Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Оренбургский государственный университет»

Кафедра программного обеспечения вычислительной техники
и автоматизированных систем

В.В. Позевалкин, Л.А. Юркевская

ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ

Рекомендовано к изданию Редакционно-издательским советом федерального государственного бюджетного образовательного учреждения высшего образования «Оренбургский государственный университет» в качестве методических указаний для студентов, обучающихся по программам высшего образования по направлению подготовки 09.03.04 Программная инженерия

Оренбург
2016

УДК 004.42(076.5)
ББК 32.973-018я7
П47

Рецензент – профессор, доктор экономических наук, В.Н. Шепель

Позевалкин, В.В.
П47 Основы алгоритмизации и программирования: методические указания для выполнения контрольной работы / В.В. Позевалкин, Л.А. Юркевская; Оренбургский гос. ун.-т. – Оренбург: ОГУ, 2016. – 40 с.

Лабораторный практикум состоит из пяти лабораторных работ и посвящен изучению основ алгоритмизации и программирования на языке высокого уровня Турбо Паскаль. Каждая лабораторная работа содержит теоретическую часть с примером реализации представленной информации, а также варианты индивидуальных заданий для самостоятельного выполнения.

Методические указания предназначены для выполнения контрольной работы по дисциплине «Информатика и программирование» студентами по направлению подготовки 09.03.04 Программная инженерия.

УДК 004.42(076.5)
ББК 32.973-018я7

Содержание

Введение

Данные методические указания посвящены основам разработки алгоритмов решения поставленных задач, а также их программной реализации на языке высокого уровня Турбо Паскаль.

Цель методических указаний - научить студентов разрабатывать свои собственные алгоритмы, дать представление о структуре программы, а также ознакомить с основами алгоритмизации. В результате занятий студент должен научиться кодировать несложные алгоритмы, а также осуществлять их программную реализацию. Опыт программирования, приобретенный студентами при изучении данного курса, будет активно применяться при дальнейшем обучении.

Язык программирования — это формальная знаковая система, предназначенная для записи компьютерных программ и определяющая набор лексических, синтаксических и семантических правил, которые определяют внешний вид программы, а также действия выполняемые ЭВМ под её управлением.

В настоящее время существует большое количество языков программирования, используемых пользователями для решения задач различной сложности. В методических указаниях рассмотрены примеры создания программ на языке Турбо Паскаль. Содержатся материалы, необходимые для самостоятельной подготовки студентов к выполнению контрольной работы по дисциплине «Информатика и программирование». В описание лабораторных работ включены цель работы, порядок ее выполнения, рассмотрены теоретические вопросы, связанные с реализацией поставленных задач, приведены варианты заданий для самостоятельного выполнения. В приложениях представлен образец оформления отчета по контрольной работе.

1 Лабораторная работа №1. Алгоритмы разветвляющейся структуры

1.1 Цель работы

Приобретение навыков составления программ, в которых на определенном этапе осуществляется выбор очередного оператора в результате анализа некоторых условий.

1.2 Методические указания

При написании программ часто приходится сталкиваться с ситуацией, в которой некое действие должно быть выполнено только при определённом условии. В таких случаях применяют разветвляющиеся алгоритмы, которые в языках программирования представлены в виде условных операторов.

Ветвление в алгоритме означает организацию выбора одного из двух альтернативных вариантов продолжения алгоритма программы в соответствии с определенным логическим выражением. Каждый из альтернативных вариантов называется ветвью в алгоритме.

Для того чтобы рационально организовать ветвление в алгоритме необходимо определить количество n вариантов возможных способов вычисления искомой величины. Затем для каждого из n вариантов необходимо составить логическое выражение, истинное только для данного варианта. После этого необходимо организовать в алгоритме n ветвей с помощью (n–1) ветвлений, используя (n–1) логическое выражение.

В языке Турбо Паскаль существует два основных вида условных операторов оператор if — else для выбора из двух вариантов и оператор множественного выбора case для выбора из нескольких вариантов. Условный оператор осуществляет выполнение или невыполнение определенного оператора либо группы операторов.

1.2.1 Условный оператор **if**

Условный оператор **if** в общем виде выглядит так **if** (<логическое выражение>) <оператор1>[**else** <оператор2>]

Оператор1 выполняется если логическое выражение является истинным, а оператор2 (необязательный) – если ложным.

Пример – На рисунках 1.1 и 1.2 представлена блок-схема алгоритма и листинг программы для вычисления функции Y при заданном значении аргумента *x*:

$$Y = \begin{cases} |x|, \text{если } x \le -3 \\ \sin(x), \text{если } -3 < x \le 3 \\ 2x, \text{если } x > 3 \end{cases}$$

Блок-схема алгоритма

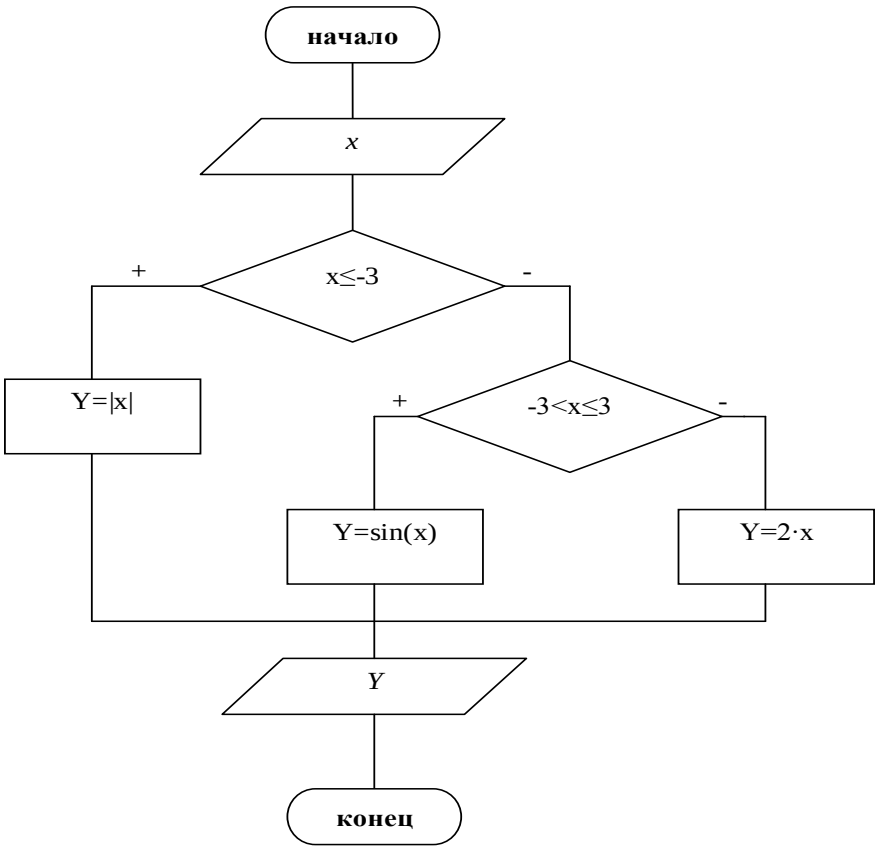


Рисунок 1.1 – Блок-схема алгоритма

Листинг программы

Рисунок 1.2 – Листинг программы

1.2.2 Оператор выбора **case**

Данный оператор позволяет сделать выбор из произвольного количества заданных вариантов в зависимости от определенного ключа. У оператора **case** также как и у оператора **if** может присутствовать слово **else**, которое имеет тот же смысл.

Оператор выбора **case** в общем виде выглядит так

case Ключ **of**

Метка1: Оператор1;

……

МеткаI: ОператорI

else Оператор

end;

Ключ – представляет собой переменную или выражение и является переключателем варианта. Метка1,…,МеткаI – постоянные значения (константы). Оператор1,…,ОператорI – это простые или составные операторы. Оператор – это совокупность операторов, которые выполняются при несоответствии ключа ни одной из меток.

Переключатель может быть любого порядкового типа при условии, что он совпадает по типу со своими метками. Метки являются внутренними и не должны описываться в разделе описаний основной программы. Значение переключателя должно быть определено заранее, и управление будет передано тому оператору, значение метки которого совпадет со значением переключателя.

1.3 Варианты индивидуальных заданий

Составить блок-схему алгоритма и написать программу для вычисления функции Y при заданном значении аргумента			
Вариант	Задание	Вариант	Задание
1	$y=\begin{cases} k-1, & \text{если } k<1 \\ (k+1)^2, & \text{если } 1\le k\le 3 \\ k, & \text{если } k>3 \end{cases}$	11	$y=\begin{cases} a^2-bx, & \text{если } x<0 \\ bx-a, & \text{если } 0\le x\le 3 \\ bx, & \text{если } x>3 \end{cases}$
2	$y=\begin{cases} 3x^2+5x-8, & \text{если } x<2 \\ 5x-\frac{a}{x}, & \text{если } -2\le x\le 3 \\ 4a^2+2a-x, & \text{если } x>3 \end{cases}$	12	$y=\begin{cases} b^2+a, & \text{если } b<0 \\ ba, & \text{если } 0\le b\le 3 \\ b^3, & \text{если } b\ge 3 \end{cases}$
3	$y=\begin{cases} az+b, & \text{если } z\le -10 \\ az^2+bz+c, & \text{если } -10<z<1 \\ z^3, & \text{если } z\ge 1 \end{cases}$	13	$y=\begin{cases} 0{,}5x, & \text{если } x<0 \\ 2x^2, & \text{если } 0\le x\le 2 \\ 0{,}3x, & \text{если } x>2 \end{cases}$
4	$y=\begin{cases} mn-a, & \text{если } a\le -5 \\ am+an, & \text{если } -5\le a\le 2 \\ am^2+an^2, & \text{если } a>2 \end{cases}$	14	$y=\begin{cases} ak+1, & \text{если } k\le -2 \\ k^2+b, & \text{если } -2\le k\le 0 \\ 5k, & \text{если } k>0 \end{cases}$
5	$y=\begin{cases} a^2, & \text{если } a<0 \\ b^2-a, & \text{если } 0\le a\le 2 \\ a^3, & \text{если } a>2 \end{cases}$	15	$y=\begin{cases} 3k^3, & \text{если } k\le -2 \\ k^2+2b, & \text{если } -2\le k\le 0 \\ 5k+b, & \text{если } k>0 \end{cases}$
6	$y=\begin{cases} ax^2+bx+10, & \text{если } x\le 3 \\ 4x+a, & \text{если } 3\le x\le 10 \\ 3b-5x, & \text{если } x>10 \end{cases}$	16	$y=\begin{cases} x^2, & \text{если } x<0 \\ x^3+5, & \text{если } 0\le x\le 2 \\ 3x-1, & \text{если } x>2 \end{cases}$

Составить блок-схему алгоритма и написать программу для вычисления функции Y при заданном значении аргумента			
Вариант	Задание	Вариант	Задание
7	$y=\begin{cases} x^2, & \text{если } x<-1 \\ x^3, & \text{если } 1\le x\le 0 \\ 2x-3, & \text{если } x>0 \end{cases}$	17	$y=\begin{cases} z, & \text{если } z<-3 \\ 3-z, & \text{если } -3\le z\le 0 \\ 8z+z^2, & \text{если } z>0 \end{cases}$
8	$y=\begin{cases} \frac{kx}{b}, & \text{если } x<-2 \\ (bx+k)^2, & \text{если } 2x\le 3 \\ 5kx, & \text{если } x>3 \end{cases}$	18	$y=\begin{cases} x^2-p^2+2, & \text{если } x<-10 \\ 2x^3-\frac{p}{3}, & \text{если } -x\le x\le 3 \\ 5x^3-4x^2+2, & \text{если } x>3 \end{cases}$
9	$y=\begin{cases} m^2-2m+1, & \text{если } m\le -5 \\ m+a, & \text{если } -5<m<2 \\ am+b, & \text{если } m\ge 2 \end{cases}$	19	$y=\begin{cases} \frac{3a+m}{5}, & \text{если } m<-5 \\ 23m^2+a^2, & \text{если } -5\le m\le 3 \\ (m-a)^2+25a, & \text{если } m>3 \end{cases}$
10	$y=\begin{cases} a\lvert x\rvert, & \text{если } x<-2 \\ ax^2+bx+ab, & \text{если } -2\le x\le 5 \\ 3x-a, & \text{если } x>5 \end{cases}$	20	$y=\begin{cases} \frac{a}{m}, & \text{если } m\le -2 \\ m^2+am, & \text{если } -2<m<5 \\ m^3, & \text{если } m\ge 5 \end{cases}$

Вариант задания определяется суммой двух последних цифр номера зачетной книжки, при этом нулевое значение соответствует двадцатому варианту.

2 Лабораторная работа №2. Алгоритмы циклических конструкций

2.1 Цель работы

Приобретение навыков составления циклических программ с помощью операторов цикла с предварительным условием, параметром и последующим условием.

2.2 Методические указания

Цикл – это многократное выполнение определённого участка алгоритма программы. Циклический алгоритм — это алгоритм, содержащий некоторое количество циклов.

Различают два основных типа циклов, цикл с известным числом повторений и цикл с неизвестным числом повторений. В обоих случаях подразумевается число повторений на стадии разработки алгоритма.

Также существует три основных типа циклических структур, а именно цикл с параметром, цикл с предусловием и цикл с постусловием.

2.2.1 Цикл с параметром

В данном цикле число повторений однозначно определяется правилом изменения параметра. Правило задается с помощью начального и конечного значений параметра с определенным шагом его изменения. Указанный цикл обычно применяется в тех случаях, когда число повторений цикла известно заранее.

```
for i:=1 to 10 do
begin
  k:=k+i;
  WriteLn(k);
end;
```

2.2.2 Цикл с предусловием

В указанном цикле количество шагов заранее не определено и зависит от входных данных. Каждый раз перед выполнением очередного шага цикла проверяется значение условного выражения. Тело цикла выполняется, если значение условного выражения истинно. Затем управление вновь передается проверке условия. Цикл повторяются пока условное выражение не примет значение ЛОЖЬ. Особенность цикла заключается в том, что если изначально условное выражение ложно, то тело цикла не выполнится ни разу.

```
while I<= N do
begin
  X: = X * I;
  I = I+1;
end;
```

2.2.3 Цикл с постусловием.

В данном цикле, как и в цикле с предусловием, число повторений тела цикла заранее не определено и зависит от входных данных. Но в отличие от цикла с предусловием, тело цикла с постусловием будет выполнено хотя бы один раз, перед проверкой условия. В указанной конструкции тело цикла будет выполняться до тех пор, пока значение условного выражения ложно. Как только оно становится истинным, выполнение цикла прекращается.

```
repeat
  i:= i+1;
  if i>10 then s:= s+1;
until (i<15);
```

2.2.4 Пустой оператор

Пустой оператор не содержит никаких символов и не выполняет никаких действий. Его удобно применять в тех случаях, когда необходимо организовать некоторый составной оператор, не содержащий в себе ни одного оператора.

```
repeat
until KeyPressed;
```

Пример - Составить блок-схему алгоритма и программу, вычисляющую сумму элементов заданного ряда с помощью оператора цикла с параметром.

$$\sum_{i=1}^{n} \frac{x + \cos(ix)}{2^i}$$

Блок-схема алгоритма

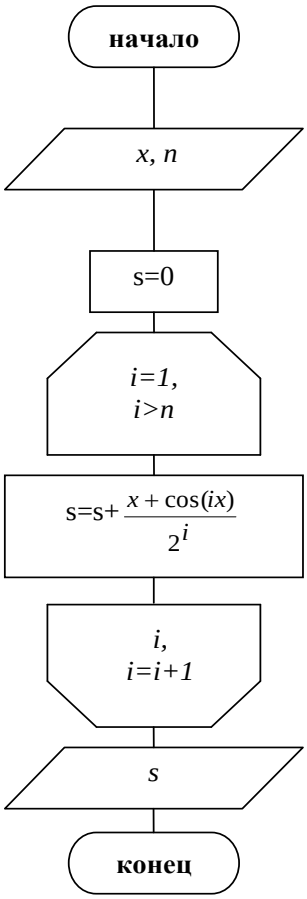


Рисунок 2.1 – Блок-схема алгоритма

Листинг программы

```
File  Edit  Search  Run  Compile  Debug  Tools  Options  Window  Help
                          CICLE.PAS                              1=[↕]
Program cicle;
Var
 i, n: Integer;
 x, s: Real;
Begin
Write('x= ');
ReadLn(x);
Write('n= ');
ReadLn(n);
s:=0;
For i:=1 To n Do
 s:=s+((x+cos(i*x))/exp(i*ln(2)));
WriteLn('s= ',s:5:5);
ReadLn;
End.
        12:5
F1 Help  F2 Save  F3 Open  Alt+F9 Compile  F9 Make  Alt+F10 Local menu
```

Рисунок 2.2 – Листинг программы

2.3 Варианты индивидуальных заданий

Составить блок-схему алгоритма и написать программу, вычисляющую сумму (произведений) элементов заданного ряда.			
Вариант	Задание	Вариант	Задание
1	$\sum_{k=1}^{n} \frac{1}{(2k+1)^2}$	11	$\sum_{k=1}^{n} \frac{5k+1}{3k^2-k-1}$
2	$\prod_{k=1}^{n} \frac{k^2}{k^2+2k+3}$	12	$\prod_{k=1}^{n} \frac{3k+1}{e^k}$
3	$\sum_{p=2}^{n} \left(1-\frac{1}{p}\right)$	13	$\sum_{k=1}^{n} \frac{k+1}{k^3}$
4	$\prod_{k=1}^{k} \frac{k+1}{k+2}$	14	$\prod_{k=1}^{n} \left(1+\frac{k+1}{k}\right)$
5	$\sum_{m=1}^{n} \left(2+\frac{1}{m}\right)$	15	$\sum_{k=3}^{n} \frac{(k-2)^2}{10(k-5)}$
6	$\prod_{i=2}^{n} \frac{i-1}{i+0,5}$	16	$\prod_{p=1}^{n} \frac{(p+2)^2}{4}$
7	$\sum_{k=2}^{n} \frac{1}{(2k)^2}$	17	$\sum_{k=1}^{n} \frac{(-1)k^2}{2k+1}$
8	$\prod_{k=2}^{n} \frac{k^2+k}{k-1}$	18	$\prod_{k=1}^{n} \frac{k^2}{(k-1)k}$
9	$\sum_{p=2}^{n} \frac{(3p-1)^2}{p-1}$	19	$\sum_{k=1}^{n} \frac{(k+2)^2}{k^2+2}$
10	$\prod_{p+1}^{n} \left(\frac{1}{p}+\frac{1}{p^2}\right)$	20	$\prod_{k=2}^{n} \frac{k(k+1)^2}{(k-1)^2}$

Вариант задания определяется суммой двух последних цифр номера зачетной книжки, при этом нулевое значение соответствует двадцатому варианту.

3 Лабораторная работа №3. Алгоритмы работы с одномерными массивами

3.1 Цель работы

Знакомство с понятием одномерного массива и приобретение навыков организации его обработки.

3.2 Методические указания

Массив - это структурированный тип данных, состоящий из фиксированного числа элементов, имеющих один и тот же тип. Допускается применение любого другого ранее описанного типа. Поэтому существуют массивы записей, массивы указателей, массивы строк и т.д. Число элементов массива фиксируется при описании и в процессе выполнения программы не меняется.

Одномерным массивом называется поименованная конечная последовательность объектов одинакового типа, которые именуются элементами массива. Графически одномерный массив представляется линейной таблицей с ячейками. Доступ к элементам массива осуществляется посредством указания имени массива и номера этого элемента в массиве. Номер элемента массива называется индексом. Под обработкой массива понимается анализ, замена, перестановка его элементов и т.п.

```
Type
  vect = array[1..n] of integer;
  mass = array[1..n] of vect;
Var
  matrix : mass;
```

Пример - Дан целочисленный массив из случайных чисел размера N. Найти сумму и количество элементов кратных 3.

Блок-схема алгоритма

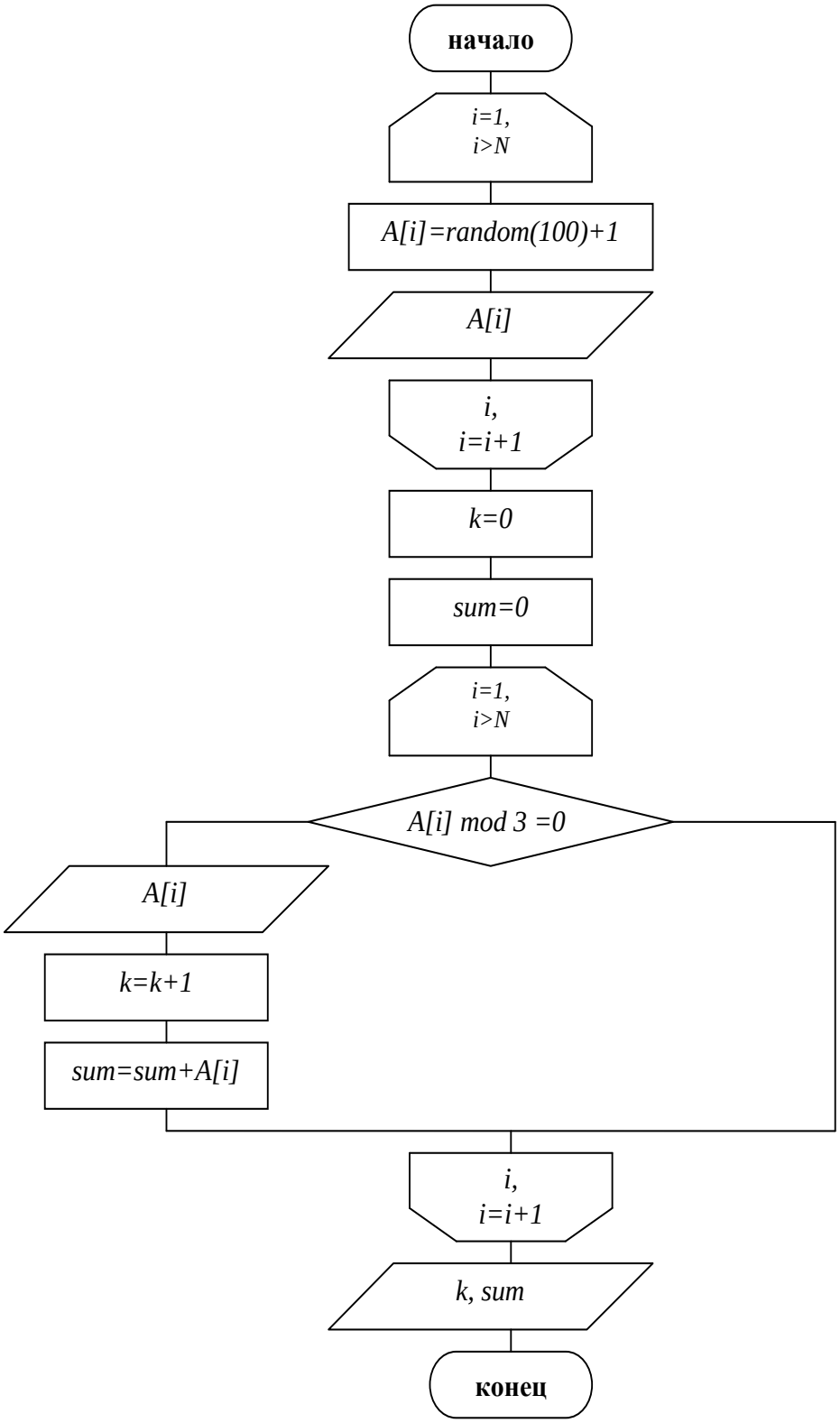


Рисунок 3.1 – Блок-схема алгоритма

Листинг программы

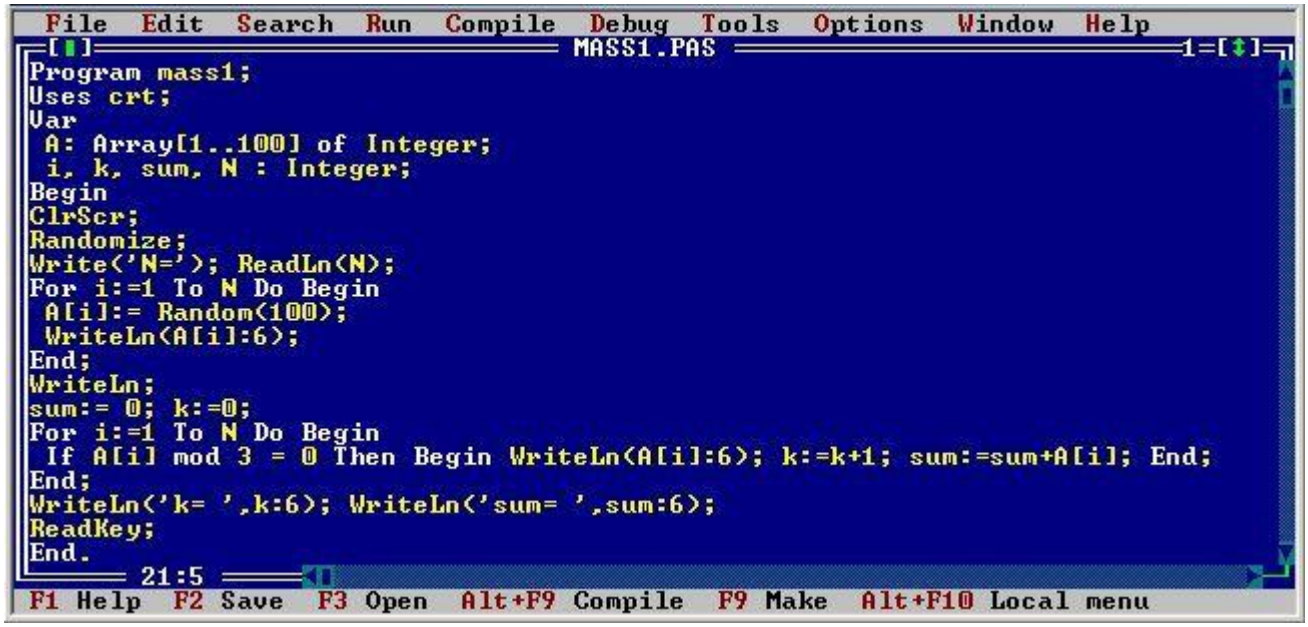


Рисунок 3.2 – Листинг программы

3.3 Варианты индивидуальных заданий

Составить блок-схему алгоритма и программу по заданию своего варианта	
Вариант	Задание
1	Дан целочисленный массив размера N. Вычислить сумму и количество положительных, произведение и количество отрицательных элементов, произведение элементов до первого отрицательного.
2	Дан целочисленный массив размером N. Вывести его элементы в обратном порядке. Вычислить среднее арифметическое положительных элементов массива.
3	Дан массив размера N. Вывести в начале его элементы с четными индексами, а затем – с нечетными. Вычислить среднее арифметическое элементов имеющих четные индексы.
4	Дан целочисленный массив размера N. Сформировать новый массив, элементы которого в 3 раза больше соответствующих элементов исходного массива.
5	Дан массив размером N. Вычислить сумму элементов стоящих на четных местах. Определить минимальный и максимальный элемент массива.
6	Дан массив размера N. Подсчитать количество положительных и отрицательных элементов. Вычислить сумму положительных элементов.
7	Дан целочисленный массив размером N. Отсортировать массив по убыванию. Вычислить сумму первого и последнего элемента массива.

Составить блок-схему алгоритма и программу по заданию своего варианта	
Вариант	Задание
8	Дан массив размера N. Заменить в этом массиве положительные элементы на 0, отрицательные элементы на 1, вывести полученный массив на экран и подсчитать количество 0 и 1.
9	Дан массив размера N. Вычислить сумму элементов массива. Если сумма больше 0, то отсортировать массив по возрастанию, если сумма меньше 0 – по убыванию.
10	Дан целочисленный массив размера N. Найти сумму всех элементов массива и разделить каждый элемент исходного массива на полученное значение. Результат поместить в новый массив. Найти сумму отрицательных элементов нового массива.
11	Дан целочисленный массив размера N. Отсортировать этот массив по убыванию. После сортировки заменить отрицательные элементы на 0, положительные на 1. Найти произведение отрицательных элементов.
12	Дан массив размера N. Сформировать новый массив, в который поместить элементы старого, начиная от k до k+7 позиции. В новом массиве найти среднее арифметическое максимального и минимального элементов.
13	Дан целочисленный массив A_i размера N. Сформировать новый массив B_i элементы, которого будут содержать элементы массива A_i удовлетворяющие условию $A_i<i$. Отсортировать массив B_i по убыванию.
14	Дан массив размера N. Подсчитать количество положительных элементов стоящих на четных местах. Вычислить произведение этих элементов.
15	Дан массив размера N. Подсчитать количество и произведение элементов до первого отрицательного.
16	Дан массив размером N. Вычислить сумму элементов стоящих на нечетных местах. Отсортировать по убыванию.
17	Дан массив размера N. Подсчитать, сколько в нем отрицательных, положительных и нулевых элементов. Вычислить среднее арифметическое суммы отрицательных и произведения положительных элементов массива.
18	Дан целочисленный массив размера N. Вычислить сумму тех элементов массива, которые по абсолютной величине больше заданного числа К. Подсчитать количество таких элементов.
19	Дан целочисленный массив размера N. Отсортировать массив по убыванию. Найти произведение только тех элементов массива, которые больше заданного числа Н.
20	В массиве размером N есть хотя – бы один нуль. Вычислить произведение элементов по первого нуля и сумму элементов после первого нуля.

Вариант задания определяется суммой двух последних цифр номера зачетной книжки, при этом нулевое значение соответствует двадцатому варианту.

4 Лабораторная работа №4. Алгоритмы работы с двумерными массивами

4.1 Цель работы

Знакомство с понятием двумерного массива и приобретение навыков организации его обработки.

4.2 Методические указания

Двумерным массивом называется поименованная конечная последовательность одномерных массивов. Графически двумерный массив представляется в виде прямоугольной таблицы.

Размерность двумерного массива обычно указывают в виде N×M, где N – это количество одномерных массивов, строк таблицы, а M – это количество элементов в каждом одномерном массиве, столбцов таблицы.

Доступ к какому-либо элементу массива, ячейке таблицы, осуществляется посредством указания имени массива и двух номеров, индексов массива. Один из индексов, чаще всего первый, является номером одномерного массива, строки таблицы, а другой – номером элемента в этом одномерном массиве.

Под обработкой массива понимается анализ, замена, перестановка его элементов и другие возможные действия.

```
Var
  mass = array[1..n,1..m] of integer;
```

Пример - Дана матрица размера N*N. Найти сумму максимальных значений в каждом столбце.

Блок-схема алгоритма

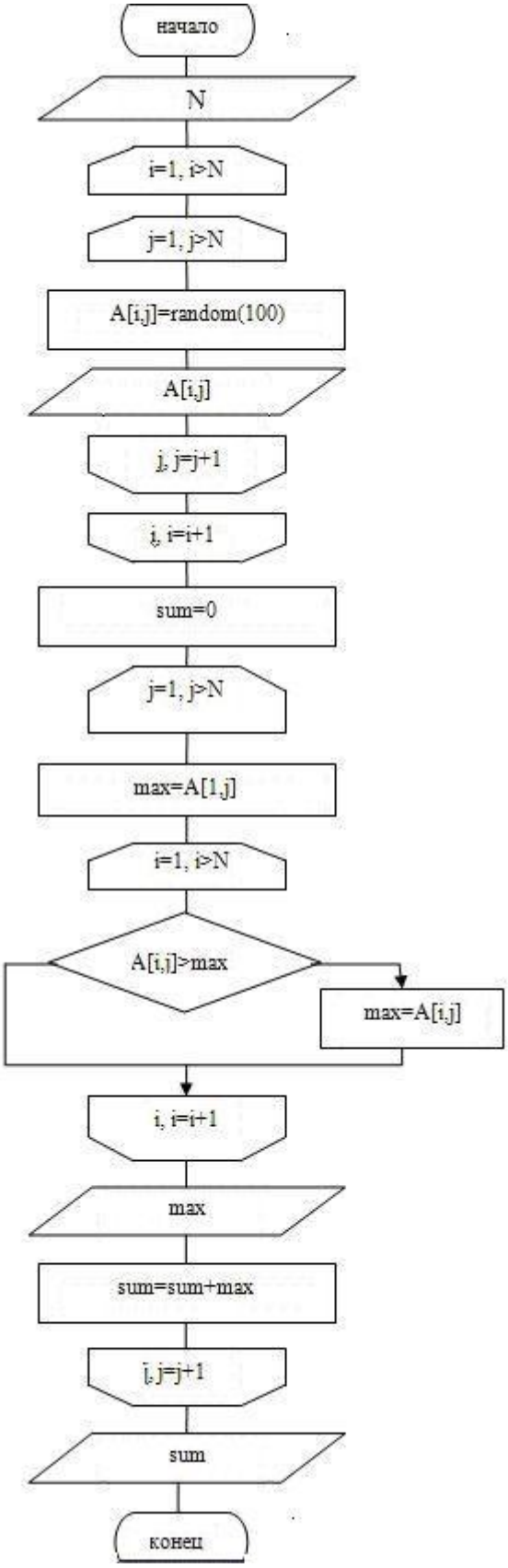


Рисунок 4.1 – Блок-схема алгоритма

Листинг программы

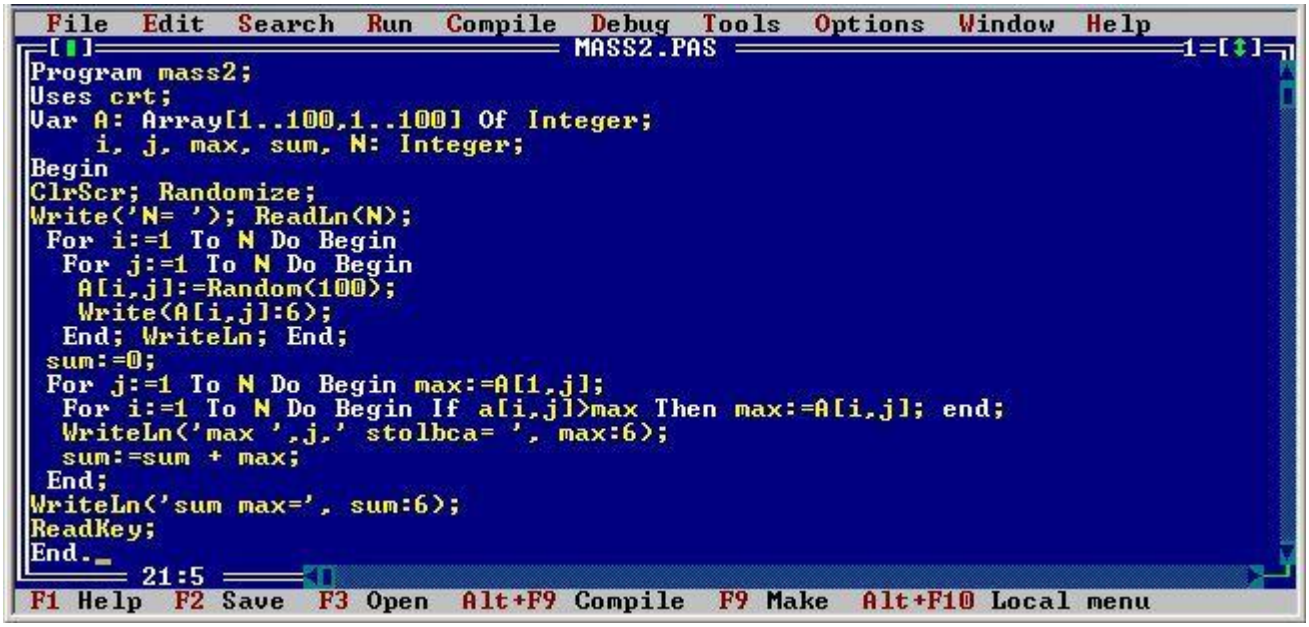


Рисунок 4.2 – Листинг программы

4.3 Варианты индивидуальных заданий

Составить блок-схему алгоритма и программу по заданию своего варианта	
Вариант	Задание
1	Дана квадратная матрица порядка N заполненная не нулевыми элементами. Заменить нулями элементы матрицы лежащие ниже главной диагонали и выше побочной. Сколько не нулевых элементов осталось в матрице.
2	Вычислить сумму и число положительных элементов матрицы размера M*N, находящихся над главной диагональю.
3	Дана квадратная матрица порядка N. Сравнить сумму элементов главной и побочной диагонали.
4	Дана матрица размера M*N. Заменить нулями отрицательные элементы матрицы и вычислить сумму положительных элементов.
5	Дана матрица размера M*N. Сравнить суммы элементов всех четных строк с суммой элементов всех ее нечетных столбцов.
6	Дана матрица размера M*N. Определить индексы нулевых элементов и подсчитать количество нулевых элементов. Ненулевые элементы заменить на 1.
7	Дана матрица размера M*N. Найти среднее арифметическое из минимальных значений в каждой строке и максимальных значений в каждом столбце.

Составить блок-схему алгоритма и программу по заданию своего варианта	
Вариант	Задание
8	Дана матрица размера M*N. Найти в каждой строке матрицы максимальный и минимальный элемент, и поменять их местами.
9	Дана матрица размера M*N. Определить индексы наибольшего и наименьшего элементов матрицы.
10	Дана квадратная матрица порядка N. Получить транспонированную матрицу. Сравнить сумму элементов главной диагонали данной матрицы с суммой элементов главной диагонали транспонированной матрицы.
11	Дана целочисленная матрица размера M*N. Найти сумму всех нечетных элементов и количество всех четных.
12	Дана матрица размера M*N. Найдите в ней равные элементы и выведите индексы этих элементов. Найти максимальный по абсолютному значению элемент.
13	Дана квадратная матрица порядка N. Вычислите произведение суммы элементов главной диагонали на сумму элементов k - ой строки.
14	Дана квадратная матрица размера N. Найти сумму элементов матрицы ниже главной диагонали, произведение не равных нулю элементов выше главной диагонали и количество элементов в главной диагонали больших N.
15	Дана матрица размера M*N. Упорядочить по возрастанию суммы элементов каждого столбца.
16	Дана матрица размера M*N. Найти в каждом столбце матрицы максимальный и минимальный элемент и поменять их местами.
17	Дана матрица размера M*N. Упорядочить по возрастанию элементы каждой строки.
18	Дана матрица размера M*N. Упорядочить по возрастанию суммы элементов каждой строки.
19	Дана матрица размера M*N. Удалить строку, которая содержит максимальный элемент всей матрицы.
20	Дана матрица размера M*N. Удалить столбец, содержащий минимальный элемент всей матрицы.

Вариант задания определяется суммой двух последних цифр номера зачетной книжки, при этом нулевое значение соответствует двадцатому варианту.

5 Лабораторная работа №5. Процедуры и функции

5.1 Цель работы

Приобретение навыков применения процедур и функций при написании программ.

5.2 Методические указания

5.2.1 Процедура

Процедура – это специальным образом оформленная последовательность операторов, которой присвоено имя.

Процедура описывается в декларативной части программы. Программа по отношению к процедуре выступает как внешняя. Процедура может быть вызвана по имени в исполнительной части программы из любой ее точки.

Общий вид описания процедуры:

Procedure <имя> (список формальных параметров, блок описания);
Var ;
…блок описания
Begin
<операторы> тело процедуры
End;

Описание процедуры включает ***заголовок*** имя процедуры и ***тело*** процедуры. Заголовок процедуры состоит из зарезервированного слова ***procedure***, имени процедуры и, заключенного в скобки, списка формальных параметров с указанием типа. ***Тело процедуры*** – представляет собой блок, по структуре аналогичный программе.

Структура процедуры копирует структуру программы в целом, за исключением завершающей процедуру точки с запятой вместо точки после последнего

оператора **End** программы. Порядок следования разделов описаний подчиняется тем же правилам, по которым строится вся программа.

Процедура может быть активирована в исполнительной части программы по ее имени. При вызове процедура будет содержать список параметров, если он присутствовал при описании в декларативной части. Типы используемых параметров при вызове процедуры не указываются.

5.2.2 Функция

Функция - это часть программы, аналог процедуры, которая определяет единственное значение. Значение может быть целое, символьное, логическое, вещественное, строковое или ссылочное, которое в свою очередь является результатом работы функции.

Заголовок функции начинается с зарезервированного слова ***Function*.** Список параметров функции, так же как и у процедуры может быть пустым. В этом случае скобки со списком параметров отсутствуют.

Содержательная часть функции, как и у процедуры, представляет собой блок, содержащий декларативную и исполнительную часть.

Общий вид описания функции:

Function <имя> (список параметров, блок описания) : <Тип Результата>;
Var…;
… блок описания
Begin
<операторы> тело функции
End;

Функция активируется в исполнительной части программы по ее имени. Функция может участвовать в выражениях в качестве операнда. В операторе присваивания функция стоит в правой его части.

При вызове функция будет содержать список параметров, если он присутствовал при ее описании в декларативной части программы. При вызове функции типы передаваемых параметров не указываются.

В отличие от процедуры функция возвращает результат выполнения операции, тип которого указывается в описании функции.

5.2.3 Рекурсивные процедуры и функции

Объект называется ***рекурсивным***, если он содержит сам себя или определен с помощью самого себя.

Процесс рекурсии происходит следующим образом. В начале происходит вызов некоторой функции или процедуры. Затем, еще до завершения своей работы, эта же функция или процедура вызывает саму себя, и в результате в памяти компьютера создается второй экземпляр той же самой функции или процедуры, второй экземпляр в ходе работы создает третий, и так далее.

Рекурсивные алгоритмы наиболее пригодны в случаях, когда поставленная задача или используемые данные определены рекурсивно. Однако следует избегать применения рекурсии в тех случаях, когда вычисляемые значения определяются с помощью простых рекуррентных соотношений.

Необходимым условием для работы рекурсивных процедур является наличие условия окончания рекурсивных вызовов, например, проверка значения изменяющегося параметра. Действия, связанные с такой проверкой, уже не должны содержать рекурсивных вызовов. Если это условие не будет выполняться, то ***глубина рекурсии*** станет бесконечной, что неизбежно приведет к аварийному завершению программы.

Пример 1 - Написать программу вычисления факториала числа N с применением рекурсивной функции.

Блок-схема алгоритма

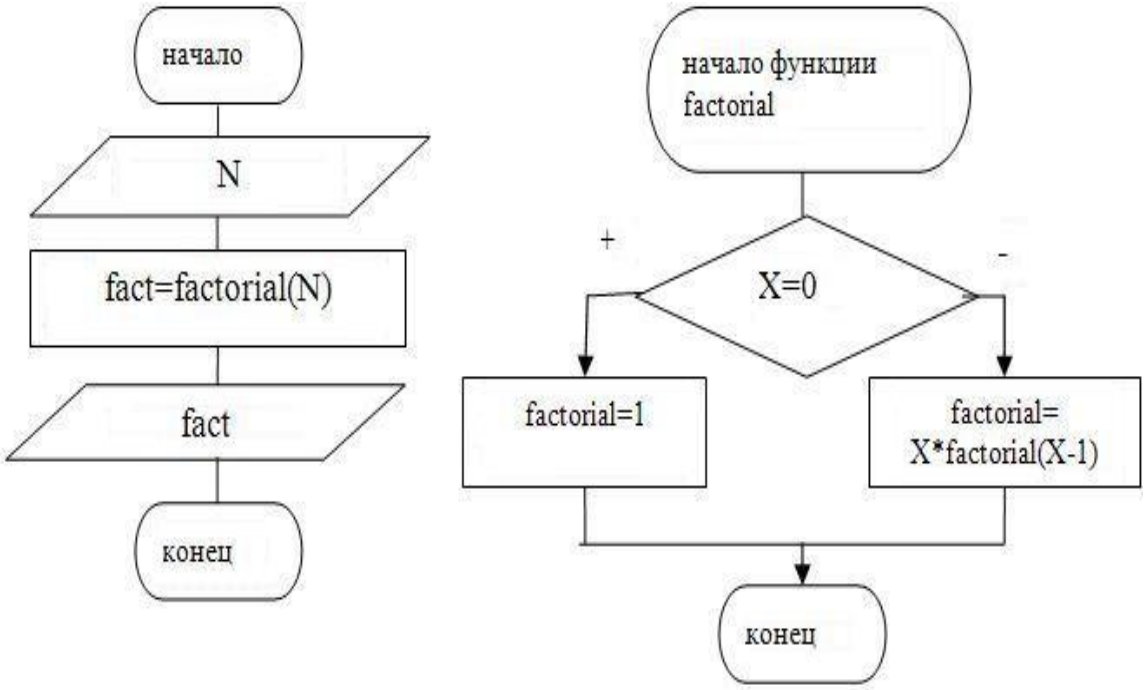


Рисунок 5.1 – Блок-схема алгоритма

Листинг программы

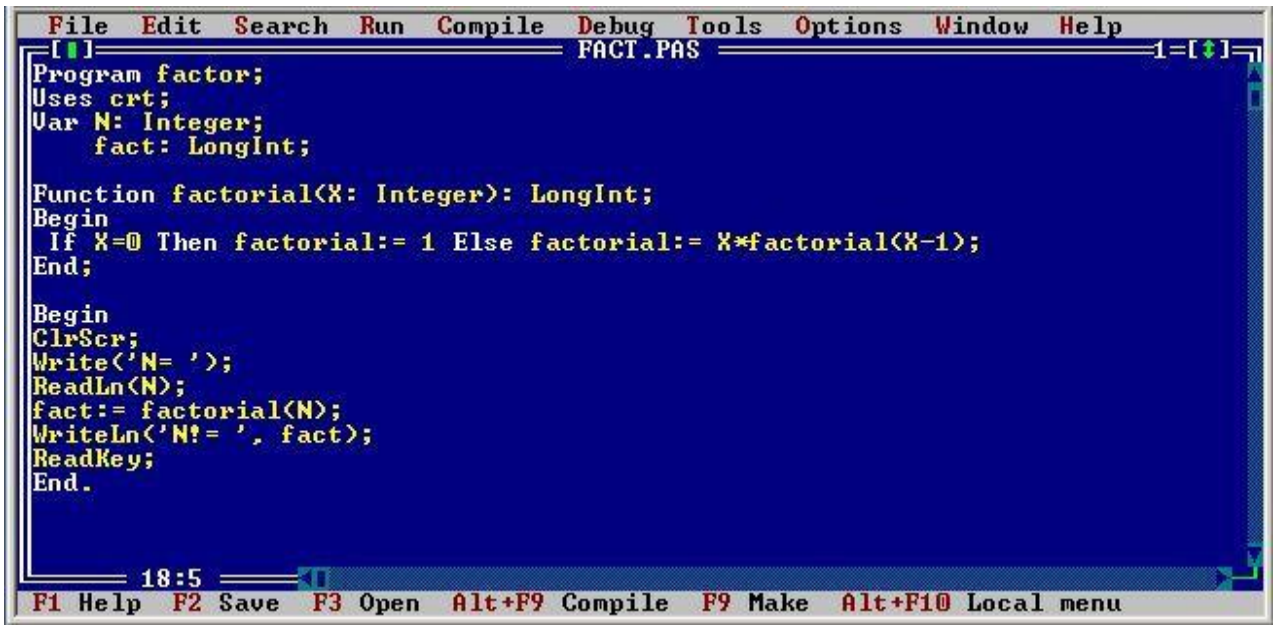


Рисунок 5.2 – Листинг программы

Пример 2 - Даны действительные числа s, *t*. При помощи функции, получить значение f(s, 2t, 0.12) + f(-1,t,s/2), вывод результата оформить в виде процедуры, где $f(a,b,c)=\dfrac{\sqrt{a^2+b^3}+c}{(a+c)(c+b)}$

Блок-схема алгоритма

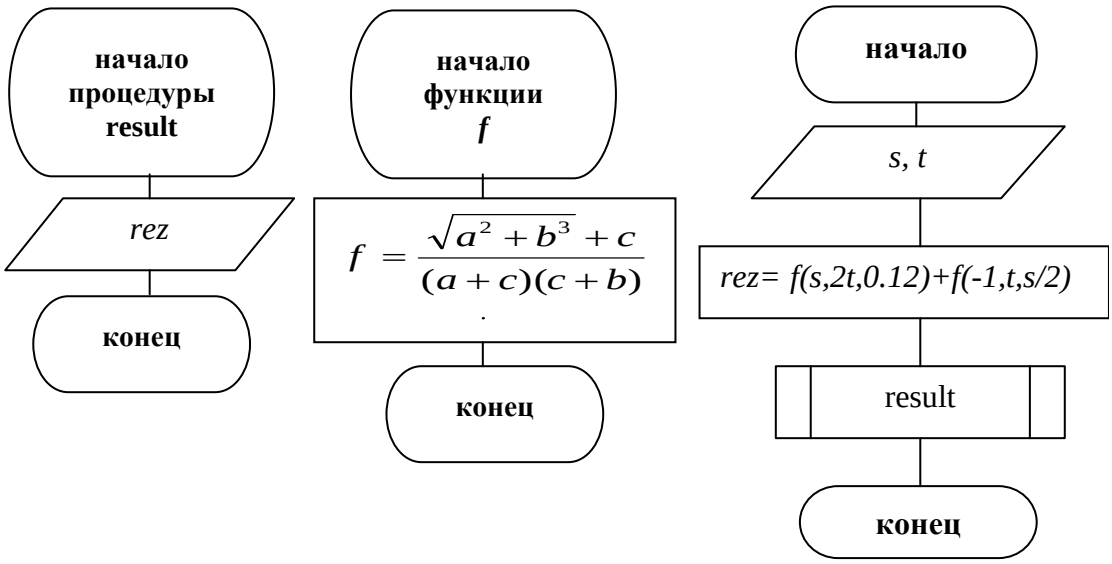


Рисунок 5.3 – Блок-схема алгоритма

Листинг программы

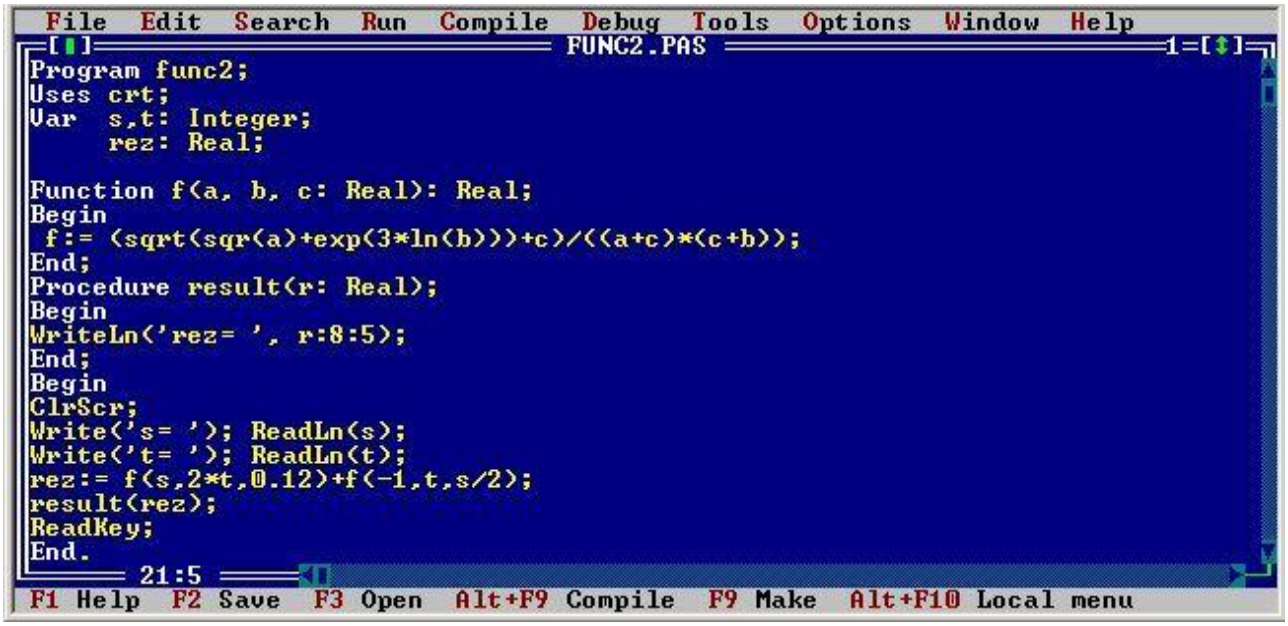


Рисунок 5.4 – Листинг программы

5.3 Варианты индивидуальных заданий

Даны действительные числа s, *t*. Применяя функцию, получить значение f(t,0.3*s)+f(s,0.2*t), вывод результата оформить в виде процедуры, где f(a,b) задан согласно варианту	
Вариант	Задание
1	$f(a,b)=\cos(\ln\lvert b^2-e^{a+0.2}\rvert)$
2	$f(a,b)=\ln\left\lvert\frac{b\cdot\sin a}{a^2\cdot\cos^2\lvert b-a\rvert}\right\rvert$
3	$f(a,b)=\left\lvert\sqrt[3]{\frac{a^5-\sqrt{\ln b}}{\cos\lvert ab\rvert}}\right\rvert$
4	$f(a,b)=\left\lvert\frac{\sin\left[\left(\sqrt{\lvert a-0.8\cdot\ln(ba)\rvert}\right)\right]}{a-b}\right\rvert$
5	$f(a,b)=\frac{0.15a+b}{\ln\lvert a^2-b^2\rvert}$
6	$f(a,b)=\frac{a^2+b\ln a^2}{\sin^2 b}$
7	$f(a,b)=\frac{\cos a-\cos b}{\sin^2 a+\sin^2 b}$
8	$f(a,b)=\frac{b+a\ln(b)}{a^2+b^2}$
9	$f(a,b)=\ln\left\lvert a+\sqrt{\lvert(a+b)\rvert}\right\rvert$
10	$f(a,b)=\frac{\lvert a^2-e^b\rvert}{\sqrt{a+b}}$
11	$f(a,b)=\cos(\ln\lvert a^2-\sin(b)\rvert)$
12	$f(a,b)=\sin(\ln\lvert b^2+ba^2\rvert)$
13	$f(a,b)=\frac{1}{\ln\lvert\sin^2 a+\cos b^2\rvert}$
14	$f(a,b)=\cos\left(\frac{\ln\lvert b^2-a^b\rvert}{\sin(ab)}\right)$
15	$f(a,b)=\left\lvert\sqrt[3]{\frac{b^2-\ln\lvert a\rvert}{\sin^2\lvert ab\rvert}}\right\rvert$

Даны действительные числа s, *t*. Применяя функцию, получить значение f(t,0.3*s)+f(s,0.2*t), вывод результата оформить в виде процедуры, где f(a,b) задан согласно варианту	
Вариант	Задание
16	$f(a,b)=\left\lvert\sqrt[5]{\frac{b^2-\sqrt{\ln a}}{\sin\lvert ab\rvert}}\right\rvert$
17	$f(a,b)=\lg(tg\lvert a\cdot b-0.5\rvert)$
18	$f(a,b)=tg\left[\frac{(b-a)\cdot tg\frac{ba}{2.4}}{\sqrt[3]{b^2-\sqrt{a}}}\right]$
19	$f(a,b)=\sqrt[5]{\frac{a-b^3}{tg(e^{\lvert ab\rvert})}}$
20	$f(a,b)=\ln\left\lvert\frac{b\cdot tga}{a^2\cdot\cos^2\lvert b-a\rvert}\right\rvert$

Вариант задания определяется суммой двух последних цифр номера зачетной книжки, при этом нулевое значение соответствует двадцатому варианту.

Список использованных источников

1 **Культин, Н.Б.** Turbo Pascal в задачах и примерах / Н.Б. Культин - СПб.: БХВ-Петербург, 2000. — 256 с.

2 **Окулов, С.М.** Программирование в алгоритмах / С.М. Окулов – М.: БИНОМ. Лаборатория знаний, 2002. – 341 с.

3 **Давыдов, В. Г.** Программирование и основы алгоритмизации: учеб. пособие для вузов / В. Г. Давыдов - М. : Высш. шк., 2003. - 447 с. : ил. - ISBN 5-06-004432-7.

4 **Долинский, М.С.** Алгоритмизация и программирование на Turbo Pascal: от простых до олимпиадных задач: учебное пособие / М.С. Долинский – СПб.: Питер Принт, 2004. – 240 с.

5 **Усковой, О. Ф.** Программирование на языке Паскаль: учеб. пособие для вузов: [задачник] / под ред. О.Ф. Усковой - СПб. [и др.] : Питер, 2005. - 336 с. - (Учебник для вузов). - Библиогр.: с. 332. - ISBN 5-318-00553-5.

6 **Окулов, С.М.** Основы программирования / С.М. Окулов – М.: БИНОМ. Лаборатория знаний, 2005. – 440 с.

7 **Столяр, С.Е.** Информатика. Представление данных и алгоритмы / С.Е. Столяр, А.А. Владыкин. – СПб.: Невский Диалект – М.: БИНОМ. Лаборатория знаний, 2007. – 382 с.

8 **Златопольский, Д.М.** Программирование: типовые задачи, алгоритмы, методы / Д.М. Златопольский – М.: БИНОМ. Лаборатория знаний, 2007. – 223 с.

9 **Великович, Л.С.** Программирование для начинающих / Л.С. Великович, М.С. Цветкова. – М.: БИНОМ. Лаборатория знаний, 2007. – 287 с.

10 **Андреева, Е. В.** Математические основы информатики. Элективный курс: учебное пособие / Е.В. Андреева, Л.Л. Босова, И.Н. Фалина. – М.: БИНОМ. Лаборатория Знаний, 2007. – 312 с.

11 **Каймин, В. А.** Информатика: учебник для вузов / В.А. Каймин – М. : ИНФРА-М, 2007, 2009. - 232с. - (Высшее образование).

12 **Фаронов, В. В.** Turbo Pascal 7.0. Практика программирования: учеб. пособие / В.В. Фаронов – М. : КноРус, 2007. - 416 с. - Прил.: с. 241-412. - Библиогр.: с. 413-415. - ISBN 978-5-85971-889-4.

13 **Охотников, Е. С.** Языки программирования: лаб. практикум / Е.С. Охотников – Тюмень: Изд-во ТюмГУ: Виндекс, 2008.

14 **Гуда, А. Н.** Информатика и программирование / А.Н. Гуда – М.: Дашков и К, 2010.

15 **Царев, Р. Ю.** Информатика и программирование : учеб. пособие / Р.Ю. Царев, А. Н. Пупков, В. В. Самарин, Е. В. Мыльникова. – Красноярск : Сиб. федер. ун-т, 2014. – 132 с. - ISBN 978-5-7638-3008-8 - Режим доступа: http://znanium.com/catalog.php?bookinfo=506203.

Приложение А

(обязательное)

Пример оформления отчета контрольной работы

Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Оренбургский государственный университет»

Факультет математики и информационных технологий

Кафедра программного обеспечения вычислительной техники
и автоматизированных систем

ОТЧЕТ

по контрольной работе
по дисциплине “Информатика и программирование”
“Разработка алгоритма и его программная реализация”

ОГУ 09.03.04.4016.018 О

Руководитель

_____ И.О. Фамилия

«_____» __ _____ 2016г.

Исполнитель
студент группы

_____ И.О. Фамилия

«_____» _____ 2016г.

Оренбург 2016

Содержание

1 Тема и цель работы

Тема: «Алгоритмы работы с двумерными массивами»

Цель:
- знакомство с понятием двумерного массива;
- приобретение навыков организации обработки двумерного массива.

2 Постановка задачи

Даны варианты заданий:

Необходимо реализовать следующие действия:
1) разработать алгоритм решения задачи;
2) построить блок-схему алгоритма;
3) написать программу, реализующую разработанный алгоритм;
4) выполнить тестирование программы с различным набором данных.

3 Теоретическая часть

Двумерным массивом называется поименованная конечная последовательность одномерных массивов. Графически двумерный массив представляется прямоугольной таблицей. Размер двумерного массива обычно указывают в виде n × m, где n – количество одномерных массивов (строк таблицы); m – количество элементов в каждом одномерном массиве (столбцов таблицы). Доступ к какому-либо элементу массива (ячейке таблицы) осуществляется посредством указания имени массива (имени таблицы) и двух номеров (индексов), один из которых (чаще всего первый) является номером одномерного массива (строки таблицы), а другой – номером элемента в этом одномерном массиве (столбца таблицы).

Под обработкой массива понимается анализ, замена, перестановка его элементов и другие возможные действия.

Под ***двумерным массивом (матрицей)*** понимают совокупность данных, каждое значение которых зависит от местоположения в строке и в столбце.

Каждый элемент матрицы описывается так – A[i,j]

где A – имя матрицы,

i – номер строки,

j – номер столбца.

Если в матрице количество строк равно количеству столбцов, то такая матрица называется ***квадратной***, иначе ***прямоугольной***.

$$A_{11} \quad A_{12} \quad A_{13}$$
$$A_{21} \quad A_{22} \quad A_{23}$$
$$A_{31} \quad A_{32} \quad A_{33}$$

Если в матрице количество строк равно количеству столбцов, то такая матрица называется ***квадратной***, иначе ***прямоугольной***.

$$B_{11} \quad B_{12} \quad B_{13}$$
$$B_{21} \quad B_{22} \quad B_{23}$$

Соотношение индексов в квадратной матрице:

1) i=j - элементы матрицы находятся на главной диагонали;
2) i<j - элементы матрицы находятся над главной диагональю;
3) i>j - элементы матрицы находятся под главной диагональю;
4) i+j=N+1 - элементы матрицы находятся на побочной диагонали;
5) i+j<N+1 - элементы матрицы находятся над побочной диагональю;
6) i+j>N+1 - элементы матрицы находятся под побочной диагональю.

Для написания программы сначала необходимо разработать алгоритм решения.

Алгоритм – это точное предписание, определяющее последовательность действий исполнителя, направленных на решение поставленной задачи.

Свойства алгоритмов:

1) Дискретность – описываемый процесс должен представлять собой упорядоченную совокупность четко разделенных друг от друга предписаний, образующих прерывную (дискретную) структуру алгоритма.

2) Понятность – предписания должны быть написаны на понятном исполнителю языке.

3) Результативность – означает возможность получения результата после выполнения конечного количества операций.

4) Определенность – состоит в совпадении получаемых результатов независимо от пользователя и применяемых технических средств.

5) Массовость – возможность применения алгоритма к целому классу однотипных задач, различающихся конкретными значениями исходных данных.

Для обеспечения возможности реализации на ЭВМ алгоритм должен быть записан на языке понятном компьютеру, т.е. на ***языке программирования.***

4 Практическая часть

4.1 Разработка алгоритма решения задачи

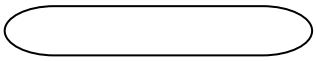
Дана матрица размера N*N. Для нахождения максимального значения в каждом столбце необходимо выбрать один элемент столбца и затем пошагово сравнить его с последующими элементами. Если текущий элемент больше значения выбранного элемента, то нужно заменить значение элемента текущим значением.

Цикл продолжается до последнего элемента в столбце. В итоге будет найдено максимальное значение из всех элементов столбца. После этого нужно найти максимальные элементы в каждом столбце. Для нахождения суммы максимальных значений в каждого столбца нужно суммировать все найденные максимальные элементы.

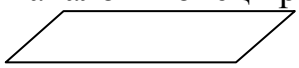
4.2 Построение блок-схемы алгоритма

Для обеспечения возможности реализации на ЭВМ алгоритм должен быть записан на языке понятном компьютеру, т.е. на ***языке программирования.*** При составлении программы также удобно использовать блок-схемы.

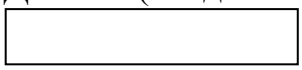
Основные блоковые конструкции:



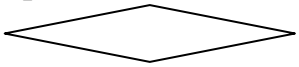
Начало и конец программы



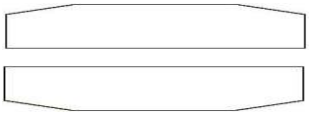
Данные (ввод-вывод)



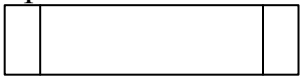
Процесс



Решение



Граница цикла



Предопределенный процесс

Соединитель

На рисунке 4.1 представлена блок-схема алгоритма.

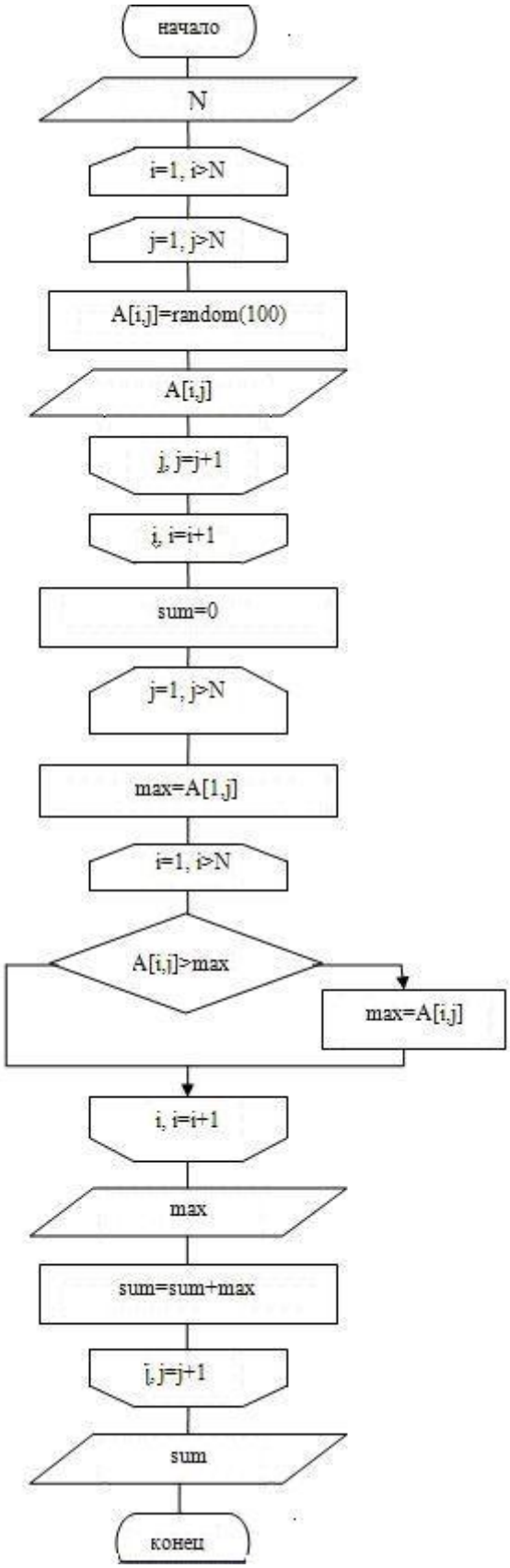


Рисунок 4.1 – Блок-схема алгоритма

4.3 Написание программы, реализующей разработанный алгоритм

Текст программы представляет собой последовательность лексем и их разделителей. *Лексема* – это минимальная единица языка, имеющая самостоятельный смысл.

Классы лексем:

Служебные (зарезервированные) слова. Каждое служебное слово представляет собой неделимое образование, смысл которого фиксирован в языке. Пример: **begin**, **end**, **var**, **program**, **if** и т.п.

Идентификаторы (имена). Идентификаторы вводятся для обозначения в программе переменных, констант, типов, меток, процедур и функций и формируются из букв и цифр в соответствии со следующим правилом: идентификатор есть последовательность букв, цифр и символов подчеркивания (_), начинающаяся с буквы.

Изображения. Эта группа лексем обозначает числа, символьные строки и некоторые другие значения.

Знаки операций. Формируются из одного или нескольких символов и предназначены для указания действий по преобразованию данных и вычисления значений.

Разделители. Используются для отделения лексем друг от друга, в тех случаях, когда их слитное написание может исказить текст программы.

В текстах программ допускаются фрагменты пояснительного характера – *комментарии.* Наличие комментариев не изменяет смысл программы и не влияет на ее выполнение.

На рисунке 4.2 представлен листинг программы реализующей разработанный алгоритм.

Рисунок 4.2 – Листинг программы

4.4 Тестирование программы

Процесс тестирования означает многократное выполнение программы с различным набором входных значений с целью выявления ошибок и несоответствий программной реализации разработанного алгоритма.

На рисунках 4.3 и 4.4 представлены результаты тестирования программы с различными входными значениями.

Заданно значение параметра N равное 5;

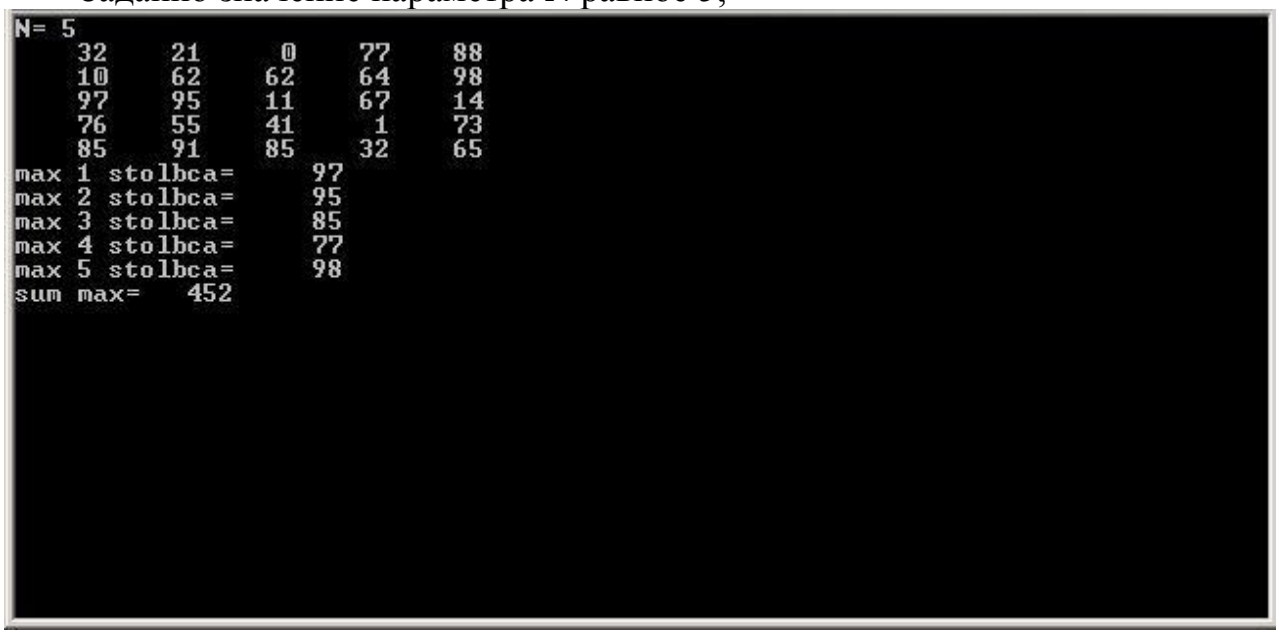


Рисунок 4.3 – Результат первого выполнения программы

Заданно значение параметра N равное 10;

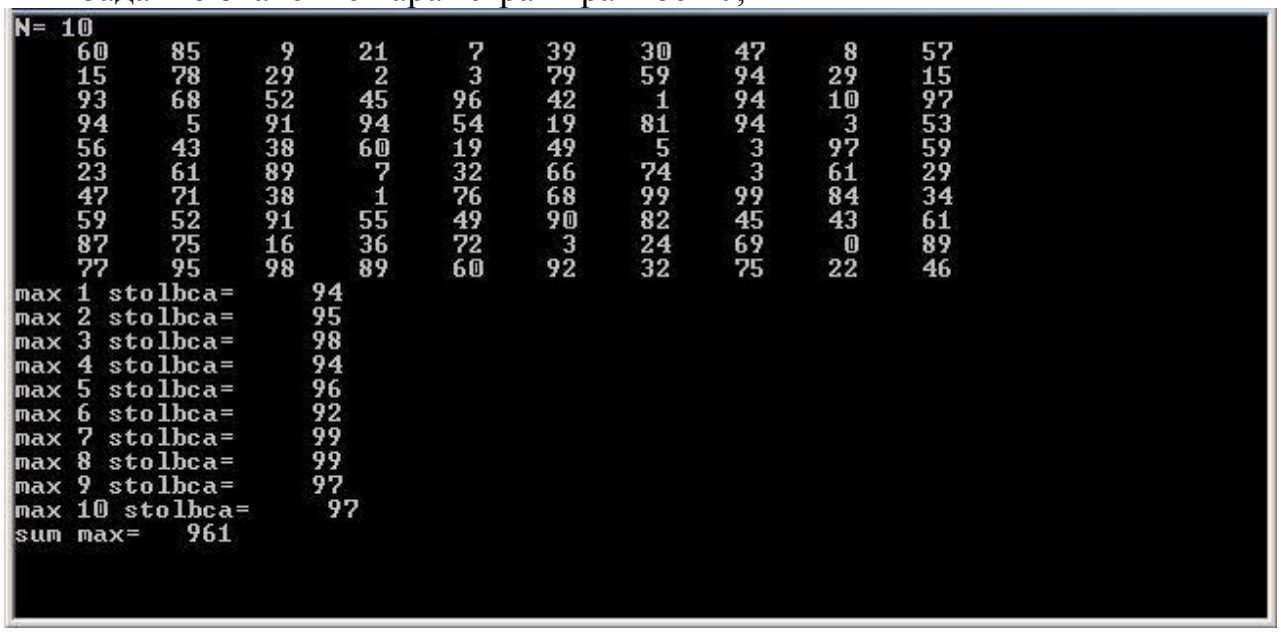


Рисунок 4.4 – Результат второго выполнения программы

Список использованных источников

1 **Фаронов, В. В.** Turbo Pascal 7.0. Практика программирования: учеб. пособие / В.В. Фаронов - М. : КноРус, 2007. - 416 с. - Прил.: с. 241-412. - Библиогр.: с. 413-415. - ISBN 978-5-85971-889-4.

2 **Великович, Л.С.** Программирование для начинающих / Л.С. Великович, М.С. Цветкова. – М.: БИНОМ. Лаборатория знаний, 2007. – 287 с.

3 **Усковой, О. Ф.** Программирование на языке Паскаль: учеб. пособие для вузов: [задачник] / под ред. О.Ф. Усковой - СПб. [и др.] : Питер, 2005. - 336 с. - (Учебник для вузов). - Библиогр.: с. 332. - ISBN 5-318-00553-5.

4 **Долинский, М.С.** Алгоритмизация и программирование на Turbo Pascal: от простых до олимпиадных задач: учебное пособие / М.С. Долинский – СПб.: Питер Принт, 2004. – 240 с.

5 **Давыдов, В. Г.** Программирование и основы алгоритмизации: учеб. пособие для вузов / В.Г. Давыдов - М. : Высш. шк., 2003. - 447 с. : ил. - ISBN 5-06-004432-7.

Приложение А

(обязательное)

Листинг программы

```
Program mass2;
Uses crt;
Var A: Array[1..100,1..100] Of Integer;
      i, j, max, sum, N: Integer;
Begin
ClrScr;
Randomize;
Write('N= ');
ReadLn(N);
For i:=1 To N Do Begin
  For j:=1 To N Do Begin
    A[i,j]:=Random(100);
    Write(A[i,j]:6);
  End;
WriteLn;
End;
sum:=0;
For j:=1 To N Do Begin max:=A[1,j];
  For i:=1 To N Do Begin
    If A[i,j]>max Then max:=A[i,j];
  End;
  WriteLn('max ', j, ' stolbca= ', max:6);
  sum:= sum + max;
End;
WriteLn(sum max=', sum:6);
ReadKey;
End.
```