

MINISTRY OF SCIENCE AND EDUCATION
NATIONAL TECHNICAL UNIVERSITY
"KHARKIV POLYTECHNIC INSTITUTE"
DEPARTMENT OF SOFTWARE ENGINEERING AND MANAGEMENT
INFORMATION TECHNOLOGIES

METHODICAL RECOMMENDATION TO
"BASICS OF SOFTWARE ENGINEERING
LABORATORY PRACTICE
PART 1"

For students of specialties
121 "Software Engineering",
122 "Computer Science and Intellectual Systems"

Kharkiv
NTU "KhPI"
2019

Methodical recommendation to “Basics of software engineering. Laboratory practice. Part 1” for students of Technical science / authors Melnyk K.V., Borisova N.V., Lutenko I.V., Ershova S.I., Smolin P.A., Grinchenko M.A. – Kharkiv : NTU “KhPI”. – 22 p.

Authors Melnyk K.V.,
 Borisova N.V.,
 Lutenko I.V.,
 Ershova S.I.,
 Smolin P.A.,
 Grinchenko M.A.

Reviewer Shmatko A.V.

Department of Software Engineering and Management Information
Technologies

CONTENT

Introduction	3
Individual task	5
Additional information about calculating the formulae.....	6
Lab № 1 Modelling of Domain.....	9
Lab № 2 Coding of Domain.....	16
Appendix A Requirements for report	20
Reference list.....	22

INTRODUCTION

Software engineering is an engineering discipline that is concerned with all aspects of software production [1].

Software engineering can be divided into sub-disciplines [2]. Some of them are:

- Software engineering management: The application of management activities – planning, coordinating, measuring, monitoring, controlling, and reporting – to ensure that the development and maintenance of software is systematic, disciplined, and quantified.
- Requirements engineering: The elicitation, analysis, specification, and validation of requirements for software.
- Software design: The process of defining the architecture, components, interfaces, and other characteristics of a system or component.
- Software construction: The detailed creation of working, meaningful software through a combination of programming, verification, unit testing, integration testing, and debugging.
- Software testing: An empirical, technical investigation conducted to provide stakeholders with information about the quality of the product or service under test.
- Software maintenance: The totality of activities required to provide cost-effective support to software.
- Software quality.

The outcome of software engineering is an efficient and reliable software product. Software product is a computer programs with all associated documentation and configuration data that is required to make these programs operate correctly [1]. Essential attributes of good software product: maintainability, dependability and security, efficiency, acceptability.

A software process is a sequence of activities that leads to the production of a software product. There are four fundamental activities that are common to all software processes. These activities are:

1. Software specification, where customers and engineers define the software that is to be produced and the constraints on its operation.
2. Software development, where the software is designed and programmed.
3. Software validation, where the software is checked to ensure that it is what the customer requires.
4. Software evolution, where the software is modified to reflect changing customer and market requirements.

The methodical recommendation is about modelling (or designing) and creating a software that will help to calculate some system of expressions with unknown variables. The values of variables can be obtained from different sources: from file or from keyboard.

INDIVIDUAL TASK

Choose your individual task from table 1.1 corresponding to number of group's list.

Table 1.1 – Individual task for lab

1	$y = \begin{cases} \sum_{i=1}^n (i+x)^2, x < 0 \\ \sum_{i=0}^{n-1} \prod_{j=1}^n \frac{x+i}{i-j}, x \geq 0 \end{cases}$	5	$y = \begin{cases} \sum_{i=2}^{n-1} \frac{x}{i}, x \leq 0 \\ \sum_{i=0}^{n-1} \sum_{j=0}^i \frac{i}{j+x}, x > 0 \end{cases}$
2	$y = \begin{cases} \sum_{i=1}^{n-1} \sum_{j=1}^n (x-i+j), x < 0 \\ \sum_{i=0}^{n-1} \frac{x}{i}, x \geq 0 \end{cases}$	6	$y = \begin{cases} \sum_{i=1}^n \sum_{j=1}^n \frac{1}{x-i-j}, x < 0 \\ \prod_{i=0}^{n-3} (x-i), x \geq 0 \end{cases}$
3	$y = \begin{cases} \prod_{j=2}^{n-2} (j+1), x < 0 \\ \sum_{i=0}^{n-1} \prod_{j=0}^{n-1} (x+i+j^2), x \geq 0 \end{cases}$	7	$y = \begin{cases} \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \frac{1}{x-i+j}, x \leq 0 \\ \prod_{i=1}^n \left(\frac{1}{x} - \frac{1}{i}\right), x > 0 \end{cases}$
4	$y = \begin{cases} \prod_{i=0}^{n-1} (i^2 + i), x \leq 0 \\ \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \frac{x}{i+j}, x > 0 \end{cases}$	8	$y = \begin{cases} \sum_{i=0}^n \prod_{j=i}^{n-1} (i^2 + j), x < 0 \\ \sum_{i=1}^{n-2} (i-x), x \geq 0 \end{cases}$
9	$y = \begin{cases} \sum_{i=1}^{n-1} \sum_{j=1}^n \frac{j}{j^2 + j}, x < 0 \\ x - \sum_{i=1}^{n-1} i, x \geq 0 \end{cases}$	11	$y = \begin{cases} \prod_{i=0}^{n-1} (i^3 + i), x \leq 0 \\ \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \frac{x}{i+j}, x > 0 \end{cases}$
10	$y = \begin{cases} \sum_{i=0}^n (x-i)^2, x \leq 0 \\ \prod_{i=1}^n \prod_{j=0}^{n-1} (x-i-j), x > 0 \end{cases}$	12	$y = \begin{cases} \prod_{j=2}^{n-2} j^2, x < 0 \\ \sum_{i=0}^{n-1} \prod_{j=0}^{n-1} (x+i^2+j), x \geq 0 \end{cases}$

ADDITIONAL INFORMATION ABOUT CALCULATING THE FORMULAE

Summation or **sigma notation** (denoted with an capital Greek sigma symbol Σ) is a convenient and simple form of shorthand used to give a concise expression for a sum of the values of a variable.

Let x_1, x_2, \dots, x_n denote a set of n numbers, where x_1 is the first number in the set, x_i represents the i th number in the set. Then a sum of the set is:

$$\sum_{i=1}^n x_i = x_1 + x_2 + \dots + x_n$$

where i represents the index of summation;

$i = 1$ is starting point of the index or lower limit of summation;

n is stopping point of the index or upper limit of summation (the index i is incremented by 1 for each successive term, stopping when $i = n$).

Examples:

$$\sum_{i=1}^3 i = 1 + 2 + 3 = 6$$

$$\sum_{i=2}^4 (i-1)^2 = (2-1)^2 + (3-1)^2 + (4-1)^2 = 14$$

There is concise expression for a **multiplication** of the values of a variable. The product operator in mathematics is indicated with capital **pi notation** Π . It is used in the same way as the Sigma symbol described above, except that succeeding numbers are multiplied instead of added:

$$\prod_{i=1}^n x_i = x_1 \cdot x_2 \cdot \dots \cdot x_n,$$

where i represents the index of multiplication;

$i=1$ is starting point of the index or lower limit of multiplication;

n is stopping point of the index or upper limit of multiplication (the index i is incremented by 1 for each successive term, stopping when $i = n$).

Examples:

$$\prod_{i=1}^3 i = 1 \cdot 2 \cdot 3 = 6$$

$$\prod_{i=0}^4 \left(i - \frac{1}{2}\right) = \left(0 - \frac{1}{2}\right) \cdot \left(1 - \frac{1}{2}\right) \cdot \left(2 - \frac{1}{2}\right) \cdot \left(3 - \frac{1}{2}\right) \approx -3.28$$

There are many complex formulae with double summation, double multiplication and mixed variant.

$$\sum_{i=1}^n \sum_{j=1}^n x_{ij} = \sum_{j=1}^n x_{1j} + \sum_{j=1}^n x_{2j} + \dots + \sum_{j=1}^n x_{nj}$$

$$\prod_{i=1}^n \prod_{j=1}^n x_{ij} = \prod_{j=1}^n x_{1j} \cdot \prod_{j=1}^n x_{2j} \cdot \dots \cdot \prod_{j=1}^n x_{nj}$$

$$\sum_{i=1}^n \prod_{j=1}^n x_{ij} = \prod_{j=1}^n x_{1j} + \prod_{j=1}^n x_{2j} + \dots + \prod_{j=1}^n x_{nj}$$

$$\prod_{i=1}^n \sum_{j=1}^n x_{ij} = \sum_{j=1}^n x_{1j} \cdot \sum_{j=1}^n x_{2j} \cdot \dots \cdot \sum_{j=1}^n x_{nj}$$

Example of double summation:

$$\sum_{i=1}^3 \sum_{j=2}^4 (i + j) = \sum_{j=2}^4 (1 + j) + \sum_{j=2}^4 (2 + j) + \sum_{j=2}^4 (3 + j) = 45$$

Authors: Melnyk K.V., Borisova N.V., Lutenko I.V., Ershova S.I., Smolin P.A., Grinchenko M.A.

Department of SEMIT, NTU «KhPI»

Kharkiv, 2019

Example of double multiplication:

$$\prod_{i=1}^3 \prod_{j=2}^4 (i + j) = \prod_{j=2}^4 (1 + j) \cdot \prod_{j=2}^4 (2 + j) \cdot \prod_{j=2}^4 (3 + j) = 15120000$$

Example of mixed variant:

$$\sum_{i=1}^3 \prod_{j=2}^4 (i + j) = \prod_{j=2}^4 (1 + j) + \prod_{j=2}^4 (2 + j) + \prod_{j=2}^4 (3 + j) = 390$$

$$\prod_{i=1}^3 \sum_{j=2}^4 (i + j) = \sum_{j=2}^4 (1 + j) \cdot \sum_{j=2}^4 (2 + j) \cdot \sum_{j=2}^4 (3 + j) = 3240$$

LAB № 1
MODELLING OF DOMAIN

Goal: Learning Use Case and Activity Diagrams by using Visual paradigm

Tasks:

1. Get the task and analyze the computational algorithms.
2. Develop the Activity diagram.
3. Develop the Use Case diagram.
4. Prepare the report of the work

Progress of the lab.

1. Get the task and analyze the computational algorithms.

According to the task in Table 1 student should analyze computational algorithms.

For example, we obtained the following task:

The application has to resolve the following system of formulae:

$$y = \begin{cases} \sum_{i=2}^n i^2, & x \leq -4 \\ x - \prod_{i=1}^{n-2} \sum_{j=1}^{n+1} (x + \frac{i}{j}), & x > -4 \end{cases}$$

The application should also provide the data input from keyboard and file and output to screen and file.

As we see in expressions, we have unknown variables x and n . To calculate the formulae student should input the values of x and n first. If we input $x \leq -4$, the application should calculate result of first expression, otherwise second. There is one constraint in the second expression for stopping point of the index i : upper

limit of multiplication should be bigger than starting value of i . Therefore, we obtain next one constraint: $n - 2 > 1 \Rightarrow n > 3$.

So, input data for resolving system of expressions are x and $n > 3$.

2. Develop the Activity diagram.

The free specialized software **Visual Paradigm Community Edition** [3] is recommended to draw the Activity diagram (Figure 1.1 - 1.2).

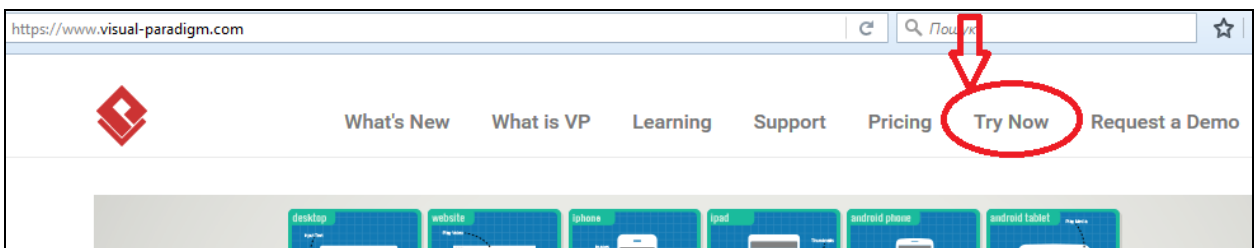


Figure 1.1 – Visual Paradigm main menu

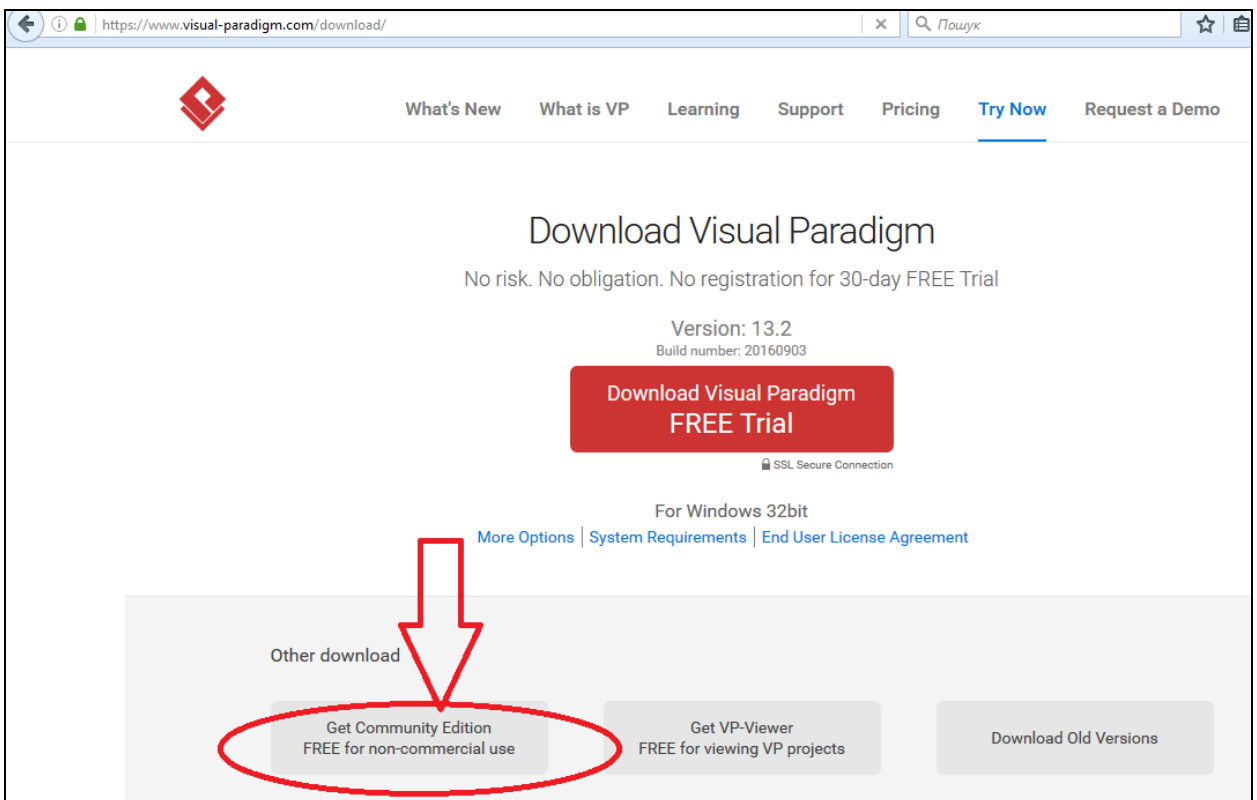


Figure 1.2 – Page with different versions of Visual Paradigm

Start Visual Paradigm. Create new project using menu: “File → New Project”. Define the name of project. Then create activity diagram and set the name using menu: “File → New Diagram → New Activity Diagram”; or click the “Activity Diagram” menu in the Diagram Navigator and then select the “New Activity Diagram”.

Activity diagram is used to describe dynamical properties of the system. For example, it is possible to use it to describe different algorithms that are necessary to implement in the system. The following nodes and edges are typically drawn on UML activity diagrams (Figure 1.3): activity, partition, action, object, control, activity edge.

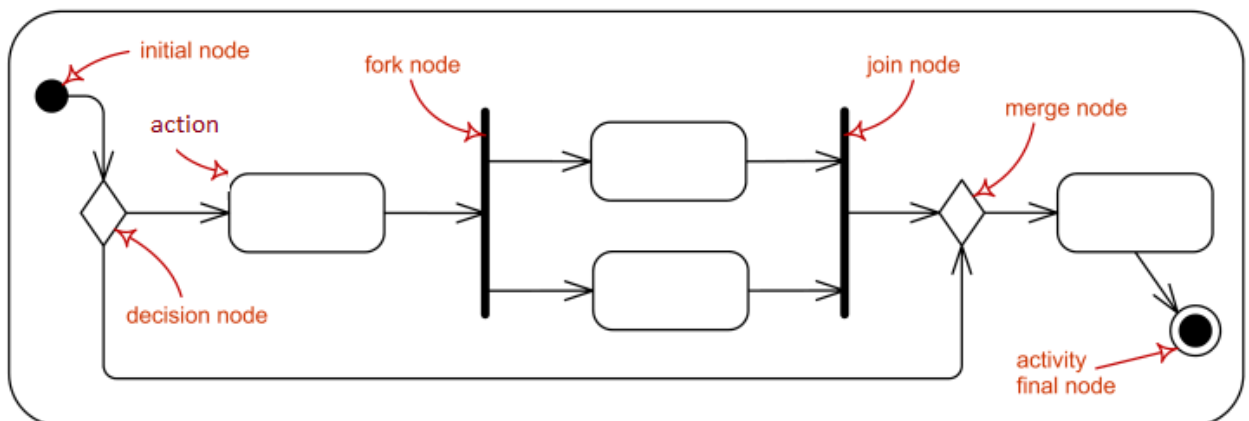


Figure 1.3 – Basic Activity Diagram elements

In our case the computational algorithm should look like shown on Figure 1.4.

There is possibility of changing font value in an activity diagram: right click an element or a diagram, then in “Styles and Formatting” menu student should choose “Formats” menu.

User can set the diagram transparency the same way: “Styles and Formatting → Transparent”.

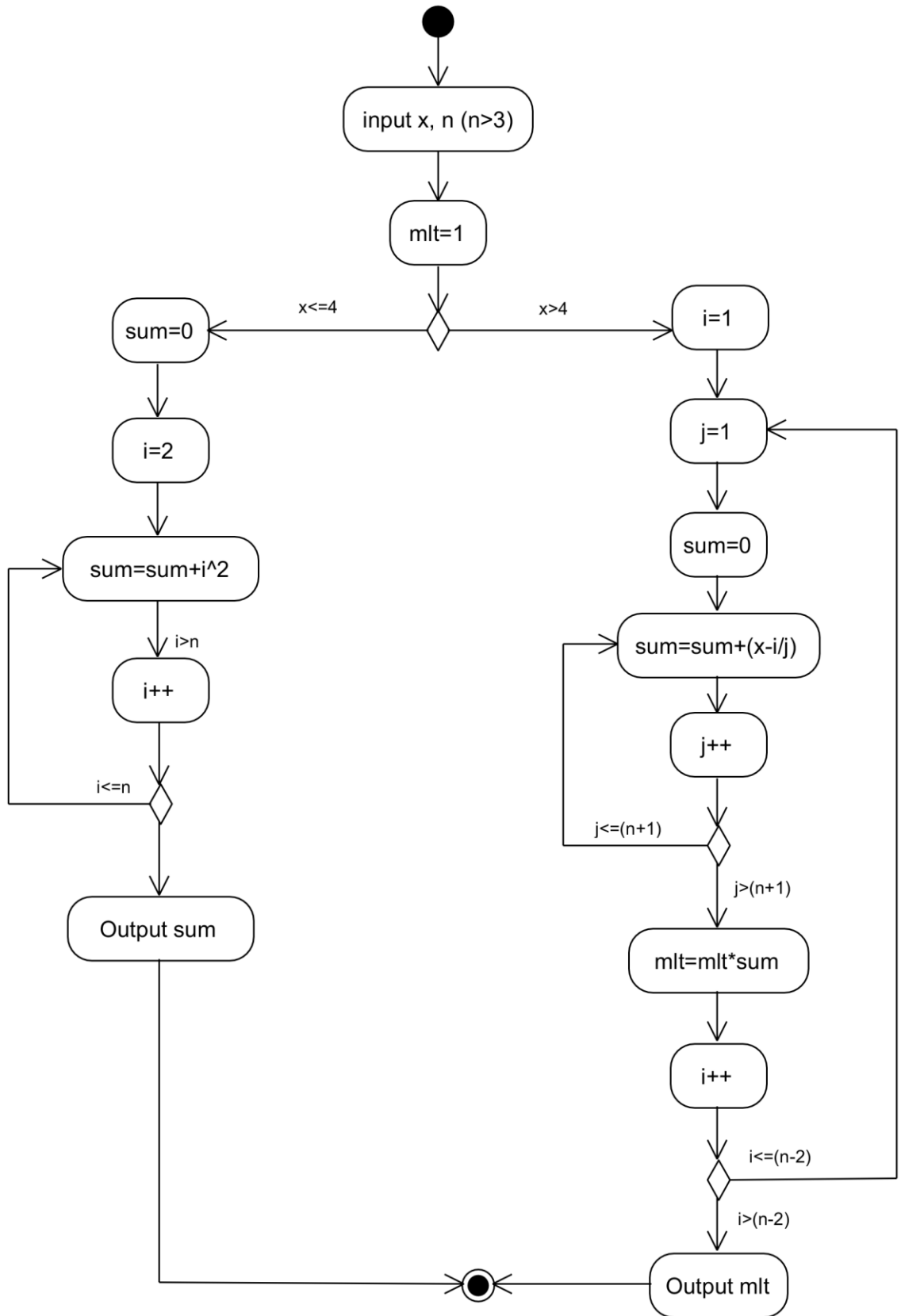




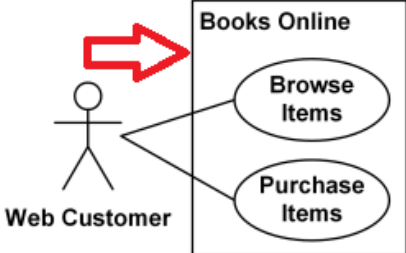
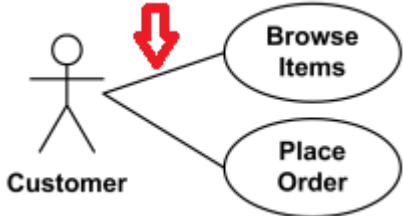
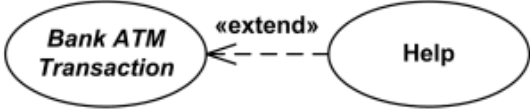
Figure 1.4 –Activity Diagram for chosen task

3. Develop the Use Case diagram.

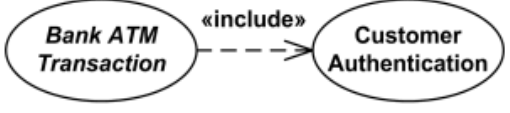
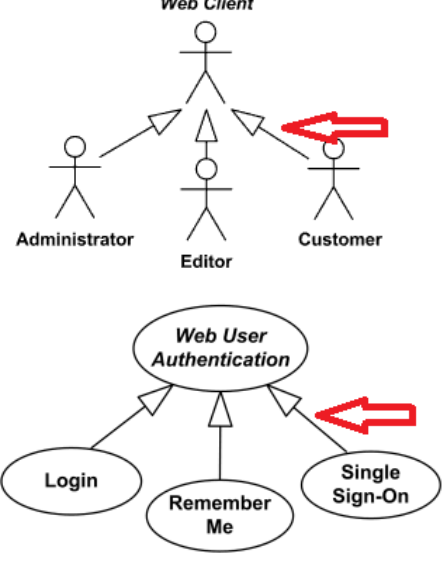
Create Use Case diagram and set the name using: “File → New Diagram → New Use Case Diagram” menu; or click the “Use Case Diagram” menu in the Diagram Navigator and then select the “New Use Case Diagram”.

Use case diagrams are used to describe a set of actions (use cases) that some system should or can perform in collaboration with one or more external users of the system (actors). The Use Case diagram reflects functional requirements of software. All main elements of Use Case diagram are described in the table 1.2.

Table 1.2 – Main elements of Use Case Diagram

 <p>An oval containing the text "User Registration".</p>	<p>Use cases describe functionality provided by systems, and determine the requirements the systems pose on their environment.</p>
 <p>A stick figure representing an actor, labeled "Student" below it.</p>	<p>An actor is a person, organization, or external system that plays a role in one or more interactions with your system.</p>
 <p>A rectangular box labeled "Books Online" containing two use cases: "Browse Items" and "Purchase Items". A stick figure labeled "Web Customer" is connected to the box by a line. A red arrow points from the box to the right.</p>	<p>System boundary boxes (optional) indicates the scope of system.</p>
 <p>A stick figure labeled "Customer" is connected to two use cases: "Browse Items" and "Place Order". A red arrow points down to the association lines.</p>	<p>An association between an actor and a use case indicates that the actor and the use case somehow interact or communicate with each other. An actor could be associated to one or several use cases.</p>
 <p>Two use cases: "Bank ATM Transaction" and "Help". A dashed arrow points from "Help" to "Bank ATM Transaction" with the label «extend» above it.</p>	<p>Extend Extending use case is optional, supplementary.</p>

End of the Table 1.2

 <p>«include»</p>	<p>Include Included use case required, not optional.</p>
	<p>Generalization Specialized use case is required, not optional, if base use case is abstract.</p>

The free specialized software Visual Paradigm Community Edition is recommended to draw the Use Case diagram. The Use Case diagram for the given task is shown on Figure 1.5.

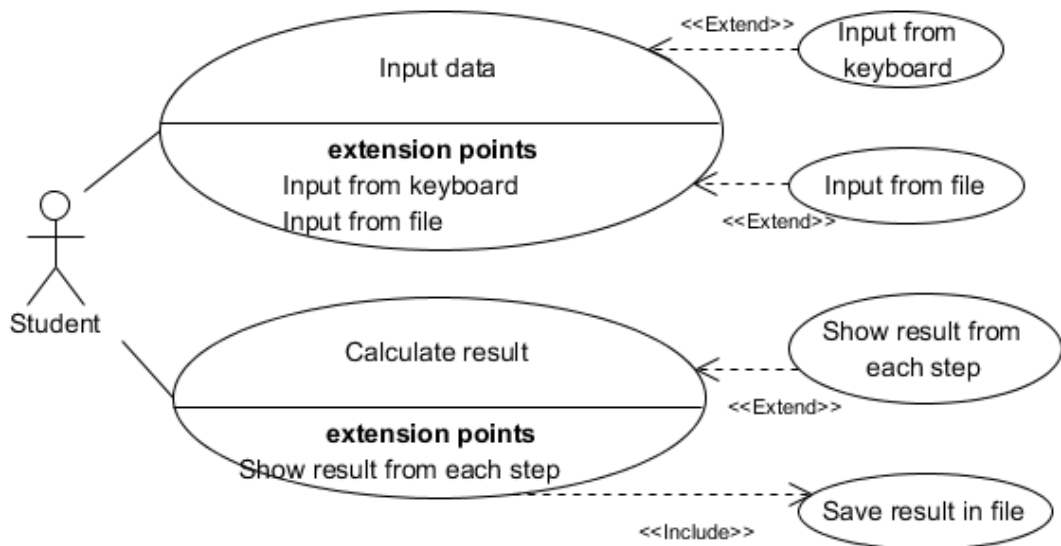


Figure 1.5 – Use Case Diagram for chosen task

There are two main use cases in the diagram:

- Input data: this functionality let user type necessary data from keyboard or read input data from outer file. The child use cases are optional, so in the diagram we should use arrow “extend”.
- Calculate result: here the application should save results in outer file always, so we should use arrow “input” for this functionality.

4. Prepare the report of the work

Requirements for the report are in Appendix A.

LAB № 2
CODING OF DOMAIN

Goal: Learning basic principles of C++

Tasks:

1. Develop the code for the task from previous lab.
2. Prepare the report of the work

Progress of the lab.

1. Develop the code for the task from previous lab.

According to the task from the previous lab you should develop computational algorithm in Visual Studio.

For example, you obtained the following task:

The application has to calculate the values by the following formula for the range with a predefined step:

$$y = \begin{cases} \sum_{i=2}^{n-2} (x-i)^2, & x < 7 \\ x + 7, & x \geq 7 \end{cases} \quad (2.1)$$

Solution. First, to calculate the first formula you should input the values of x and n . The following constraint: $n - 2 > 2 \Rightarrow n > 4$, so value of n should be greater than 4:

The following are some examples:

$$n = 5; x = 6 \Rightarrow y = \sum_{i=2}^{5-2} (x-i)^2 = \sum_{i=2}^3 (x-i)^2 = (6-2)^2 + (6-3)^2 = 25$$

$$n = 5; x = 8 \Rightarrow y = x + 7 = 8 + 7 = 15$$

Create a new Visual Studio C++ Project. On the menu bar, choose File→New→Project (Figure 2.1.).

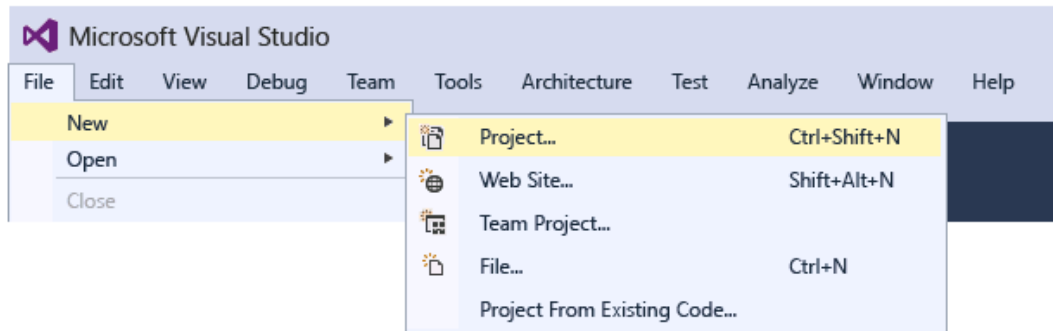


Figure 2.1 – Process of creating a blank project

In the Visual C++ category, choose the Win32 Console Application template, and then name the project (Figure 2.2).

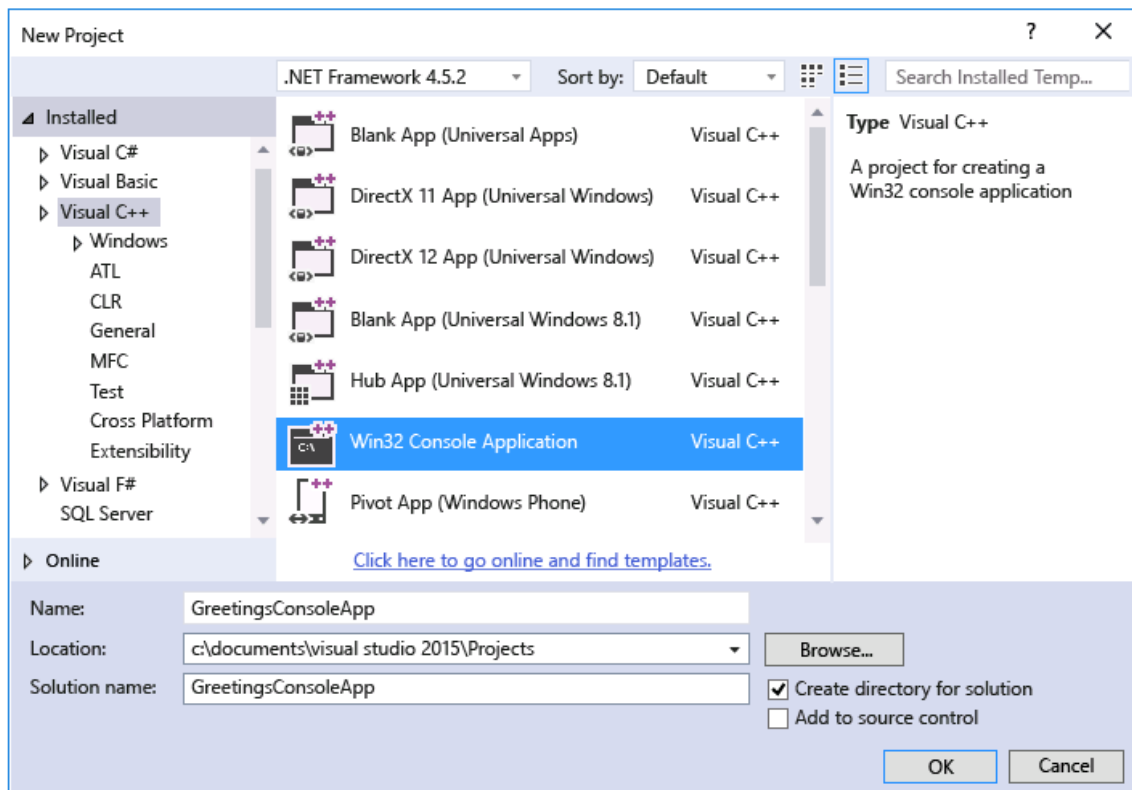


Figure 2.2 – Process of creating a blank project

When the Win32 Application Wizard appears, choose the Finish button.
Implement the task from the previous lab.

Additional information about C++ you may find in [4, 5].

The program for calculation of the function y is given on Figure 2.3.
Samples calculated by the program can be found on Figure 2.4.

```

1  #include <iostream>
2  #include <math.h>
3  using namespace std;
4
5  int main()
6  {
7
8  int x,n;
9  double y=0;
10
11     cout << "Input n>=4, n=";
12     cin >> n;
13     cout << "Input x= " ;
14     cin >> x;
15
16     if (x<7)
17     {
18         for (int i=2; i<=n-2; i++)
19         {
20             y+=pow((x-i),2);
21         }
22     }
23     else
24     {
25         y=x+7;
26     }
27     cout << "x= " << x << "; " << "y= " << y << endl;
28     return 0;
29 }

```

Figure 2.3 – Program for chosen task

```

Input n>=4, n=5
Input x= 6
x= 6; y= 25

```

```

Input n>=4, n=5
Input x= 8
x= 8; y= 15

```

Figure 2.4 – Result of execution the program

The next step is to modify the program. It should have the following possibilities:

- you should input predefined range (a,b) and step;
- program should calculate function y for each value x from range (a,b) (Figure 2.5);
- the results of each step should be displayed on the screen.

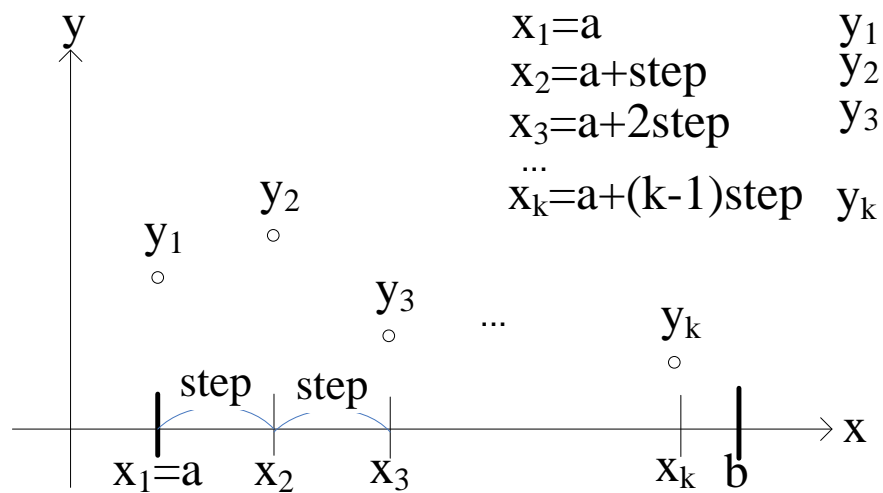


Figure 2.5 – Graphical representation of calculation

2. Prepare the report of the work

Requirements for the report are in Appendix A.

APPENDIX A

REQUIREMENTS FOR REPORT

Requirements for the report.

1. The report shall contain a cover page. Example for the cover page is shown on figure A.1.
2. The report shall contain the results of implementation of all the tasks of the lab in accordance with the individual task.
3. A structure of report is shown below:
 - Theme;
 - Goal;
 - Tasks;
 - Progress of work;
 - Conclusions (Example of Conclusions: during execution of this laboratory training we got skills and knowledge about ... We described...)
4. The report shall be drawn up according to the department standards. Template can be found by the following path:
\\SELENA\Method\Paper work\STVUZ.dot

MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE
NATIONAL TECHNICAL UNIVERSITY
“KHARKIV POLYTECHNIC INSTITUTE”
DEPARTMENT OF SOFTWARE ENGINEERING AND MANAGEMENT
INFORMATION TECHNOLOGIES”

Report of laboratory work № 1
Discipline «Basics of Software Engineering»

Executed by: Student of group KN-36zh

Ivanov I.

Checked by: Associate professor

Melnyk K. V.

Kharkov

2018

Figure A.1 – Cover page of report

REFERENCE LIST

1. Sommerville I. Software engineering / I. Sommerville // Boston: Pearson, 2011. ISBN 0-13-705346-0.
2. IEEE Computer Society. Software Engineering Body of Knowledge (SWEBOK Version 3) [Electronic resource] / Mode of access: www.swebok.org – 12.11.2018.
3. Visual Paradigm Community Edition [Electronic resource] / Mode of access: <https://www.visual-paradigm.com/> – 12.11.2018.
4. Richard L. Halterman. Fundamentals of Programming C++ / Richard L. Halterman // Publisher: Southern Adventist University, 2018. – 742 p.
5. Jon Kalb. Title C++ Today: The Beast Is Back / Jon Kalb, Gasper Azman // ISBN-13: 978-1491931660. – Publisher: O'Reilly Media, 2015. – 74 p.