2) NOT NULL default '0',
  `NumeroNational` varchar(25) character set utf8 NOT NULL,
  PRIMARY KEY  (`NumeroNational`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 COMMENT='table du cadre 2 de la declaration

--
-- Contenu de la table `CADRE2`
--


-- --------------------------------------------------------

--
-- Structure de la table `CADRE3`
--

CREATE TABLE `CADRE3` (
  `v1100` double NOT NULL default '0',
  `v2100` double NOT NULL default '0',
  `v1101` double NOT NULL default '0',
  `v2101` double NOT NULL default '0',
  `v1104` int(2) NOT NULL default '0',
  `v1105` double NOT NULL default '0',
  `v2105` double NOT NULL default '0',
  `v1106` double NOT NULL default '0',
  `v2106` double NOT NULL default '0',
  `v1107` double NOT NULL default '0',
  `v2107` double NOT NULL default '0',
  `v1108` double NOT NULL default '0',
  `v2108` double NOT NULL default '0',
  `v1109` double NOT NULL default '0',
  `v2109` double NOT NULL default '0',
  `v1110` double NOT NULL default '0',
  `v2110` double NOT NULL default '0',
  `v1111` double NOT NULL default '0',
```

```
  `v2111` double NOT NULL default '0',
  `v1112` double NOT NULL default '0',
  `v2112` double NOT NULL default '0',
  `v1113` double NOT NULL default '0',
  `v2113` double NOT NULL default '0',
  `v1115` double NOT NULL default '0',
  `v2115` double NOT NULL default '0',
  `v1116` double NOT NULL default '0',
  `v2116` double NOT NULL default '0',
  `v1114` double NOT NULL default '0',
  `v2114` double NOT NULL default '0',
  `v1123` double NOT NULL default '0',
  `v2123` double NOT NULL default '0',
  `v1124` double NOT NULL default '0',
  `v2124` double NOT NULL default '0',
  `v1125` double NOT NULL default '0',
  `v2125` double NOT NULL default '0',
  `v1130` double NOT NULL default '0',
  `v2130` double NOT NULL default '0',
  `v1131` double NOT NULL default '0',
  `v2131` double NOT NULL default '0',
  `v1132` double NOT NULL default '0',
  `v2132` double NOT NULL default '0',
  `v1147` double NOT NULL default '0',
  `v2147` double NOT NULL default '0',
  `NumeroNational` varchar(25) character set utf8 NOT NULL,
  PRIMARY KEY  (`NumeroNational`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

--
-- Contenu de la table `CADRE3`
--


-- --------------------------------------------------------

--
-- Structure de la table `CADRE4A`
--

CREATE TABLE `CADRE4A` (
  `v1250` double NOT NULL default '0',
  `v2250` double NOT NULL default '0',
```

```
        `v1249` double NOT NULL default '0',
        `v2249` double NOT NULL default '0',
        `v1248` double NOT NULL default '0',
215     `v2248` double NOT NULL default '0',
        `v1251` double NOT NULL default '0',
        `v2251` double NOT NULL default '0',
        `v1252` double NOT NULL default '0',
        `v2252` double NOT NULL default '0',
220     `v1253` double NOT NULL default '0',
        `v2253` double NOT NULL default '0',
        `v1245` double NOT NULL default '0',
        `v2245` double NOT NULL default '0',
        `v1291` double NOT NULL default '0',
225     `v2291` double NOT NULL default '0',
        `v1254` double NOT NULL default '0',
        `v2254` double NOT NULL default '0',
        `v1255` double NOT NULL default '0',
        `v2255` double NOT NULL default '0',
230     `v1256` double NOT NULL default '0',
        `v2256` double NOT NULL default '0',
        `v1257` double NOT NULL default '0',
        `v2257` double NOT NULL default '0',
        `v1258` double NOT NULL default '0',
235     `v2258` double NOT NULL default '0',
        `v1260` double NOT NULL default '0',
        `v2260` double NOT NULL default '0',
        `v1261` double NOT NULL default '0',
        `v2261` double NOT NULL default '0',
240     `v1262` double NOT NULL default '0',
        `v2262` double NOT NULL default '0',
        `v1263` double NOT NULL default '0',
        `v2263` double NOT NULL default '0',
        `v1264` double NOT NULL default '0',
245     `v2264` double NOT NULL default '0',
        `v1265` double NOT NULL default '0',
        `v2265` double NOT NULL default '0',
        `v1266` double NOT NULL default '0',
        `v2266` double NOT NULL default '0',
250     `v1268` double NOT NULL default '0',
        `v2268` double NOT NULL default '0',
        `v1292` double NOT NULL default '0',
        `v2292` double NOT NULL default '0',
        `v1293` double NOT NULL default '0',
```

```
255     `v2293` double NOT NULL default '0',
        `v1296` double NOT NULL default '0',
        `v2296` double NOT NULL default '0',
        `v1294` double NOT NULL default '0',
        `v2294` double NOT NULL default '0',
260     `v1295` double NOT NULL default '0',
        `v2295` double NOT NULL default '0',
        `v1297` varchar(10) character set utf8 default NULL,
        `v2297` varchar(10) character set utf8 default NULL,
        `v1269` double NOT NULL default '0',
265     `v2269` double NOT NULL default '0',
        `v1270` double NOT NULL default '0',
        `v2270` double NOT NULL default '0',
        `v1271` double NOT NULL default '0',
        `v2271` double NOT NULL default '0',
270     `v1272` double NOT NULL default '0',
        `v2272` double NOT NULL default '0',
        `NumeroNational` varchar(25) character set utf8 NOT NULL,
        PRIMARY KEY  (`NumeroNational`)
      ) ENGINE=MyISAM DEFAULT CHARSET=latin1 COLLATE=latin1_general_ci;
275
      --
      -- Contenu de la table `CADRE4A`
      --


280
      -- --------------------------------------------------------

      --
      -- Structure de la table `CADRE4B`
285   --

      CREATE TABLE `CADRE4B` (
        `v1281` double NOT NULL default '0',
        `v2281` double NOT NULL default '0',
290     `v1282` double NOT NULL default '0',
        `v2282` double NOT NULL default '0',
        `v1285` double NOT NULL default '0',
        `v2285` double NOT NULL default '0',
        `v1283` double NOT NULL default '0',
295     `v2283` double NOT NULL default '0',
        `v1246` double NOT NULL default '0',
        `v2246` double NOT NULL default '0',
```

```
  `v1247` double NOT NULL default '0',
  `v2247` double NOT NULL default '0',
  `v1286` double NOT NULL default '0',
  `v2286` double NOT NULL default '0',
  `v1287` double NOT NULL default '0',
  `v2287` double NOT NULL default '0',
  `v1290` varchar(10) character set utf8 default NULL,
  `v2290` varchar(10) character set utf8 default NULL,
  `NumeroNational` varchar(25) character set utf8 NOT NULL,
  PRIMARY KEY  (`NumeroNational`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

--
-- Contenu de la table `CADRE4B`
--


-- --------------------------------------------------------

--
-- Structure de la table `CADRE5`
--

CREATE TABLE `CADRE5` (
  `v1211` double NOT NULL default '0',
  `v2211` double NOT NULL default '0',
  `v1212` double NOT NULL default '0',
  `v2212` double NOT NULL default '0',
  `v1213` double NOT NULL default '0',
  `v2213` double NOT NULL default '0',
  `v1214` double NOT NULL default '0',
  `v2214` double NOT NULL default '0',
  `v1215` double NOT NULL default '0',
  `v2215` double NOT NULL default '0',
  `v1216` double NOT NULL default '0',
  `v2216` double NOT NULL default '0',
  `v1218` double NOT NULL default '0',
  `v2218` double NOT NULL default '0',
  `v1217` double NOT NULL default '0',
  `v2217` double NOT NULL default '0',
  `v1224` double NOT NULL default '0',
  `v2224` double NOT NULL default '0',
  `v1226` double NOT NULL default '0',
```

```
  `v2226` double NOT NULL default '0',
  `v1227` double NOT NULL default '0',
  `v2227` double NOT NULL default '0',
  `v1219` double NOT NULL default '0',
  `v2219` double NOT NULL default '0',
  `v1220` double NOT NULL default '0',
  `v2220` double NOT NULL default '0',
  `v1221` double NOT NULL default '0',
  `v2221` double NOT NULL default '0',
  `v1222` double NOT NULL default '0',
  `v2222` double NOT NULL default '0',
  `v1223` double NOT NULL default '0',
  `v2223` double NOT NULL default '0',
  `v1225` double NOT NULL default '0',
  `v2225` double NOT NULL default '0',
  `NumeroNational` varchar(25) character set utf8 NOT NULL,
  PRIMARY KEY  (`NumeroNational`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

--
-- Contenu de la table `CADRE5`
--


-- --------------------------------------------------------

--
-- Structure de la table `CADRE6`
--

CREATE TABLE `CADRE6` (
  `v1192` double NOT NULL default '0',
  `v2192` double NOT NULL default '0',
  `v1193` double NOT NULL default '0',
  `v2193` double NOT NULL default '0',
  `v1194` double NOT NULL default '0',
  `v2194` double NOT NULL default '0',
  `v1195` date default NULL,
  `v2195` date default NULL,
  `v1196` double NOT NULL default '0',
  `v2196` double NOT NULL default '0',
  `NumeroNational` varchar(25) character set utf8 NOT NULL,
  PRIMARY KEY  (`NumeroNational`)
```

```
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

--
-- Contenu de la table `CADRE6`
--


-- --------------------------------------------------------

--
-- Structure de la table `CADRE7`
--

CREATE TABLE `CADRE7` (
  `v1350` double NOT NULL default '0',
  `v2350` double NOT NULL default '0',
  `v1349` double NOT NULL default '0',
  `v2348` double NOT NULL default '0',
  `v1390` double NOT NULL default '0',
  `v2390` double NOT NULL default '0',
  `v1392` double NOT NULL default '0',
  `v1393` double NOT NULL default '0',
  `v2393` double NOT NULL default '0',
  `v1394` double NOT NULL default '0',
  `v1384` double NOT NULL default '0',
  `v1385` double NOT NULL default '0',
  `v2385` double NOT NULL default '0',
  `v1387` double NOT NULL default '0',
  `v1388` double NOT NULL default '0',
  `v1389` double NOT NULL default '0',
  `NumeroNational` varchar(25) character set utf8 NOT NULL,
  PRIMARY KEY  (`NumeroNational`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

--
-- Contenu de la table `CADRE7`
--


-- --------------------------------------------------------

--
-- Structure de la table `CADRE8`
```

```
--

CREATE TABLE `CADRE8` (
  `v1370` double NOT NULL default '0',
  `v2370` double NOT NULL default '0',
  `v1372` varchar(10) character set utf8 default NULL,
  `v2372` varchar(10) character set utf8 default NULL,
  `v1373` double NOT NULL default '0',
  `v2373` double NOT NULL default '0',
  `v1371` double NOT NULL default '0',
  `v2371` double NOT NULL default '0',
  `v1138` double NOT NULL default '0',
  `v2138` double NOT NULL default '0',
  `v1139` double NOT NULL default '0',
  `v2139` double NOT NULL default '0',
  `v1140` date default NULL,
  `v2140` date default NULL,
  `v1141` double NOT NULL default '0',
  `v2141` double NOT NULL default '0',
  `v1142` double NOT NULL default '0',
  `v2142` double NOT NULL default '0',
  `v1144` date default NULL,
  `v2144` date default NULL,
  `v1145` double NOT NULL default '0',
  `v2145` double NOT NULL default '0',
  `v1148` float NOT NULL default '0',
  `v2148` float NOT NULL default '0',
  `v1149` float NOT NULL default '0',
  `v2149` float NOT NULL default '0',
  `v1150` varchar(10) character set utf8 default NULL,
  `v1146` double NOT NULL default '0',
  `v2146` double NOT NULL default '0',
  `v1355` double NOT NULL default '0',
  `v2355` double NOT NULL default '0',
  `v1356` double NOT NULL default '0',
  `v2356` double NOT NULL default '0',
  `v1357` double NOT NULL default '0',
  `v2357` double NOT NULL default '0',
  `v1358` double NOT NULL default '0',
  `v2358` double NOT NULL default '0',
  `v1359` double NOT NULL default '0',
  `v2359` double NOT NULL default '0',
  `v1360` double NOT NULL default '0',
```

```
470     `v2360` double NOT NULL default '0',
        `v1351` double NOT NULL default '0',
        `v2351` double NOT NULL default '0',
        `v1352` double NOT NULL default '0',
        `v2352` double NOT NULL default '0',
475     `v1353` double NOT NULL default '0',
        `v2353` double NOT NULL default '0',
        `v1354` double NOT NULL default '0',
        `v2345` double NOT NULL default '0',
        `NumeroNational` varchar(25) NOT NULL,
480     PRIMARY KEY  (`NumeroNational`)
      ) ENGINE=MyISAM DEFAULT CHARSET=latin1;

      --
      -- Contenu de la table `CADRE8`
485   --


      -- --------------------------------------------------------

490   --
      -- Structure de la table `CADRE9`
      --

      CREATE TABLE `CADRE9` (
495     `v1361` double NOT NULL default '0',
        `v2361` double NOT NULL default '0',
        `v1362` double NOT NULL default '0',
        `v2362` double NOT NULL default '0',
        `v1365` double NOT NULL default '0',
500     `v2365` double NOT NULL default '0',
        `v1364` double NOT NULL default '0',
        `v2364` double NOT NULL default '0',
        `v1378` double NOT NULL default '0',
        `v2378` double NOT NULL default '0',
505     `v1379` double NOT NULL default '0',
        `v2379` double NOT NULL default '0',
        `v1363` double NOT NULL default '0',
        `v2363` double NOT NULL default '0',
        `v1369` double NOT NULL default '0',
510     `v1396` double NOT NULL default '0',
        `v2396` double NOT NULL default '0',
        `v1380` double NOT NULL default '0',
```

```
  `v2380` double NOT NULL default '0',
  `v1381` double NOT NULL default '0',
  `v2381` double NOT NULL default '0',
  `NumeroNational` varchar(25) character set utf8 NOT NULL,
  PRIMARY KEY  (`NumeroNational`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

--
-- Contenu de la table `CADRE9`
--
```

Bibliographie

[1] Shishir Gundavaram. *CGI Programming on the World Wide Web*. O'Reilly & Associates, Inc.

[2] Daniel Diaz. *GNU Prolog : A Native Prolog Compiler with Constraint Solving over Finite Domains*, 2002.

[3] Ulf Nilsson and Jan Maluszynski. *Logic, Programming and Prolog*. 1995.

[4] Kim Marriott and Peter J. Stuckey. *Programming with Constraints : An Introduction*. The MIT press, 1998.

[5] Wim Vanhoof. *Cours de programmation fonctionnelle et logique*. FUNDP Namur, 2005.

[6] Jean-Marie Jacquet. *Cours de technique d'intelligence artificielle*. FUNDP Namur, 2006.

[7] Alexis Saurin. L'implantation des contraintes globales en gnu prolog. Master's thesis, INRIA Rocquencourt, 2001.

[8] `http://www.phpmyadmin.net/home_page/docs.php`.

[9] `http://clip.dia.fi.upm.es/Software/Ciao/ciao_html/ciao_144.html`.

[10] Giovanni Sartor. Artificial intelligence and law : An introduction. *University of Bologna*.

[11] *L'almanach du contribuable*. 2007.

[12] Tayeb Lemlouma and Abdelmadjid Boudina. *Programmation logique avec contraintes*.

[13] `http://doc.ubuntu-fr.org/lamp`.

[14] `http://www.midwinter.com/~koreth/uncgi-c.html`.

[15] `http://www.belgium.be/eportal/index.jsp`.