МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ
НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
“ХАРЬКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ”

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к лабораторным работам
«Основы теории алгоритмов.
Ознакомление с рабочей средой Visual Basic»
из раздела «Основы теории алгоритмов.
Решения задач при помощи компьютера»
дисциплины «Информатика»
для студентов направления подготовки
6.050801 «Микро- и наноэлектроника»

Харьков НТУ «ХПИ» 2013

Методические указания к лабораторным работам «Основы теории алгоритмов. Ознакомление с рабочей средой Visual Basic» из раздела «Основы теории алгоритмов. Решения задач при помощи компьютера» дисциплины «Информатика» для студентов направления подготовки 6.050801 «Микро- и наноэлектроника» / Состав.: В.И. Шкалето, Р.В. Зайцев. – Харьков: НТУ «ХПИ», 2013. – 47 с.

Составители: В.И. Шкалето,
Р.В. Зайцев

Рецензент проф. О.О. Булгаков

Кафедра физического материаловедения для электроники и гелио-энергетики

ВВЕДЕНИЕ

Методические указания к лабораторным раотам по разделу «Основы теории алгоритмов. Решения задач при помощи компьютера» дисциплины «Информатика» касаются трех лабораторних работ: «Алгоритмы. Способы описания. Основные алгоритмические структуры», «Составление алгоритма методом пошаговой детализации. Сспомогательные алгоритмы» и «Ознакомление с рабочей средой Visual Basic. Первый проект».

Теория алгоритмов как самостоятельная наука появилась в 30-40х годах XX-века и имеет огромное значение во всех направлениях математических наук. Благодаря этой теории находят свои точные определения такие фундаментальне понятия как алгоритм, доказуемость, сложность.

Теория алгоритмов вместе с математической логикой служит основой для построения теории вычислений. Они составляют теоретическую основу для проектирования и применения вычислительных устройств к плохо формализуемым объектам. Именно, благодаря теории алгоритмов происходит внедрение математических методов в экономику, лингвистику, психологию, педагогику и другие гуманитарные науки.

Основным понятием теория алгоритмов является понятие «алгоритм». Слово алгоритм связано с именем великого средневекового математика, жившего в IX-веке, Мухаммеда ибн Мусе из Хорезми, который впервые изложил десятичную систему счисления и правила выполнения простых арифметических действий над элементами этой системы.

Одна из основных его работ, так называемый «арифметический трактат», которая была переведена на латинский язык в XII веке, начинается словами «Dixit algorizmi», что в переводе означает «правил автор, т.е. сказал Ал Хорезми». Отсюда и появилось слово алгоритм.

Таким образом, слово «алгоритм» – латинизированное имя Ал-Хоразми. Как научный термин это слово первоначально обозначало лишь правила арифметических операций над целыми числами и простыми дробями. Затем его стали употреблять в более широком смысле.

Со временем в математических науках под алгоритмом стали понимать точные предписания, правила о выполнении некоторых простых операций, определяющих процесс преобразования исходных данных в искомый результат.

Исполнителем алгоритма может быть как человек (кулинарные рецепты, различные инструкции, алгоритмы математических вычислений), так и техническое устройство. Различные машины (компьютеры, промышленные роботы, современная бытовая техника) являются формальными исполнителями алгоритмов. От формального исполнителя не требуется понимание сущности решаемой задачи, но требуется точное выполнение последовательности команд.

ЛАБОРАТОРНАЯ РОБОТА 1

АЛГОРИТМЫ. СПОСОБЫ ОПИСАНИЯ. ОСНОВНЫЕ АЛГОРИТМИЧЕСКИЕ СТРУКТУРЫ

Цель работы – Изучить способы записи алгоритмов. Изучить типовые алгоритмы: линейные, разветвленные и циклические. Научиться записывать алгоритмы в построчной записи и в виде блок-схемы.

1.1 Общие сведения

1.1.1 Понятие алгоритма

Алгоритм – это система правил, описывающая последовательность действий, которые необходимо выполнить, чтобы решить задачу. Понятие алгоритма в информатике является фундаментальным, таким, каким являются понятия точки, прямой и плоскости в геометрии, множества – в математике, пространства и времени – в физике, вещества – в химии. Как и для всякого фундаментального понятия, для алгоритма невозможно дать строгое определение. Поэтому формулировка, приведенная выше, лишь приближенно описывает алгоритм.

Разработанный алгоритм может исполнять как человек, так и техническое устройство (например, компьютер, робот), которые понимают правила записи алгоритма и умеют выполнять предписываемые им действия. Придерживаясь указаний алгоритма, исполнитель получит пригодный результат даже в том случае, когда он не понимает существа решаемой задачи.

Выполняя инструкции, записанные в виде программы (это тоже форма записи алгоритма), компьютер получит результат, удовлетворяющий пользователя.

Всякий алгоритм обработки информации характеризуется дискретностью, определенностью, результативностью и массовостью.

Дискретность означает, что выполнение алгоритма разбивается на последовательность законченных действий – шагов. Каждое действие должно быть завершено исполнителем прежде, чем он перейдет к выполнению следующего. Значения величин в каждом шаге алгоритма получаются по определенным правилам из значений величин, определенных на предшествующем шаге.

Под определенностью понимается то обстоятельство, что каждое правило алгоритма настолько четко и однозначно, что значения величин, получаемые на каком-либо шаге, однозначно определяются значениями величин, полученными на предыдущем шаге, и при этом точно известно, какой шаг будет выполнен следующим. Это означает также, что исполнитель, например компьютер, должен быть в состоянии выполнить каждую команду алгоритма в строгом соответствии с ее назначением.

Результативность (или конечность) алгоритма предполагает, что его исполнение сводится к выполнению конечного числа действий и всегда приводит к некоторому результату. В качестве одного из возможных результатов является и установление того факта, что задача решения не имеет.

Под массовостью понимается, что алгоритм решения задачи разрабатывается в общем виде так, чтобы его можно было применить для целого класса задач, различающихся лишь наборами исходных данных. При этом исходные данные могут выбираться из некоторой области, называемой областью применимости алгоритма. В свойстве массовости заключена основная практическая ценность алгоритмов.

Алгоритмизация – процесс получения и формулирования алгоритма. Принято выделять общую (абстрактную) и прикладную (структурную) теории алгоритмов.

Общая теория алгоритмов рассматривает вопросы принципиальной осуществимости алгоритма без учета ограничений, накладываемых технической системой на его реализацию. Основная проблема общей теории алгоритмов – проблема алгоритмической разрешимости той или иной задачи или класса задач. При этом под алгоритмической разрешимостью понимается возможность алгоритмизации исходной задачи.

Для составления алгоритма решения задачи на ЭВМ необходимо, прежде всего, формализовать задачу, т.е. представить ее в виде последовательности математических соотношений. Затем преобразовать эти соотношения в последовательность действий и логических связей между ними.

Многие задачи могут быть решены с помощью различных точных или численных методов и вычислительных методик, выбор которых связан с требованиями, предъявляемыми постановкой задачи и возможностями ее реализации на конкретной ЭВМ (точность решения, быстрота получения результата, необходимость получения и сохранения промежуточных результатов, стоимость подготовки и решения и т.д.).

При составлении алгоритма следует учитывать и ограничения, не допускающие автоматической работы ЭВМ: деление на нуль, деление на разность близких чисел (вызовет переполнение разрядной сетки), невозможность представления логарифмов отрицательных чисел и нуля и т.п.

1.1.2 Способы описания алгоритмов

Любой алгоритм записывается в виде определенной последовательности выполнения некоторых действий над входной информацией. Эти действия будем называть операциями. Операции могут кодироваться определенными символами, изображающими арифметические, логические и другие операции. В зависимости от используемых символов возможны различные способы записи алгоритмов.

Основными среди них являются:

- словесное описание алгоритма на естественном языке (вербальная форма);
- построчная запись алгоритма;
- операторный способ (на языке операторных схем);
- предикативный способ (в виде предикатных функций);
- блок-схема (графический способ);
- запись на каком-либо языке программирования.

Каждый из этих способов должен удовлетворять требованиям наглядности и немногочисленности используемых символов, исключения неоднозначных толкований и возможности составления обозримых алгоритмов.

Рассмотрим особенности каждой из названных форм и в качестве примера представим в каждой форме один и тот же алгоритм для определения наибольшего общего делителя (НОД) двух целых положительных чисел (алгоритм Евклида).

1.1.2.1 Словесное описание

Словесное описание имеет минимум ограничений и является наименее формализованным. Однако при этом алгоритм получается и наименее строгим, допускающим появление неопределенностей. Алгоритм в вербальной форме может оказаться очень объемным и трудным для восприятия человеком.

Например, алгоритм Евклида: если числа равны, НОД равен одному из них. В противном случае надо из большего числа вычесть меньшее, полученную разность запомнить вместо значения большего числа и повторить все сначала.

1.1.2.2 Построчная запись алгоритма

Это запись на естественном языке, но с соблюдением некоторых дополнительных правил. Сформулируем эти правила.

1. Шаги (предписания) алгоритма нумеруются.
2. Исполнение алгоритма происходит в порядке возрастания номеров шагов, начиная с первого (если не встречается никаких специальных указаний).
3. Типичными шагами алгоритма являются:

а) чтение (ввод) данных, которое записывается в виде

`Чтение A, B, ...,` где A, B,... – обозначение исходных данных;

б) обработка данных (вычисления) по формулам, записанная в виде

`V = …,` где V – переменная, значение которой определяется как результат, полученный после выполнения операций, указанных в правой части.

По существу знак « = » обозначает специальную операцию, которая называется присваиванием;

в) сообщение (вывод) результата, который имеет вид

`Запись x, y, …`, где x, y, ... – обозначение переменных, значение которых необходимо узнать;

г) проверка условия, запись которого

`Если … идти к N`, где вместо многоточия записывается условие, при выполнении которого осуществляется переход к шагу с номером N (если условие не выполняется, то производится переход к следующему по порядку шагу);

д) переход к шагу с номером N:

`Идти к N;`

е) конец вычислений:

`Останов.`

Пример – алгоритм Евклида:

```
[1] Чтение A, B
[2] Если A = B, идти к [8]
[3] Если A > B, идти к [6]
[4] B = B – A
[5] Идти к [2]
[6] A = A – B
[7] Идти к [2]
[8] НОД = A
[9] Запись НОД
[10] Останов
```

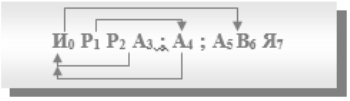
Построчная запись алгоритма позволяет избежать неопределенностей в алгоритме, не требует, по существу, никаких специальных знаний (понимание правил, перечисленных выше, трудности не составляет) и в то же время обеспечивает отработку навыков логически строгого изложения хода решения задачи (последовательности вычислений, возможных вариантов перехода к различным шагам алгоритма и т.д.) и облегчает последующее изучение алгоритмических языков.

Однако построчная запись алгоритма воспринимается человеком тяжело и требует большого внимания при изучении (или записи) алгоритмов в этой форме представления.

1.1.2.3 Язык операторных схем

Язык операторных схем был предложен в 1953г. А.А. Ляпуновым. В качестве операторов используются буквенные символы русского алфавита (обозначения операторов могут быть и другими). Алгоритм строится в виде последовательности операторов, пронумерованных по порядку и записанных в одну строку. Принимается, что выполнение операторов осуществляется последовательно слева направо. Если такая последовательность нарушается, то между операторами ставится знак “;”, а переход к иному оператору обозначается символом “⌊k” (при переходе слева направо) или “⌋k” (при переходе справа налево), где k – номер оператора, какому передается управление.

Приведенный выше пример (алгоритм Евклида) на языке операторных схем будет иметь вид, приведенный на рисунке 1.1.



или

И0, Р1 L5 Р2 L4 А3 ⌋1; А4 ⌋1; А5 В6 Я5,

$И_i$ – оператор ввода исходных значений;
$В_l$ - оператор вывода результатов;
$А_j$ – арифметический оператор;
$Р_k$ – логический оператор;
$Я_n$ – оператор останова;
$\lfloor_m$ – переход слева направо;
$\rfloor_m$ – переход справа налево;
m – целое число, принимающее значения i, j, k, n.

Рисунок 1.1 – Пример алгоритма Евклида на языке операторных схем

1.1.2.4 Предикатная форма записи алгоритма

При использовании предикатной формы записи алгоритмов все операции записываются в виде:

`P{L}a; b,`

где P{L} – предикатная функция;

L – условие, принимающее одно из значений «Да» или «Нет»;

a, b – обозначение каких-либо действий или команд.

При этом запись расшифровывается так: если условие L выполняется, то перейти к операции или команде, обозначенной буквой “a”; если условие L не выполняется, то перейти к операции или команде, обозначенной буквой “b”.

Для условий приведенного выше примера (алгоритм Евклида) запись алгоритма будет иметь вид:

1. `P{T ≤ Δt};` - ввод значений переменных A и B
2. `P{A = B} a;` - проверка условия A = B
3. `P{A > B} b; c,` - проверка условия A > B

где t – текущее время;

T – время цикла ввода A и B;

a – формирование вычисления НОД = A и конец;

b – формирование вычисления A = A – B, переход к 2;

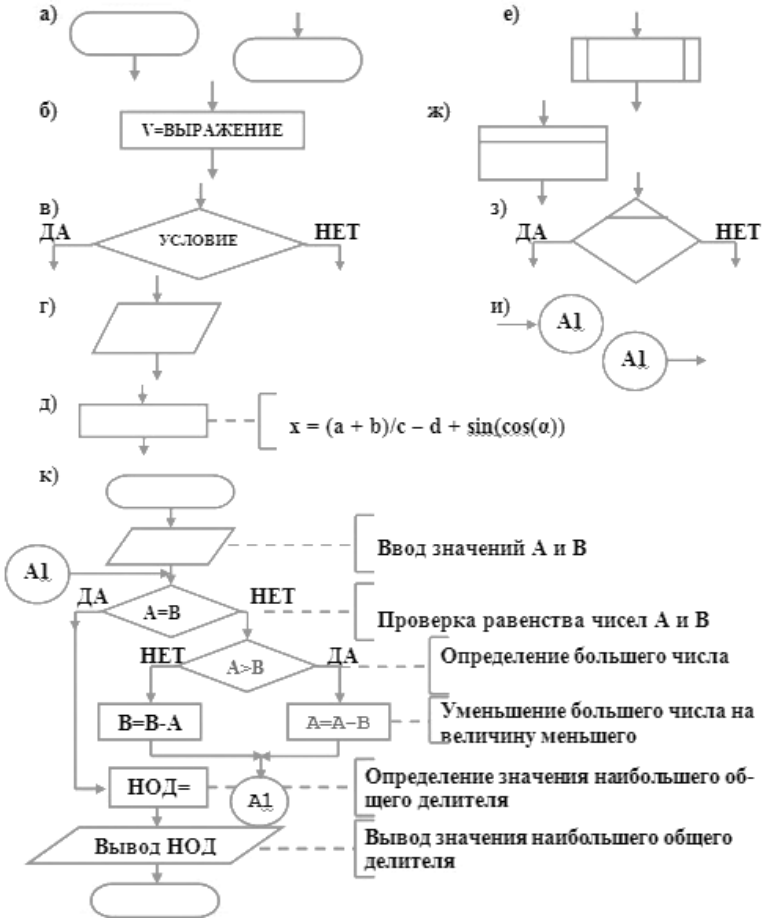
c – формирование вычисления B = B – A, переход к 2.

Область применения предикатной формы – запись алгоритмов с большим количеством логических условий. Наиболее существенный недостаток этой формы записи – плохая обозримость алгоритма.

1.1.2.5 Блок-схема

Изображение алгоритма в виде блок-схемы отличается высокой степенью наглядности. Блок-схема состоит из соединенных между собой

стрелками (линиями потока) блоков различного вида (рис. 1.2, а – и) и необходимого количества комментариев.



а – начало, конец; б – обработка данных; в – проверка условия; г – ввод-вывод; д – блок обработки данных с комментарием; е – предопределенный процесс; ж, з - детализированная программа; и – соединитель; к – пример блок-схемы для алгоритма Евклида

Рисунок 1.2 – Обозначения на блок-схемах

Выполнение алгоритма всегда начинается с блока начала и оканчивается при попадании на блок конца. Порядок вычислений определяется стрелками.

В блоке обработки данных содержится описание тех действий, которые должны быть выполнены над объектами при попадании на этот блок по входящей в него стрелке. Здесь вычисляются выражения и присваиваются новые значения переменным.

Проверка условия изображается с помощью блока принятия решения, внутри которого записывается это условие. В результате проверки выбирается одна из двух стрелок, определяющая направление дальнейших вычислений.

Внутри блока ввода или вывода пишется (необязательно) слово «Ввод» или «Вывод» и перечисляются переменные, значения которых должны быть введены или выведены в данном месте схемы.

Комментарии используются в тех случаях, когда пояснение не помещается внутри блока. Совокупность комментариев должна делать блок-схему понятной для любого пользователя.

Нередко возникает необходимость применения уже имеющихся (разработанных кем-то) алгоритмов. В этом случае можно использовать блок «предопределенный процесс». Если же часть алгоритма подлежит уточнению (детальной проработке) в дальнейшем, то в блок-схеме эта часть изображается с помощью блока «детализированная программа».

При большой насыщенности блок-схем блоками допускается прерывать стрелки, а затем продолжать их в нужном месте (т.е. как бы удалять часть стрелки, пересекающую блок-схему). В этом случае начало и конец удаленных участков обозначаются соединителями, внут-ри которых записываются (для каждой стрелки) одни и те же обозначения: буква, буква и цифра или цифра.

Помимо этих блоков допустимо использование и некоторых других, однако перечисленных выше вполне достаточно для разработки учебных блок-схем.

Пример записи алгоритма Евклида показан на рисунке 1.2, к.

1.1.2.6 Запись алгоритма на языке программирования

Эта запись представляет собой форму изображения алгоритма в том случае, когда исполнителем алгоритма является компьютер. Языки программирования имеют более жесткие правила, чем, например, правила описания алгоритма на естественном языке, так как их должна «понимать» машина. Однако лишние сложности записи алгоритма на таких языках окупаются возможностью их автоматического «прочтения» и исполнения.

Запись алгоритма Евклида на языке БЕЙСИК

```
10 INPUT "Введите числа A и B"; A, B: REM Ввод чисел A и B
20 IF A = B THEN 80: REM Проверка равенства чисел A и B
30 IF A>B THEN 60: REM Больше ли A, чем B?
40 B = B - A: REM Уменьшение числа B
50 COTO 20: REM Переход на проверку условия равенства A и B
60 A = A - B: REM Уменьшение числа A
70 GOTO 20: REM Переход на проверку условия равенства A и B
```

```
80 NOD = A: REM Присваивание значения переменной NOD
90 PRINT "NOD="; NOD: REM Вывод значения NOD
100 END: REM Конец вычислений
```

В этом примере после слов REM приводятся комментарии, расшифровывающие действия команд в предыдущих строках примера, благодаря чему легко обнаруживается тесная связь между командами языка программирования (в данном случае языка БЕЙСИК) и предписаниями построчной записи.

Наибольшей наглядностью обладает графический способ записи, который позволяет применять различную степень детализации. Недостаток этого способа записи алгоритмов – невозможность выполнить формальные преобразования, что не позволяет упрощать структуру алгоритма. В этом отношении операторный способ обладает большими возможностями. Дальнейшее развитие этого способа – разработка специального языка записи алгоритмов, позволяющего выполнять равносильные преобразования алгоритмов, что составляет предмет изучения специального раздела прикладной математики.

1.1.3 Основные алгоритмические структуры

Таких структур всего три: линейная, разветвляющаяся и циклическая.

Линейная структура предполагает однократное выполнение одной и той же последовательности шагов при любых наборах исходных данных.

Для разветвляющейся структуры также характерно однократное выполнение последовательности шагов, однако состав этой последовательности определяется результатами проверки некоторого условия, т. е. зависит от обрабатываемой информации.

Циклическая структура обеспечивает многократное выполнение одной и той же последовательности шагов тела цикла с модифицируемой (изменяемой) информацией.

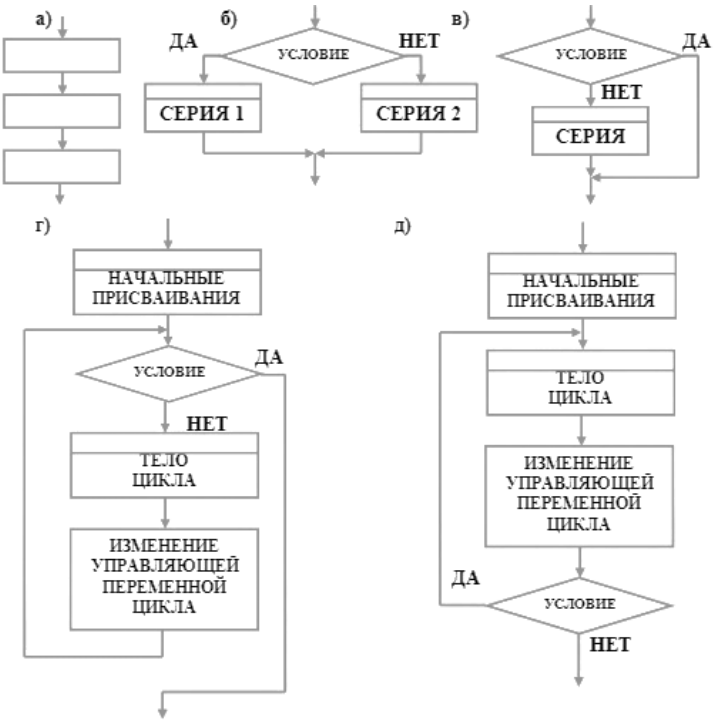
На рисунке 1.3 основные типы алгоритмических структур показаны в виде фрагментов блок-схем.

Заметим, что для одного набора исходных данных линейная и разветвляющаяся структуры выполнятся только по одному разу, цикл с постусловием – по крайней мере, один раз (так как первая проверка условия окончания цикла происходит после того, как тело цикла выполнено), цикл с предусловием может не выполниться ни разу.

Как для цикла с предусловием, так и для цикла с постусловием перед входом в цикл обязательно начальное присваивание, т.е. задание начальных значений тем переменным, которые используются для управления количеством повторений цикла и для накапливания результата.

Можно выделить циклы с известным заранее количеством повторений и так называемые итерационные циклы, в которых число прохож-

дений тела цикла заранее не определяется. Для циклов с известным количеством повторений задаются начальное и конечное значения переменной цикла. Ее значение изменяется при каждом повторении цикла по некоторому закону, который также задается. Этим и определяется количество повторений.



а – линейная; б – разветвляющаяся, в которой выполняется либо одна, либо другая серия шагов; в – разветвляющаяся, в которой серия шагов либо выполняется, либо пропускается; г – циклическая, в которой повторяющиеся шаги и изменение переменной цикла выполняются после проверки условия окончания цикла (цикл «пока», цикл с предусловием); д – циклическая, в которой повторяющиеся шаги и изменение переменной цикла осуществляются до проверки условия окончания цикла (цикл «до», цикл с постусловием)

Рисунок 1.3 - Основные типы алгоритмических структур

В итерационных циклах закон изменения переменной цикла в явном виде не описывается.

1.2 Порядок выполнения работы

1.2.1 Задания для выполнения работы

1. Включить компьютер.
2. Запустить на выполнение программу «Word».
3. Записать алгоритмы в построчной записи с помощью текстового редактора «Word».
4. Составить блок-схемы алгоритмов с помощью графической системы текстового редактора «Word».

1.2.2 Порядок выполнения

1. Согласно своему варианту записать линейный алгоритм 1 предложенной задачи с помощью построчной записи и в виде блок-схемы.
2. Согласно своему варианту записать разветвленный алгоритм 2 предложенной задачи в виде блок-схемы и привести его в виде построчной записи.
3. Согласно своему варианту записать циклический алгоритм 3 предложенной задачи в виде построчной записи и привести его в виде блок-схемы. Разработанный алгоритм предста-вить в трех возможных формах: в цикле перечисления, с предусловием и с постусловием.

1.2.3 Варианты заданий

Вариант № 1.

Алгоритм 1. Вычислить объем прямоугольной призмы, в основании которой – правильный треугольник.

Алгоритм 2. Вычислить значение функции:

$$\mathbf{f(x,y)} = \begin{cases} 3 * \mathbf{x}^2 + \mathbf{y}^2, \text{если } x > 0 \text{ и } y > 1 \\ 2 + 2 * x^2 - 3 * \mathbf{y}^2, \text{если } x \le 0 \text{ и } y \le 1 \\ x^3 + \mathbf{y}^3, \text{в других случаях} \end{cases}$$

Алгоритм 3. Вычислить сумму произведений соседних чисел в одномерном массиве чисел.

Вариант № 2.

Алгоритм 1. Вычислить объем конуса.

Алгоритм 2. Вычислить значение функции:

$$\mathbf{f(x,y)} = \begin{cases} 2 * \mathbf{x}^3 + \mathbf{y}^3, \text{если } x > 0 \text{ и } y > 1 \\ 1 + 4 * x^2 - 2 * \mathbf{y}^2, \text{ если } x \le 0 \text{ и } y \le 1 \\ x^{2/3} + \mathbf{y}^{2/3}, \text{ в остальных случаях.} \end{cases}$$

Алгоритм 3. Вычислить произведение двух матриц размером **n*m** и **m*n**.

Вариант № 3.

Алгоритм 1. Вычислить площадь трапеции.

Алгоритм 2. Вычислить значение функции:

$$z(x, y) = \begin{cases} x^{1/2} + 2 * y^{2/3}, \textit{если } x > 0 \textit{ и } y > 0 \\ 2 - x^2 + y^2, \textit{если } x \le 0 \textit{ и } y \le 0 \\ x^2 - y^{3/5}, \text{в остальных случаях.} \end{cases}$$

Алгоритм 3. Вычислить произведение логарифмов всех положительных чисел в прямоугольной матрице.

Вариант № 4.

Алгоритм 1. Вычислить площадь треугольника, из которого вырезан ромб.

Алгоритм 2. Вычислить значение функции:

$$z(x, y) = \begin{cases} x^{3/2} + 2 * y^{2/5}, \textit{если } x > 0 \textit{ и } y < 0 \\ 2 - x^{2/3} + y^2, \textit{если } x \le 0 \textit{ и } y \ge 0 \\ x^{2/5} - y^{2/5}, \text{в остальных случаях.} \end{cases}$$

Алгоритм 3. Вычислить сумму всех положительных чисел в прямоугольной матрице.

Вариант № 5.

Алгоритм 1. Вычислить площадь квадрата, из которого вырезан круг.

Алгоритм 2. Вычислить значение функции:

$$y(x, z) = \begin{cases} x^{1/3} + z^{1/3}, \text{если } x < 0 \text{ и } z > 0 \\ 1 + x^{1/2} - z^{1/2}, \text{если } x \ge 0 \text{ и } z \le 0 \\ x^2 - z^3, \text{в остальных случаях} \end{cases}$$

Алгоритм 3. Найти минимальное положительное число в прямоугольной матрице.

Вариант № 6.

Алгоритм 1. Вычислить объем параллелепипеда.

Алгоритм 2. Вычислить значение функции:

$$y = \begin{cases} \textbf{arctan}(2 * x), \text{если } x \ge \pi / 3 \\ \textbf{arcsin}(3 * x), \text{если } x \le 0 \\ x^2, \text{если } 0 < x < \pi/3 \end{cases}$$

Алгоритм 3. Найти минимальное число среди модулей отрицательных чисел в квадратной матрице.

Вариант № 7.

Алгоритм 1. Вычислить площадь треугольника, из которого вырезан круг.

Алгоритм 2. Вычислить значение функции:

$$y = \begin{cases} \mathbf{sin(}2 * x\mathbf{)}, \text{если } x \geq 2 * \pi / 3 \\ \mathbf{cos(}3 * x\mathbf{)}, \text{если } x \leq 0 \\ x/2, \text{если } 0 < x < 2 * \pi/3 \end{cases}$$

Алгоритм 3. Вычислить сумму логарифмов из абсолютных величин отрицательных чисел в прямоугольной матрице.

Вариант № 8.

Алгоритм 1. Вычислить полную площадь поверхности цилиндра.

Алгоритм 2. Вычислить значение функции:

$$y = \begin{cases} \mathbf{tg(}2 * x\mathbf{)}, \text{если } x \geq 5 * \pi / 3 \\ \mathbf{cosec(}3 * x\mathbf{)}, \text{если } x \leq 0 \\ 2/x, \text{если } 0 < x < 5 * \pi/3 \end{cases}$$

Алгоритм 3. Вычислить произведение всех отрицательных чисел в квадратной матрице.

Вариант № 9.

Алгоритм 1. Вычислить площадь трапеции, из которой вырезан круг.

Алгоритм 2. Вычислить значение функции:

$$z(x, y) = \begin{cases} \mathbf{sin}\left(x^{1/2} * y^{2/3}\right), \textit{если } x > 0 \textit{ и } y > 0 \\ \mathbf{exp}\left[-\left(x^2 + y^2\right)\right], \textit{если } x \leq 0 \textit{ и } y \leq 0 \\ x^2 - y^{3/5}, \text{в остальных случаях.} \end{cases}$$

Алгоритм 3. Вычислить сумму корней из абсолютных величин отрицательных чисел в прямоугольной матрице.

Вариант № 10.

Алгоритм 1. Вычислить площадь ромба, из которого вырезан прямоугольный равнобедренный треугольник.

Алгоритм 2. Вычислить значение функции:

$$z(x, y) = \begin{cases} x^{3/2} * y^{2/5}, \textit{если } x > 0 \textit{ и } y < 0 \\ x^{2/3} y^2, \textit{если } x \leq 0 \textit{ и } y \geq 0 \\ x^{2/5} y^{2/5}, \text{в остальных случаях.} \end{cases}$$

Алгоритм 3. Найти произведение всех положительных чисел в квадратной матрице.

Вариант № 11.

Алгоритм 1. Вычислить полную площадь поверхности октаэдра.

Алгоритм 2. Вычислить значение функции:

$$\mathbf{y(x,z)} = \begin{cases} \sqrt{\mathbf{x}^{1/3}\mathbf{z}^{1/3}}, \text{если } x < 0 \text{ и } z > 0 \\ \sqrt{x^{1/2}\,\mathbf{z}^{1/2}}, \text{если } x \geq 0 \text{ и } z \leq 0 \\ x^2\,\mathbf{z}^3, \text{in other case} \end{cases}$$

Алгоритм 3. Найти максимальное число в прямоугольной матрице.

Вариант № 12.

Алгоритм 1. Вычислить объем куба, из которого вырезана сфера.

Алгоритм 2. Вычислить значение функции:

$$y = \begin{cases} \exp(-2x), \text{ если } x \geq \pi/3 \\ \arcsin(3*x), \text{ если } x \leq 0 \\ x^2, \text{ если } 0<x<\pi/3 \end{cases}$$

Алгоритм 3. Вычислить произведение всех целых чисел в прямоугольной матрице.

Вариант № 13.

Алгоритм 1. Вычислить площадь круга, из которого вырезан правильный треугольник.

Алгоритм 2. Вычислить значение функции:

$$\boldsymbol{y} = \begin{cases} \mathbf{ln(2*x)}, \text{если } \mathbf{x} \geq 2*\boldsymbol{\pi}/3 \\ \mathbf{cos(3*x)}, \text{если } x \leq 0 \\ \exp(x/2), \text{если } 0 < x < 2*\boldsymbol{\pi}/3 \end{cases}$$

Алгоритм 3. Вычислить произведение из корней неотрицательных чисел в прямоугольной матрице.

Вариант № 14.

Алгоритм 1. Вычислить площадь круга, из которого вырезан прямоугольный равнобедренный треугольник.

Алгоритм 2. Вычислить значение функции:

$$\boldsymbol{y} = \begin{cases} \mathbf{cos(2*x)}, \text{если } \mathbf{x} \geq 1 \\ \mathbf{cos(3*x)}, \text{если } x \leq -1 \\ \arcsin(x/2), \text{если } -1 < x < 1 \end{cases}$$

Алгоритм 3. Найти произведение всех логарифмов положительных чисел в квадратной матрице.

Вариант № 15.

Алгоритм 1. Вычислить площадь боковой поверхности четырехгранной пирамиды.

Алгоритм 2. Вычислить значение функции:

$$z(x, y) = \begin{cases} \sin\left(x^{3/2}\right) y^{2/5}, \textit{если } x > 0 \textit{ и } y < 0 \\ \cos\left(y^{2}\right) x^{2/3}, \textit{если } x \le 0 \textit{ и } y \ge 0 \\ tg\left(x^{2/5} y^{2/5}\right), \text{в остальных случаях.} \end{cases}$$

Алгоритм 3. Вычислить сумму всех чисел в матрице, превышающих заданное.

1.2.4 Содержание отчета

В отчете должны быть представлены:

1. Цель работы.
2. Общие сведения.
3. Алгоритмы 1, 2 и 3 в виде построчной записи и в виде блок-схем.
4. Ответы на контрольные вопросы.

1.3 Контрольные вопросы

1. Дать определение алгоритма.
2. Что или кто является исполнителем алгоритма?
3. Перечислите основные характеристики алгоритма.
4. Что такое дискретность алгоритма?
5. Что понимается под определенностью алгоритма?
6. Для чего нужна конечность алгоритма?
7. Что такое массовость алгоритма?
8. Сколько форм представления алгоритма Вы знаете?
9. Что такое «Словесное описание» алгоритма?
10. Что такое «Построчная запись алгоритма» алгоритма?
11. Что такое «Язык операторных схем» при написании алгоритма?
12. Что такое «Предикатная форма записи алгоритма»?
13. Что такое «Блок-схема» алгоритма?
14. Является ли «Запись алгоритма на языке программирования» формой представления алгоритма?
15. Перечислите основные типы алгоритмических структур.
16. Что представляет собой «Линейная структура»?
17. Что представляет собой «Разветвляющаяся структура»?
18. Что представляет собой «Циклическая структура»?
19. Какие типы циклических структур Вы знаете?
20. Опишите циклическую структуру «перечисления» и порядок ее выполнения. Сколько раз может выполняться такая структура?
21. Опишите циклическую структуру «с предусловием» и порядок ее выполнения. Сколько раз может выполняться такая структура?

22. Опишите циклическую структуру «с постусловием» и порядок ее выполнения. Сколько раз может выполняться такая структура?

23. Опишите циклическую структуру «итерационные циклы» и порядок ее выполнения. Сколько раз может выполняться такая структура?

24. Какая структура может не выполниться ни разу?

ЛАБОРАТОРНАЯ РОБОТА 2

СОСТАВЛЕНИЕ АЛГОРИТМА МЕТОДОМ ПОШАГОВОЙ ДЕТАЛИЗАЦИИ. ВСПОМОГАТЕЛЬНЫЕ АЛГОРИТМЫ

Цель работы – Научиться разрабатывать типовые алгоритмы. Научиться разрабатывать и использовать типовые алгоритмы с помощью построчной запись алгоритма. Научиться разрабатывать и использовать типовые алгоритмы в виде блок-схемы.

2.1 Общие сведения

2.1.1 Вспомогательные алгоритмы

При разработке новых алгоритмов нередко возникают ситуации, когда в разных местах алгоритма необходимо обращение к одной и той же последовательности шагов обработки данных. Для такой последовательности создают отдельный алгоритм, называемый вспомогательным (обычно вспомогательные алгоритмы представляют собой алгоритмы некоторых процедур или функций), а в основном алгоритме предусматривают обращение к нему (программа имеет блочную структуру, в которой можно выделить главную программу и набор процедур и функций). В качестве вспомогательного алгоритма могут использоваться также алгоритмы, разработанные для других задач. Вспомогательный алгоритм делает структуру алгоритма более понятной и облегчает заимствование чужого опыта и знаний.

В алгоритмах, представляемых с помощью блок-схем, для обращения к вспомогательным алгоритмам используется блок «предопределенный процесс», внутри которого должно быть записано название (имя) вспомогательного алгоритма. Это же имя должно быть записано в блоке начала самого вспомогательного алгоритма.

Разработка алгоритмов, использующих вспомогательные алгоритмы, требует согласования имен переменных в основном и вспомогательном алгоритмах. Значения переменных основного алгоритма, которые используются во вспомогательном алгоритме (фактические параметры процедур и функций), требуется присвоить соответствующим переменным (формальным параметрам процедур и функций) вспомогательного алгоритма, а затем значения результатов, полученных во вспомогательном алгоритме, присвоить соответствующим переменным основного алгоритма.

Всякий алгоритм, содержащий обращение к вспомогательному, сам может быть использован в качестве вспомогательного алгоритма при разработке какого-то другого алгоритма.

Использование вспомогательных алгоритмов создает предпосылки к созданию и применению библиотеки алгоритмов. При этом работа поль-

зователя сводится к поиску библиотечных (проверенных и испытанных) алгоритмов и созданию основной программы очень простой структуры, состоящей в основном из обращений к вспомогательным библиотечным алгоритмам. Фрагмент алгоритма решения задачи «Сформировать одномерный массив A из минимальных элементов столбцов двумерного массива (матрицы) B, имеющего N строк и M столбцов.», использующего вспомогательный алгоритм МИН, представлен на рисунке 2.1.

В данном примере для вспомогательного алгоритма используются те же имена переменных, что и для основного алгоритма, В таком случае говорят, что параметры вспомогательного алгоритма являются глобальными, т. е. определенными еще до обращения к вспомогательному алгоритму.

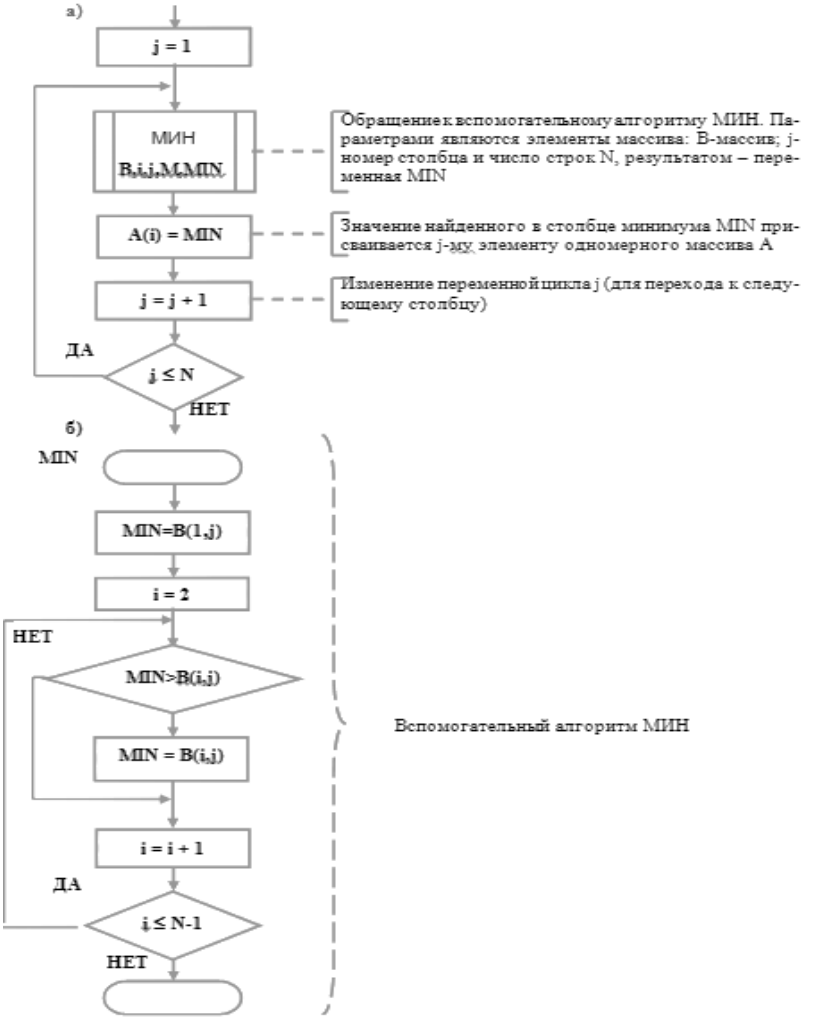
Вообще отработке знаний и умений согласования параметров основного и вспомогательного алгоритмов необходимо уделять большое внимание, так как любая попытка применения нужного алгоритма сразу ставит перед необходимостью решения этой задачи.

В схеме, показанной на рисунке 2.1.б, вспомогательный алгоритм создан специально для основного алгоритма, блок-схема которого приведена на рисунке 2.1.а. Однако в этом вспомогательном алгоритме решается задача определении минимума для одномерного массива. Поэтому более естественным было бы использование во вспомогательном алгоритме имен элементов одномерных массивов. В то же время основной алгоритм предназначен для обработки двумерного массива, поэтому в нем должна быть предусмотрена передача значений элементов, соответствующих столбцов двумерного массива основного алгоритма элементам одномерного массива вспомогательного алгоритма.

Алгоритм решения задачи для такого случая использования вспомогательного алгоритма показан на рисунке 2.2.

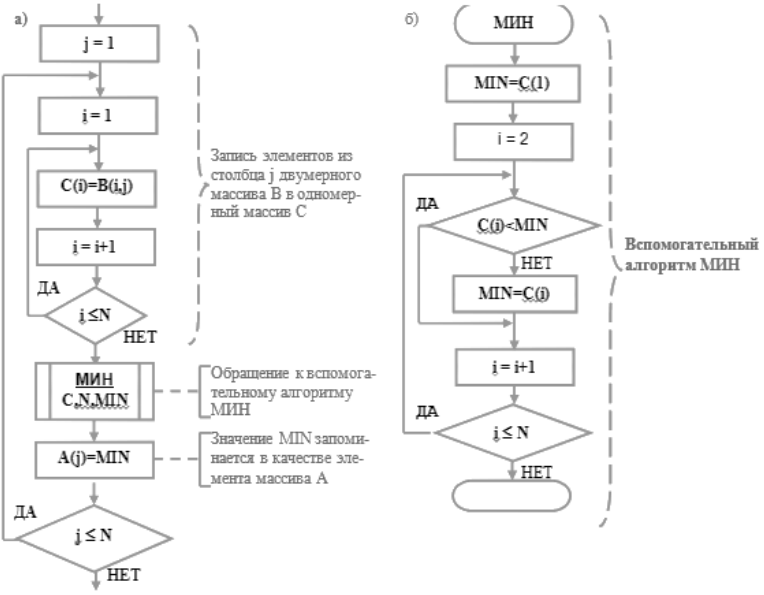
2.1.2 Разработка алгоритмов методом пошаговой детализации

Для специалистов, решающих с помощью ЭВМ те или иные задачи, разработка алгоритма решения является важнейшим делом. Существуют различные методы разработки алгоритмов (и программ), но наиболее важным является метод пошаговой детализации (или метод разработки «сверху – вниз»). При этом методе первоначально продумывается и фиксируется множество данных и результатов алгоритма без детальной проработки отдельных его частей. Тогда блок-схема любого алгоритма выглядит так, как показано на рисунке 2.3 (пример задачи: «Дан массив оценок успеваемости группы студентов по предмету O(i), i = l, 2, …, N. Определить количество студентов, оценка которых выше среднего балла группы»).



а) – основной алгоритм; б) – вспомогательный алгоритм MIN

Рисунок 2.1 - Фрагмент блок-схемы алгоритма формирования одномерного массива из минимальных элементов столбцов двумерного массива



а) – основной алгоритм; б) – вспомогательный алгоритм МИН

Рисунок 2.2 - Фрагмент блок-схем алгоритма формирования одномерного массива из минимальных элементов столбцов двумерного массива с использованием буферного массива

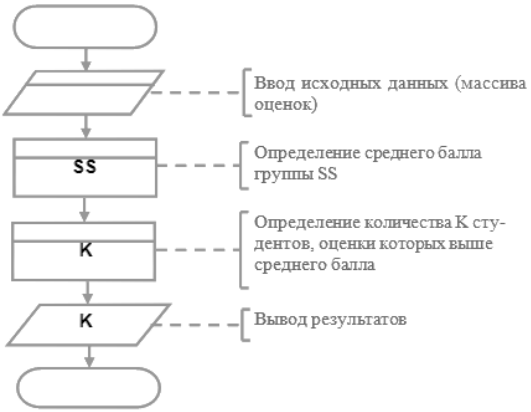


Рисунок 2.3 - Блок-схема алгоритма определения количества студентов, оценки которых выше среднего балла группы (I этап детализации)

Задачу разбивают на автономные части, каждая из которых существенно проще. Может оказаться необходимым повторять процесс детализации многократно, но это определяется только сложностью решаемых задач. Участки блок-схем, требующие дальнейшей детализации, обозначаются блоками «детализируемая программа», целиком заимствуемые части алгоритма обозначаются блоками «предопределенный процесс».

Конечным уровнем детализации алгоритма можно считать такой, при котором в алгоритме нет действий более крупных, чем: обращение к библиотечному (вспомогательному) алгоритму; вычисление арифметического выражения и присваивание значения переменной; сравнение арифметических выражений (или переменных); ввод (вывод) данных и т. п.

После первого этапа составления алгоритма получается перечень относительно крупных задач, каждая из которых (кроме последней) подлежит детализации. Блок-схема алгоритма (второй этап детализации) приведена на рисунке 2.4.

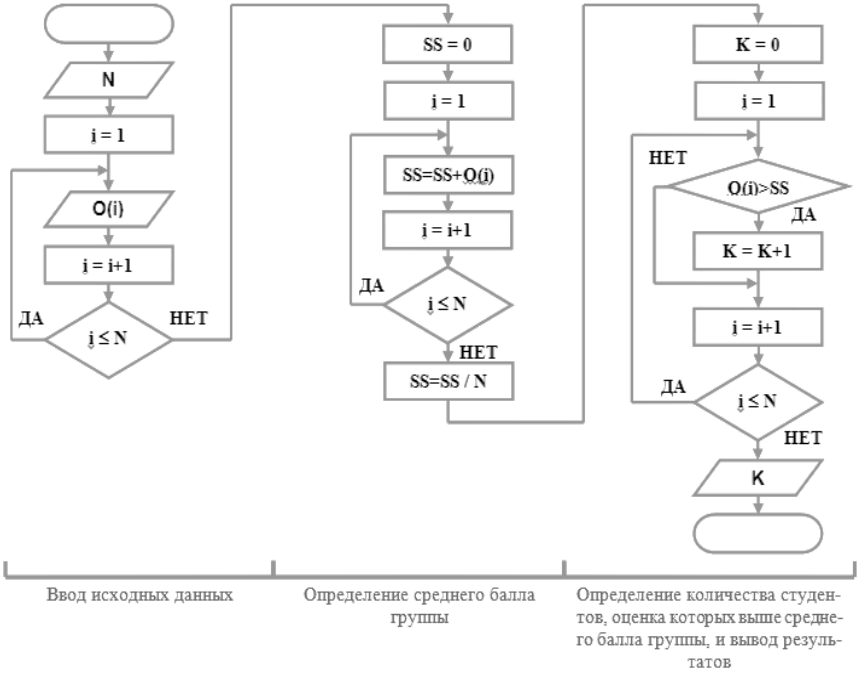


Рисунок 2.4 - Блок-схема алгоритма определения количества студентов, оценки которых выше среднего балла группы
(II этап детализации, структурированный алгоритм)

2.2 Порядок выполнения работы

2.2.1 Задания для выполнения работы

1. Включить компьютер.
2. Запустить на выполнение программу «Word».
3. Для массивов случайных чисел, приведенных в варианте задания проследить за изменениями в массиве при выполнении соответствующего типового алгоритма.
4. Написать алгоритмы, согласно варианту задания.

2.2.2 Порядок выполнения

1. Для массивов случайных чисел, приведенных в варианте задания, записать типовые алгоритмы и проследить за изменениями, как в самом массиве, так и в других переменных при выполнении соответствующего типового алгоритма.
2. Написать алгоритм 1 предложенной задачи по обработке численной информации в построчной записи и в виде блок-схемы.
3. Написать алгоритм 2 предложенной задачи по обработке численной информации в построчной записи и в виде блок-схемы.
4. Написать алгоритм 3 предложенной задачи по обработке текстовой информации в построчной записи и в виде блок-схемы.

2.2.3 Варианты заданий

2.2.3.1 Алгоритм 1

В следующих задачах переменные X, Y, Z предполагаются описанными как тип данных, определяемый пользователем:

```
Public Type ComplexNumber
RealPart As Double ' действ. часть комплексного числа
ImaginPart As Double  ' мнимая часть комплексного числа
End Type.
```

Вычисление функций комплексных переменных:

Вариант № 1. X, Y и Z: ln(Z) и X^Y.

Вариант № 2. Z: e^Z и arccos(Z).

Вариант № 3. Z и X: $\sqrt{Z}$ и sec(X).

Вариант № 4. Z и X: arcsin(Z) и cosec(X).

Вариант № 5. Z: sin(Z) и ch(Z).

Вариант № 6. Z: arctg(Z) и arcsec(Z).

Вариант № 7. Z (X – компл. осн. логарифма): arcctg(Z) и $Log_X(Z)$.

Вариант № 8. Z: cos(Z) и sh(Z).

Вариант № 9. Z: ctg(Z) и cth(Z).

Вариант № 10. Z: arcsh(Z) и arccosec(Z).

Вариант № 11. Z: tg(Z) и th(Z)

Вариант № 12. Z: arcch(Z) и sech(Z).

Вариант № 13. Z: arcth(Z) и cosech(Z).

Вариант № 14. Z: arccth(Z) и arcsh(Z).
Вариант № 15. X и Z: arcctg(Z) – arcsin(X).

2.2.3.2 Алгоритм 1

В следующих задачах переменные x, y, z предполагаются описанными как массивы целых чисел с индексами от 1 до n (n - некоторое натуральное число, большее 0), если иное не оговорено явно.

Размерность массива n и границы диапазона случайных чисел [a, b] должны вводиться. При вводе данных предусмотреть проверку на возможность ошибочного ввода данных (должно быть a < b).

Вариант № 1. Создать массив x: массив целых чисел с индексами от m до n, причём m < n – слу-чайные числа в диапазоне от a до b, а xm<=xm+1<= ... <=xn и находятся в интервале от c до d. Подсчитать количество чисел в массиве x, лежащих в диапазоне от k до f.

Вариант № 2. Создать массив x: массив целых чисел с индексами от m до n, причём m < n – слу-чайные числа в диапазоне от a до b, а xm<=xm+1<= ... <=xn в диапазоне от c до d. Подсчитать количество положительных чисел в массиве x.

Вариант № 3. Создать массив x: массив целых чисел с индексами от m до n, причём m < n – слу-чайные числа в диапазоне от a до b, а xm<=xm+1<= ... <=xn в диапазоне от c до d. Подсчитать количество отрицательных чисел в массиве x.

Вариант № 4. Создать массив x: массив целых чисел с индексами от m до n, причём m < n – слу-чайные числа в диапазоне от a до b и заполнить его случайными целыми числами из диапазона от a до b. Найти максимум из x[1]..x[n].

Вариант № 5. Создать массив x: массив целых чисел с индексами от m до n, причём m < n – случайные числа в диапазоне от a до b, а xm<=xm+1<= ... <=xn в диапазоне от c до d. Найти количество различных чисел среди элементов этого массива.

Вариант № 6. Создать массив x: массив целых чисел с индексами от 1 до n, причём x1<=x2<= ... <=xn в диапазоне от a до b. Подсчитать количество отрицательных чисел в массиве x

Вариант № 7. Создать массив x: массив целых чисел с индексами от m до n, причём m < n – слу-чайные числа в диапазоне от a до b, а xm<=xm+1<= ... <=xn в диапазоне от c до d. Подсчитать количество положительных чисел в массиве x.

Вариант № 8. Создать массив x случайных целых чисел из диапазона от a до b с индексами от 1 до n (n вводится). Найти максимальный элемент массива: x1..xn.

Вариант № 9. Создать массив x: массив целых чисел с индексами от m до n, причём xm<=xm+1<= ... <=xn в диапазоне от a до b. Найти количество различных чисел среди элементов этого массива.

Вариант № 10. Создать массив x: массив целых чисел с индексами от m до n, причём xm>=xm+1>= ... >=xn в диапазоне от a до b. Подсчитать количество положительных чисел в массиве x.

Вариант № 11. Создать массив x случайных целых чисел из диапазона от a до b. Подсчитать количество чисел в массиве x, лежащих в диапазоне от c до d.

Вариант № 12. Создать массив x, размерность которого является случайным числом. Заполнить элементы массива x[i] случайными действительными числами, из диапазона от c до d, (c и d вводятся).

Вариант № 13. Создать массив x: массив целых чисел с индексами от m до n, причём m, n – случайные числа в диапазоне от a до b. Заполнить его случайными числами из диапазона [-c, c](a, b и c вводятся). Подсчитать количество отрицательных чисел в массиве x.

Вариант № 14. Создать массив y массив целых чисел с индексами от 1 до n, причём y[1]<=y[2]<= ... <=y[n] в диапазоне от a до b. Не используя оператора присваивания для массивов, составить фрагмент программы, эквивалентный оператору x := y.

Вариант № 15. Создать массив x: массив целых чисел с индексами от m до n, причём m < n – случайные числа в диапазоне от a до b. Заполнить его случайными числами из диапазона [-c, c]. Подсчитать количество нулей в массиве x.

2.2.3.3 Алгоритм 1

В следующих задачах переменные a, b, c, d() предполагаются описанными как текстовые переменные. Текст описан как массив строк d(N). Для поиска комбинации символов можно использовать предопределенный процесс (функцию): InStr(I, strInput, strSearch). I – номер символа, с которого начинается в строке strInput поиск строки strSearch. Результат работы функции – N – № символа вхождения строки strSearch в строку strInput. Если такой подстроки нет, то функция дает 0.

Вариант № 1. Во всем тексте найти комбинацию символов, заданную в переменной a, и заменить ее словом, заданным в переменной b. Подсчитать количество сделанных замен.

Вариант № 2. В выделенном в тексте слове произвести расшифровку кодов ASCII всех букв. Добавить к коду каждой буквы один бит. Перевести полученные числа в литеры. Вывести все результаты на разных элементах управления.

Вариант № 3. В выделенном элементе строкового массива выделить цепочку символов, заданную в переменной a и заканчивая символом, заданным в переменной b. Длина цепочки должна быть не короче m символов. Подсчитать количество сделанных замен

Вариант № 4. В выделенном в тексте слове произвести расшифровку кодов ASCII. Ответ представить в двоичной и шестнадцатиричной системах счисления.

Вариант № 5. В каждом элементе строкового массива выделить цепочку символов, начиная с первого, заданного в переменной a и заканчивая символом, заданным в переменной b. Длина цепочки должна быть не короче m символов.

Вариант № 6. Во всем тексте найти слово, заданное в переменной a, и заменить его словом, заданным в переменной b. Подсчитать количество сделанных замен.

Вариант № 7. Во всем тексте найти последовательность символов, заданную в переменной a, и заменить ее символами, заданными в переменной b. Подсчитать количество сделанных замен.

Вариант № 8. Зашифровать текст, добавляя к коду каждого символа m байт.

Вариант № 9. В каждом элементе строкового массива заменить последние n знаков на первые m ($m \neq n$). Проверить различные соотношения между m и n.

Вариант № 10. В выделенной части текста (несколько строк) найти последовательность символов, заданную в переменной a, и заменить ее символами, заданными в переменной b. Подсчитать количество сделанных замен.

Вариант № 11. Зашифровать текст, добавляя m бит к коду каждого символа.

Вариант № 12. В выделенном в тексте слове произвести замену русских букв на соответствующие латинские.

Вариант № 13. В выделенном в тексте слове произвести замену строчных букв на заглавные и наоборот.

Вариант № 14. Подсчитать количество вхождений гласных букв в текст.

Вариант № 15. Подсчитать количество символов в текстовом массиве.

2.2.4 Содержание отчета

В отчете должны быть представлены:

1. Цель работы
2. Основные сведения о теме работы
3. Алгоритмы соответствующих частей разработанного проекта.
4. Перечень форм и элементов управления на них.
5. Процедуры обработки данных.
6. Исходные данные и результаты их обработки для каждого из заданий.

2.3 Контрольные вопросы

1. Для чего используются типовые алгоритмы?
2. Перечислите известные Вам типовые алгоритмы.
3. Что такое вспомогательные алгоритмы?

4. Дайте определение фактическим параметрам процедур и функций.
5. Дайте определение формальным параметрам процедур и функций.
6. Для чего используются библиотеки алгоритмов?
7. В чем состоит сущность метода разработки алгоритмов методом пошаговой детализации?
8. Что такое структурное программирование «сверху – вниз»?

ЛАБОРАТОРНАЯ РОБОТА 3

ОЗНАКОМЛЕНИЕ С РАБОЧЕЙ СРЕДОЙ VISUAL BASIC. ПЕРВЫЙ ПРОЕКТ

Цель работы – Научиться работать с оболочкой Visual Basic. Выучить назначение отдельных пунктов меню, панелей инструментов и элементов, различных окон. Разработать первый проект на Visual Basic 6.

3.1 Общие сведения

3.1.1 Идея Visual Basic

Название Visual Basic говорит само за себя. Если вы уже работали с другими, традиционными системами программирования, то вскоре убедитесь, что Visual Basic представляет совершенно другой стиль программирования. Уже по слову "Visual" можно догадаться, что в Visual Basic реализован визуальный стиль программирования. Как уже говорилось выше, вы даже не программируете, а проектируете приложение. Ваша первая задача при этом — создать рабочую среду, прежде чем начать набирать первую строку кода.

Слово Basic в названии указывает лишь на то, что синтаксис программ и операторы опираются на язык высокого уровня Basic (Beginners All-purpose Symbolic Instruction Code).Но если вы знаете обычный Basic, то очень скоро убедитесь, что Visual Basic заметно от него отличается.

Компилятор или интерпретатор? На вопрос, чем является Visual Basic - компилятором или интерпретатором, можно получить ответ: "И тем, и другим". Его нельзя однозначно отнести ни к компиляторам, ни к интерпретаторам.

Основным признаком интерпретатора является то, что созданные в нем программы выполняются только в среде разработки. Программу можно запустить непосредственно из среды и если в ней есть ошибки, они сразу же распознаются. Все это наблюдается и в Visual Basic, где можно запустить приложение непосредственно в среде разработки. При этом Visual Basic использует технологию Threaded p Code, при которой каждая введенная строка кода преобразуется в промежуточный код - Threaded p Code. Это еще не совсем машинный код, но такой код выполняется быстрее, чем при работе с обычным интерпретатором. Во-первых, Visual Basic сразу же проверяет синтаксис программы и выдает сообщение об обнаруженной ошибке. Другим преимуществом этой технологии является возможность поиска ошибок.

Однако Visual Basic - не просто интерпретатор, так как это означало бы, что приложения выполняются только в среде Visual Basic. Visual Basic предоставляет возможность создавать и исполняемые EXE файлы, поэтому его можно отнести и к компиляторам.

Visual Basic нельзя назвать чистым компилятором, так как в отличие, например, от Visual C++, Visual Basic не создает исполняемый файл сразу же при запуске из среды разработки. Для создания такого файла необходимо сделать это явно (команда File \Make ***.EXE).Начиная с пятой версии, Visual Basic располагает так называемым "Native Compiler", т.е. компилятором, который может создавать машинный код. Таким образом, Visual Basic объединяет в себе возможности, как интерпретатора, так и компилятора. И это имеет больше преимуществ, чем недостатков.

3.1.2 Рабочая среда Visual Basic

Запустить Visual Basic можно с помощью команды меню или двойным щелчком на пиктограмме про-граммы.

После запуска Visual Basic на экране появляется диалоговое окно, приведенное на рис.3.1, и в котором вы можете выбрать тип создаваемого приложения. Из этого же окна можно загрузить уже существующий проект.

Рисунок 3.1 – Стартовое диалоговое окно среды Visual Basic

За некоторыми пиктограммами диалогового окна скрываются мастера (Wizards), сопровождающие разработчика при создании приложений и берущие на себя часть его работы, например подключение базы данных или создание формы. Один из основных мастеров - мастер приложения Visual Basic, с помощью которого можно создать основной "каркас" для обычных Windows приложений.

В процессе работы мастера создается почти готовое приложение с различными формами, соответствующей рабочей средой, меню, панелью инструментов и т.п. Это приложение можно потом совершенствовать и настраивать.

Рабочая среда Visual Basic 6 заметно отличается от среды предыдущих версий. Многие считают именно среду разработки с ее многообразием различных окон одним из самых слабых звеньев Visual Basic. Однако, освоив среду, вы будете чувствовать себя в ней достаточно комфортно.

Главное окно среды разработчика содержит несколько окон. Все окна подчиняются главному окну Visual Basic, и могут "прикрепляться" (dockable) к одному из его краев. При необходимости такой многодокументный (MDI — Multiple Document Interface) вид среды разработки можно настроить. Для этого следует установить флажок SDI Development Environment вкладки Advanced диалогового окна Options, которое открывается с помощью одноименной команды меню Tools.

3.1.2.1 Панель инструментов Visual Basic

В верхней части экрана находится центр управления Visual Basic - меню и панель инструментов (Toolbar), приведенная на рис.3.2. Ее можно настраивать, как это обычно делается в приложениях Microsoft.

Рисунок 3.2 – Меню и панель инструментов рабочей среды Visual Basic 6

3.1.2.2 Панель элементов

Кнопки, поля ввода и другие элементы управления, которые нужны для создания приложения, расположены на панели элементов (Toolbox) приведенной на рис.3.3. Для выбора элемента управления (Control) нужно щелкнуть на нем и затем с помощью мыши установить в форме его размер и позицию. После двойного щелчка на пиктограмме элемента в центре формы появляется соответствующий элемент стандартного размера.

Рисунок 3.3 – Панель элементов

3.1.2.3 Окно формы

Окно формы, приведенное на рис. 3.4, часто называемое просто "форма", является главным элементом приложения. Форма представляет собой контейнер для элементов управления. Точки сетки на форме только помогают размещению элементов и при работе приложения не видны. При запуске Visual Basic открывающаяся на экране форма не содержит элементов управления. После щелчка на пиктограмме требуемого элемента управления курсор мыши принимает форму крестика. Теперь нужно указать в форме начальный угол элемента управления, нажать левую кнопку мыши и, не отпуская

ее, установить размер элемента. После достижения нужного размера кнопка отпускается и в форме появляется выбранный элемент управления.

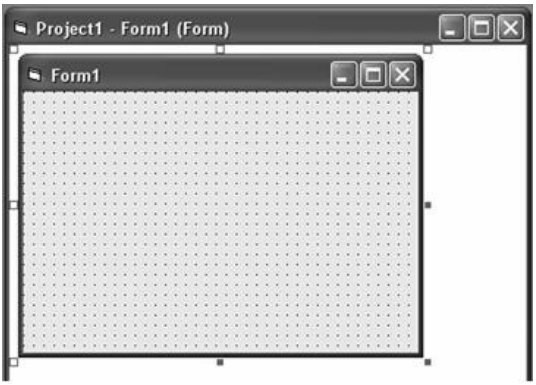


Рисунок 3.4 – Окно формы

3.1.2.4 Окно свойств — Properties

В этом окне (рис. 3.5) задаются свойства выбранного элемента управления. В строке заголовка окна свойств рядом с текстом Properties указывается имя формы, которой принадлежит элемент управления. Поле со списком под строкой заголовка позволяет выбрать требуемый элемент управления. В списке, расположенном ниже, перечислены свойства этого элемента (в алфавитном порядке либо по категориям). Набор свойств зависит от типа элемента управления.

Рисунок 3.5 – Окно свойств

Список свойств состоит из двух столбцов: в правом перечислены названия свойств, а в левом — их значения. Редактирование свойства

осуществляется либо вручную (например, ввод имени элемента), либо выбором соответствующего поля из списка, либо при помощи диалогового окна настройки свойства. Краткое описание выбранного свойства отображается в нижней части окна.

3.1.2.5 Окно проекта

В окне проекта (рис. 3.6) отображаются все элементы приложения: формы, модули, классы и т.п., сгруппированные по категориям. В Visual Basic все разрабатываемые приложения называются проектами. Проект содержит несколько групп компонентов (формы, модули и т.п.). Все приложения Visual Basic строятся по модульному принципу, поэтому и объектный код состоит не из одного большого файла, а из нескольких частей. Несколько приложений также могут объединяться в группы.

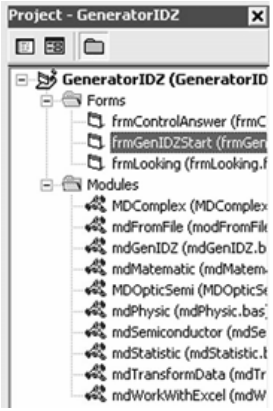


Рисунок 3.6 – Окно проэкта

Чтобы сохранить существующий элемент (форму, модуль и др.), нужно выделить его в списке окна про-екта и выбрать команду File \Save или File \Save As. Для записи всего проекта (включая все компоненты) выберите команду File \Save Project или File \Save Project As (для сохранения проекта под другим именем).

Чтобы добавить в проект новый элемент, необходимо вызвать команду Project \Add. Для удаления элемента нужно выделить его в окне проекта и затем выбрать команду меню Project \Remove.

Все эти элементы сохраняются как отдельные и независимые файлы. Поэтому их можно в любое время загружать и сохранять. Это дает возможность использовать в проекте формы и коды, созданные для других проектов, что экономит рабочее время.

Содержимое окна проекта сохраняется в специальном файле. Он имеет расширение VBP и содержит список элементов, которые нужно

загрузить в среду разработки. Если несколько проектов объединяются в группу, их имена сохраняются в файле с расширением VBG.

3.1.2.6 Окно кода

Сразу после запуска Visual Basic это окно не отображается. Тем не менее, окно кода (рис. 3.7) едва ли не самое важное в Visual Basic, так как именно в нем вводится программный код. Код в Visual Basic разделяется на процедуры и, как правило, непосредственно связан с определенными элементами управления. Это позволяет открыть окно кода двойным щелчком на элементе управления в форме.

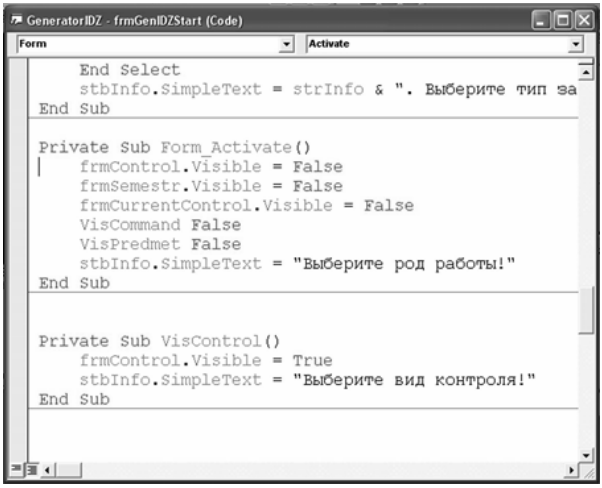


Рисунок 3.7 – Окно кода

И еще несколько слов о редакторе кода Visual Basic. В принципе приемы работы в нем такие же, как и при редактировании текстов в приложениях Windows. Набранные символы вставляются на месте курсора ввода. После нажатия клавиши [Insert] текстовый курсор принимает форму бруска, что свидетельствует об активизации режима замены. Повторное нажатие клавиши [Insert] переводит редактор обратно в режим вставки. Выделенный текст при вводе заменяется новым.

Комбинация клавиш [Ctrl+X] удаляет выделенный текст и помещает его в буфер обмена Windows. Клавиши [Ctrl+C] служат для копирования текста в буфер обмена, а [Ctrl+V] — для вставки содержимого буфера обмена. Кроме того, комби-нация клавиш [Ctrl+Y] помещает в буфер обмена строку, в которой находится текстовой курсор. Комбинация [Ctrl+N] вставляет перед текущей строкой пустую строку. Отдельные процедуры можно просматривать с помощью комбинаций клавиш [Ctrl+T] или

[Ctrl+4]. Клавишей [Tab] создается отступ в строке или во всех выделенных строках. С помощью [Shift+Tab] текст сдвигается влево.

Над вертикальной полосой прокрутки находится маленькое поле, которое можно перетаскивать с помощью мыши вниз для разделения окна на две части (split window). Это дает возможность редактировать в одном окне две разные процедуры. Разделение отменяется, если разделительную линию переместить к самому краю окна или выполнить двойной щелчок на разделительной линии.

Вид и размер шрифта можно изменять на вкладке Editor Format диалогового окна Options (команда меню Tools \Options).С помощью поля Tab Width вкладки Editor устанавливается число символов для отступа клавишей [Tab]. При установленном флажке Auto Indent нажатие клавиши [Enter] помещает курсор ввода в колонку, с которой начиналась предыдущая строка. Выбор Default to Full Module View позволяет просматривать в окне кода несколько процедур формы. Установить и отменить этот режим можно при помощи кнопок, расположенных слева от горизонтальной полосы прокрутки окна кода.

3.1.2.7 Управление раскладкой

Обычной проблемой разработки приложений в Visual Basic являются различные разрешения экрана монитора разработчика и пользователей. В основном приложения разрабатывают в среде, где мониторы имеют высокое разрешение экранов. Это может вызывать неудобства для пользователя, имеющего монитор с низким разрешением. В окне управления раскладкой можно устанавливать размер и позицию формы на экране, отобразив вспомогательные линии для различных расширений.

3.1.3 Событийно управляемое программирование

Ориентирование на события — это стержень создания Windows приложений в Visual Basic. При этом разработка выполняется не только в Windows, но и как Windows.

Microsoft Windows — это система, базирующаяся на сообщениях и событиях. Это значит, что каждое действие в Windows вызывает событие, которое в виде сообщения передается в приложение. Приложение анализирует сообщение и выполняет соответствующие действия.

Основанием для подобной обработки действий служит сама концепция Windows. В среде Windows пользователь может работать одновременно с несколькими приложениями. Например, он может редактировать текст в текстовом редакторе и, переключаясь в программу обработки электронных таблиц, выполнять некоторые расчеты. Поэтому ни одно приложение не может функционировать само по себе, не взаимодействуя с другими приложениями и с операционной системой. Вышестоящая инстанция должна ему сообщить, что происходит, и только тогда приложение реагирует на это.

Допустим, что в Windows выполняются два приложения. Ни одно из них не может просто перехватить инициативу и среагировать на нажатие клавиш, так как это событие может быть предназначено другому приложению. Поэтому Windows сначала воспринимает событие, вызванное нажатием клавиш, а затем решает, кому передать обработку этого события. Затем нажатие клавиши в виде сообщения посылается приложению. Приложение обрабатывает событие, связанное с клавиатурой, и анализирует клавиатурный ввод, отображая его, например, в активном элементе управления.

Разумеется, такое представление является слишком упрощенным. Обычно при обработке нажатия клавиш вызывается значительно больше сообщений и событий.

Приложения, созданные с помощью Visual Basic, также работают по этому принципу. Но при этом система Visual Basic берет на себя часть работы. Она перехватывает сообщение и передает его соответствующему объекту (например, кнопке), где затем вызывает соответствующее событие (например, событие Click).

3.1.3.1 События

Для выполнения программного кода всегда необходимо событие. Это одно из важнейших правил создания приложений в Visual Basic. Ни один код не выполняется без события.

Если вы уже разрабатывали приложения в других системах программирования, основанных на линейном принципе построения программ, то здесь вам придется перестроиться. В программах, управляемых событиями, нет сплошного кода, который выполняется от начала до конца. Это значит, что после запуска программы у пользователя больше нет четко определенного пути. Он может в любое время нажать какую-нибудь кнопку в приложении, может выполнить ввод текста в поле, может в любое время прекратить обработку и вызвать другую программу.

Windows и Visual Basic предоставляют ряд различных событий. Приложению следует подождать, пока одно из этих событий не наступит, и только потом выполнять код.

3.1.3.2 Объекты

Объекты являются причиной использования в Visual Basic понятий из объектно-ориентированного программирования, хотя сам по себе Visual Basic не является объектно-ориентированным языком.

Многое, или почти все из того, с чем вы работаете в Visual Basic, является объектами. Так, командная кнопка – это объект. Линия в форме - это также объект. Объектами являются команды меню, принтер, базы данных и т.д.

Объекты на панели элементов называются элементами управления (Control). Работая с ними, пользователь инициирует определенные события, и в результате может управлять программой.

Каждый объект характеризуется определенными параметрами, которые можно разделить на три категории:

- события;
- методы;
- свойства.

3.1.3.3 Классы

Объекты объединяются в классы. К одному классу принадлежат объекты с одинаковым набором свойств, методов и событий.

События связаны с определенными действиями пользователя и могут вызывать код Visual Basic. Методы — это рабочие операторы объекта. Например, метод Move позволяет переместить элемент управления в заданную позицию. Свойства отвечают за внешний вид и поведение объекта. Например, свойство Caption определяет текст надписи на объекте.

3.1.3.4 Методы и свойства

Границы между свойствами и методами расплывчаты. Есть, например, метод Move, который изменяет позицию объекта. Но есть и некоторые свойства (Top, Left), выполняющие аналогичные действия. Основное различие между методами и свойствами заключается в том, что со свойствами можно работать как во время разработки проекта, так и во время выполнения приложения, тогда как методы доступны только при выполнении. Следует заметить, что некоторые свойства могут быть также недоступны при разработке приложения, а во время его работы доступны только для чтения.

Список свойств, которые разработчик может изменять при разработке приложения, отображается в окне свойств элемента управления.

Для отображения окна свойств следует нажать клавишу [F4], или соответствующую кнопку на панели инструментов либо выбрать команду Properties Window меню View.

В списке под строкой заголовка окна свойств, а также и из самого заголовка видно, свойства какого объекта отображены в настоящий момент. Обычно в окне свойств отображен выделенный элемент управления. Если не выделен ни один элемент управления, то отображаются свойства активной формы.

При изменении свойств объекта всегда следует обращать внимание на то, какой объект управления выделен.

Давайте детальнее рассмотрим командную кнопку. В верхней области окна свойств найдите свойство Name. Это свойство - одно из основных свойств большинства объектов. В коде программы объект идентифицируется по имени, которое указывается в этом поле. По умолчанию первому объекту Command Button Visual Basic присваивает имя Command1, которое и отображено в поле значения свойства Name. Оно появляется и в первой строке процедуры события:

```
Private Sub Command1_Click()
```

Если теперь изменить имя кнопки, этот код будет недействительным, так как объекта с именем Command1 больше нет!

Имя каждого объекта следует изменять до написания кода для этого элемента, так как это предотвращает неприятности и путаницу со стандартным именем. Следует также использовать информативные имена, например AddRecord. При вставке нескольких кнопок по умолчанию каждой последующей кнопке Visual Basic присваивал бы имена Command2, Command3 и т.д.

В таблице 3.1 представлены свойства кнопки (CommandButton). Некоторыми свойствами, например Font или Height и Width, обладают и другие объекты. Но в основном каждый объект имеет свой специфический набор свойств. Для их просмотра нужно выбрать имя объекта в первой строке окна свойств или выделить его в форме.

Таблица 3.1 - Некоторые свойства кнопок

Свойство	Значение
Appearance	Трехмерный эффект
BackColor	Цвет фона
Cancel	Кнопка включается нажатием клавиши [Esc]
Caption	Надпись
Default	Кнопка включается нажатием клавиши [Enter]
Draglcon	Пиктограмма при перемещении
DragMode	Ручной или автоматический режим перетаскивания (Drag&Drop)
Enabled	Доступность элемента
Font	Вид шрифта
Height	Высота объекта
HelpContextId	Привязка к собственному справочному файлу
Index	Индекс элемента управления в массиве
Left	Левый верхний угол, координата X
Mouselcon	Изображение указателя мыши на кнопке
MousePointer	Форма указателя мыши на кнопке
Name	Имя элемента управления
Tablndex	Порядок перемещения фокуса при нажатии клавиши [Tab]
TabStop	Возможность перехода к элементу управления с помощью клавиши [Tab] (True/False)
Tag	Содержит любую необходимую дополнительную информацию
Top	Левый верхний угол, координата Y
Visible	Видимость (True/False)
WhatsThisHelpID	Привязка к собственному справочному файлу
Width	Ширина объекта

Значения одних свойств могут быть произвольными и вводятся с клавиатуры (например, свойство Caption), значения других - фиксированы и выбираются из списка значений, (например, True или False для свойства Visible), третьи могут устанавливаться из дополнительного диалогового окна (например, Font).

Для быстрого изменения свойства достаточно дважды щелкнуть на его имени. Если значения свойства можно установить с помощью клавиатуры, текст свойства выделяется. Если значения свойства выбираются из множества фиксированных значений, то очередное значение выделяется в списке свойств или отображается в диалоговом окне установки значения свойства.

После двойного щелчка на кнопке открывается окно кода. В верхней части окна слева расположено поле со списком (Object), а справа - (Procedure). Их расположение подчеркивает, что каждое событие всегда связано с объектами. Имя процедуры составляется из имени объекта и имени события, разделенных символом подчеркивания. Подобным образом осуществляется доступ к свойствам объекта, но в качестве разделителя ставится точка:

```
Command1.Caption = "Hello world"
```

Таким же образом получают доступ и к методам:

```
Command1.Move 120,250
```

3.1.4 Каталог объектов

Получить информацию о некотором объекте можно также из каталога объектов, который содержит список всех объектов Visual Basic, сгруппированный по категориям. Эти категории называются библиотеками объектов.

Каталог объектов можно открыть при помощи кнопки на панели инструментов или команды меню View \Object Browser. В верхнем поле списка выберите необходимую библиотеку либо объекты из всех библиотек (стандартные элементы управления содержит библиотека VB (Visual Basic Objects and Procedures)). В списке Classes перечислены все объекты Visual Basic. После выбора объекта в списке Members of <Имя класса> выводятся все относящиеся к нему свойства и методы. При этом в нижней части диалогового окна дается краткое описание свойства или метода. Нажав клавишу [F1] либо кнопку со знаком вопроса, можно вызвать справоч-ную информацию с подробным пояснением ключевого слова. Свойство или метод можно скопировать в буфер обмена, а затем вставить в окне кода.

3.2 Порядок выполнения работы

3.2.1 Задания для выполнения работы

1. Включить компьютер.
2. Запустить на выполнение программу «**Visual Basic** 6.0».
3. Открыть проект **Standard.EXE**.
4. Проверить работу пунктов меню: **Файл, Правка, Вид, Проект**.
5. Проверить работу отдельных кнопок панели инструментов.
6. Проверить работу окна **Project (Проект).**
7. Проверить работу окна **Properties (Свойства).**
8. Разработать проект на **Visual Basic** 6.0
9. Проверить работу окна «План формы», изменяя свойства формы **Top, Height, Left и Width.**
10. Проверить работу панели элементов: Разместить на форме несколько элементов управления.
11. Написать простейшие коды обработки событий

3.2.2 Порядок выполнения

1. На стадии проектирования поместить на форму необходимые для выполнения задания элементы управления. Установить различные графические свойства различным элементам управления. Все элементы управления, кроме необходимых, для начала работы, – невидимы.

2. Ввести текст в текстовое окно (если это предусмотрено заданием).

3. При нажатии кнопки «Модификация» выполнить действия, согласно варианту задания.

4. Предусмотреть выход из проекта с помощью пункта меню и/или кнопки на форме.

3.2.3 Варианты заданий

Вариант № 1. Перенести текст из текстового окна на метку. Сделать невидимой кнопку «Модификация» и видимой кнопку «Выход». Сделать видимой метку и невидимым текстовое окно. Примечание: Используйте свойства элемента управления Visible, Text и Caption.

Вариант № 2. Сделать невидимой кнопку «Модификация» и видимыми кнопки «Выход» и «Цвет». При нажатии кнопки «Цвет» изменить цвет фона и переднего плана текстового окна. Примечание: Используйте свойства элемента управления Visible, ForeColor и BackColor.

Вариант № 3. Перенести текст из текстового окна на метку. Сделать невидимой кнопку «Модификация» и видимыми кнопки «Выход» и «Цвет». Сделать видимой метку и невидимым текстовое окно. При нажатии кнопки «Цвет» изменить цвет фона и переднего плана метки. Примечание: Используйте свойства элемента управления Visible, Text, Caption, ForeColor и BackColor.

Вариант № 4. Сделать невидимой кнопку «Модификация» и видимыми кнопки «Выход» и «Перемещение». При нажатии кнопки «Перемещение» изменить расположение текстового окна на форме. Примечание: Используйте свойства элемента управления Visible, Top и Left.

Вариант № 5. Перенести текст из текстового окна на метку. Сделать невидимой кнопку «Модификация» и видимыми кнопки «Выход» и «Перемещение». Сделать видимой метку и невидимым текстовое окно. При нажатии кнопки «Перемещение» изменить расположение метки на форме. Примечание: Используйте свойства элемента управления Visible, Text, Caption, Top и Left.

Вариант № 6. Сделать невидимой кнопку «Модификация» и видимыми кнопки «Выход» и «Размеры». При нажатии кнопки «Размеры» изменить размеры текстового окна и шрифта. Примечание: Используйте свойства элемента управления Visible, Height, Width и FontSize.

7. Перенести текст из текстового окна на метку. Сделать невидимой кнопку «Модификация» и видимыми кнопки «Выход» и «Размеры». Сделать видимой метку и невидимым текстовое окно. При нажатии кнопки «Размеры» изменить размеры метки и шрифта. Примечание: Используйте свойства элемента управления Visible, Text, Caption, Height, Width и FontSize.

Вариант № 8. Сделать невидимой кнопку «Модификация» и видимой кнопку «Выход» и «Движение». При нажатии кнопки «Движение» запустить таймер. При обработке события Timer изменить горизонтальное положение окна на 10 твипов. Прекратить движение текстового окна при достижении границы формы; остановить таймер; сделать видимой кнопку «Выход». Примечание: Используйте свойства элемента управления Visible, Left. Для запуска Timer используйте свойства Enabled, Interval. Для остановки движения проверьте соотношение между положением объекта (Left или Left + Width) на форме (размер формы 0 – слева и Width – справа).

Вариант № 9. Сделать невидимой кнопку «Модификация» и видимой кнопку «Выход» и «Движение». При нажатии кнопки «Движение» запустить таймер. При обработке события Timer изменить вертикальное положение окна на 10 твипов. Прекратить движение текстового окна при достижении границы формы; остановить таймер; сделать видимой кнопку «Выход». Примечание: Используйте свойства элемента управления Visible, Top. Для запуска Timer используйте свойства Enabled, Interval. Для остановки движения проверьте со-отношение между положением объекта (Top или Top + Height) на форме (размер формы 0 – сверх и Height – снизу).

Вариант № 10. Перенести текст из текстового окна на метку. Сделать невидимой кнопку «Модификация» и видимыми кнопки «Выход» и «Переход». При нажатии кнопки «Переход» запустить таймер. При обработке события Timer через 500 мс делать поочередно видимыми метку и

текстовое окно. Примечание: Используйте свойства элемента управления Visible, Text, Caption. Для запуска Timer используйте свойства Enabled, Interval.

Вариант № 11. Перенести текст из текстового окна на метку, а содержимое метки (свойство Caption) напечатать на форме. Сделать невидимой кнопку «Модификация» и видимой кнопку «Выход». Сделать видимой метку и невидимым текстовое окно. Изменить цвета объектов на форме. Примечание: Используйте свойства элемента управления Visible, Text, Caption, ForeColor и BackColor.

Вариант № 12. Сделать невидимой кнопку «Модификация» и видимыми кнопки «Выход» и «Цвет». При нажатии кнопки «Цвет» изменить цвет фона и переднего плана текстового окна и формы, причем цвет одного из них должен быть негативным по отношению к другому. Примечание: Используйте свойства элемента управления Visible, ForeColor и BackColor. При назначении цвета используйте функцию выбора цвета RGB(Red, Green, Blue), где Red, Green, Blue – целые из диапазона 0 – 255. Негативный цвет получается, если ко всем цветам применить операцию: негативный цвет = 255 – прямой цвет.

Вариант № 13. Перенести текст из текстового окна на метку. Сделать невидимой кнопку «Модификация» и видимыми кнопки «Выход» и «Цвет». Сделать видимой метку и невидимым текстовое окно. При нажатии кнопки «Цвет» изменить цвет фона и переднего плана метки и текстового окна, причем цвет одного из них должен быть негативным по отношению к другому. Примечание: Используйте свойства элемента управления Visible, Text, Caption, ForeColor и BackColor. При назначении цвета используйте функцию выбора цвета RGB(Red, Green, Blue), где Red, Green, Blue – целые из диапазона 0 – 255. Негативный цвет получается, если ко всем цветам применить операцию: негативный цвет = 255 – прямой цвет.

Вариант № 14. Сделать невидимой кнопку «Модификация» и видимыми кнопки «Выход» и «Перемещение». При нажатии кнопки «Перемещение» изменить расположение текстового окна на форме, а также изменить цвет фона и переднего плана метки и текстового окна, причем цвет одного из них должен быть негативным по отношению к другому. Примечание: Используйте свойства элемента управления Visible, Top и Left, ForeColor и BackColor. При назначении цвета используйте функцию выбора цвета RGB(Red, Green, Blue), где Red, Green, Blue – целые из диапазона 0 – 255. Негативный цвет получается, если ко всем цветам применить операцию: негативный цвет = 255 – прямой цвет.

Вариант № 15. Перенести текст из текстового окна на метку. Сделать невидимой кнопку «Модификация» и видимыми кнопки «Выход» и «Перемещение». Сделать видимой метку и невидимым текстовое окно. При нажатии кнопки «Перемещение» изменить расположение метки на форме, а также изменить цвет фона и переднего плана метки и текстового

окна, причем цвет одного из них должен быть негативным по отношению к другому. Примечание: Используйте свойства элемента управления Visible, Text, Caption, Top и Left, ForeColor и BackColor. При назначении цвета используйте функцию выбора цвета RGB(Red, Green, Blue), где Red, Green, Blue – целые из диапазона 0 – 255. Негативный цвет получается, если ко всем цветам применить операцию: негативный цвет = 255 – прямой цвет.

3.2.4 Содержание отчета

В отчете должны быть представлены:

1. Цель работы.
2. Задание для выполнения работы.
3. Алгоритм работы первого проекта.
4. Рисунок разработанной формы с элементами управления.
5. Ответы на контрольные вопросы.

3.3 Контрольные вопросы

1. Описать отличие компилятора от интерпретатора.
2. Дать описание назначения главного меню, панелей инструментов и элементов, окон формы, свойств и проекта, а также каталога объектов.
3. Дать описание назначение окна кодов
4. Что такое событийно-управляемое программирование?
5. Дать определение объектам, с которыми работает Visual Basic.
6. В чем состоит отличие между свойствами, событиями и методами объектов Visual Basic?
7. Для чего предназначены элементы управления Visual Basic?
8. Перечислите основные свойства объектов, определяющие их размеры, положение, цвета.
9. В чем состоит отличие свойств Name и Caption?
10. В чем состоит отличие свойств Enabled и Visible?

СПИСОК ЛИТЕРАТУРЫ

1. Матросов В.Л. Теория алгоритмов. – М., 1989. – 188 с.
2. Мальцев А.И. Алгоритмы и рекурсивные функции. 2–е изд. - М., 1986. - 211 с.
3. Лавров И.А., Максимов Л.Л. Задачи по теории множеств, математической логики и теории алгоритмов. – М., 1984. – 224 с.
4. Сайлер Б., Споттс Дж. Использование Visual Basic 6. Классическое издание. - М.: Вильямс, 2007. – 832 с.
5. Сергеев В. Visual Basic 6.0. Наиболее полное руководство для профессиональной работы в среде Visual Basic 6.0. – СПб.: БХВ-Петербург, 2004. – 992 с.
6. Balena F. Programming Microsoft Visual Basic 6.0. – USA: Microsoft Press, 1999. – 1312 p.
7. Halvorson M. Microsoft Visual Basic Professional 6.0 Step by Step. – USA: Microsoft Press, 1998. – 672 p.
8. Microsoft Visual Basic 6. Шаг за шагом: Практ. пособ. / Пер. с англ. – М.: Изд. ЭКОМ., Изд. 2-е, исправленное, 2003. – 432 с.

СОДЕРЖАНИЕ

Учебное издание

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
к лабораторным работам
«Основы теории алгоритмов.
Ознакомление с рабочей средой Visual Basic»
из раздела «Основы теории алгоритмов.
Решения задач при помощи компьютера»
дисциплины «Информатика»
для студентов направления подготовки
6.050801 «Микро- и наноэлектроника»

Составители: ШКАЛЕТО Владимир Иванович
ЗАЙЦЕВ Роман Валентинович

Ответственный за выпуск М.В. Кириченко

Редактор

План 2013 р.

Підписано до друку _____. Формат 60×84 1/16. Папір друк. №2.

Друк – ризографія. Гарнітура Times New Roman. Ум. друк. арк. 2,5.

Обл.-вид. 3,0. Тираж 50 прим. Зам. № _____. Ціна договірна

Видавничий центр НТУ “ХПІ”. 61002, Харків, вул. Фрунзе, 21.
Свідоцтво про державну реєстрацію ДК № 116 від 10.07.2000 р.

Друкарня НТУ “ХПІ”. 61002, Харків, вул. Фрунзе, 21.