Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ

Кафедра автоматизации обработки информации (АОИ)

Н. В. Пермякова

ИНФОРМАТИКА И ПРОГРАММИРОВАНИЕ

Методические указания по выполнению лабораторных работ

2016

Корректор: Сарина С. Д.

Пермякова Н. В.
Информатика и программирование : методические указания по выполнению лабораторных работ. – Томск : ФДО ТУСУР, 2016. – 67 с.

Методические указания содержат краткое изложение теоретического материала по соответствующим темам, описание последовательности выполнения работ, варианты лабораторных работ и правила оформления отчетов.

Предназначены для студентов направления 231000.62 (09.03.04) «Программная инженерия», а также всех, начинающих изучать основы структурного программирования на языке Си.

СОДЕРЖАНИЕ

1 ВВЕДЕНИЕ

На основании рабочей программы по курсу «Информатика и программирование» студенты за время изучения курса должны выполнить четыре лабораторные работы. Целью выполнения лабораторных работ является формирование компетенций, которыми должен обладать будущий дипломированный специалист:

- понимание основных концепций, принципов, теорий и фактов, связанных с информатикой (ПК-1);
- умение применять основы информатики и программирования к проектированию, конструированию и тестированию программных продуктов (ПК-10).

В ходе выполнения лабораторных работ у студентов закрепляется знание синтаксиса языка программирования Си, навыки составления алгоритмов, умение излагать описанные алгоритмы на языке программирования Си.

Основная цель первой лабораторной работы – научить студентов основам построения алгоритмов, использующих условную конструкцию. Примером таких алгоритмов являются задачи геометрии на плоскости. При выполнении лабораторной работы студент должен обладать знаниями из школьного курса геометрии.

Вторая лабораторная работа посвящена обработке одномерных массивов. Основная цель этой работы – сформировать навыки применения рассмотренных в курсе различных алгоритмов поиска в массивах.

В третьей лабораторной работе студентам предлагается реализовать алгоритмы обработки матриц, написав собственные функции.

Варианты четвертой лабораторной работы содержат задания на обработку текстовых массивов.

Лабораторные работы могут выполняться на свободно распространяемом пакете *Dev-C++*. Скачать пакет можно по ссылке: http://www.bloodshed.net/dev/devcpp.html.

2 ЛАБОРАТОРНАЯ РАБОТА № 1 «ПРОВЕРКА УСЛОВИЙ»

Цель работы: закрепить навыки работы с условным оператором языка Си.

2.1 Задачи проверки вхождения точки с заданными координатами в ограниченную область

Проверка расположения точки с координатами (*x*, *y*) *относительно прямой*

Пусть уравнение прямой задано в каноническом виде $y = ax + b$. Тогда все точки, лежащие *на линии* прямой (рис. 2.1), подчиняются условию $y = ax + b$. На рисунке это условие выполняется для точки с координатами (x_3, y_3). Все точки, лежащие *левее линии* прямой, подчиняются условию $y < ax + b$, это условие выполняется для точки с координатами (x_1, y_1). Все точки, лежащие *правее линии* прямой, подчиняются условию $y > ax + b$. Это условие является истинным для точки с координатами (x_2, y_2). Тогда для выбранных трех точек являются истинными условия:

- $y_3 = ax_3 + b$;
- $y_2 > ax_2 + b$;
- $y_1 < ax_1 + b$.

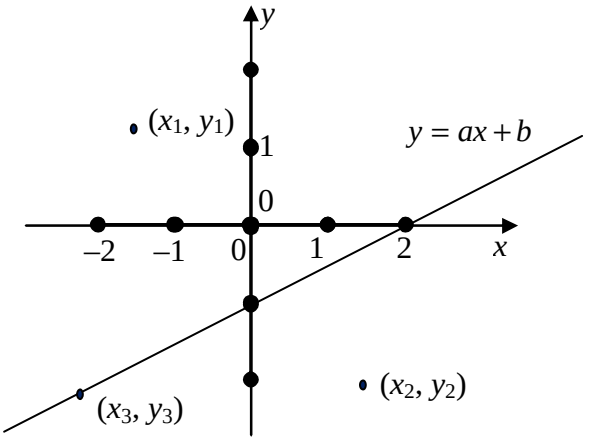


Рис. 2.1 – Расположение точки относительно прямой $y = ax + b$

Для прямой, изображенной на рис. 2.1, составим уравнение прямой по двум заданным точкам: прямая проходит через точки с координатами (0, –1) и (2, 0). Найдем коэффициенты уравнения a и b. Для этого решим систему уравнений:

$$\begin{cases} -1 = a \cdot 0 + b \\ 0 = a \cdot 2 + b \end{cases} \Rightarrow \begin{cases} b = -1 \\ a = \dfrac{-b}{2} \end{cases} \Rightarrow \begin{cases} b = -1 \\ a = 0.5 \end{cases} \Rightarrow \quad y = 0.5x - 1.$$

Таким образом, проверить местоположение произвольной точки с координатами (x, y) можно, написав следующий код:

```
…
if (y<0.5*x–1) printf(“Точка расположена левее прямой”);
   else   if(y>0.5*x–1) ) printf(“Точка расположена правее прямой”);
            else ) printf(“Точка расположена на прямой”);
…
```

Проверка расположения точки относительно окружности с заданным центром известного радиуса

Каноническое уравнение окружности выглядит следующим образом:

$$R^2 = (x - x_1)^2 + (y - y_1)^2, \qquad (1)$$

где R – радиус окружности,

(x_1, y_1) – координаты центра окружности.

Тогда (рис. 2.2) для точки с координатами (x_4, y_4), выполняется равенство:

$$R^2 = (x_4 - x_1)^2 + (y_4 - y_1)^2.$$

Выражение (1) истинно для всех точек, лежащих на линии окружности. Для точки с координатами (x_2, y_2) и для всех точек, лежащих за окружностью, выполняется неравенство:

$$R^2 < (x_2 - x_1)^2 + (y_2 - y_1)^2.$$

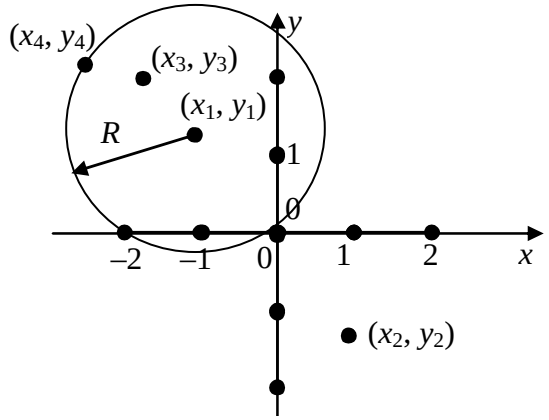


Рис. 2.2 – Расположение точки относительно окружности $R^2 = (x - x_1)^2 + (y - y_1)^2$

Из этого неравенства следует, что радиус R окружности меньше радиуса окружности с центром в точке (x_1, y_1), на которой лежит точка с координатами (x_2, y_2). Соответственно, для точки с координатами (x_3, y_3) выполняется неравенство:

$$R^2 > (x_3 - x_1)^2 + (y_3 - y_1)^2.$$

То есть радиус R окружности больше радиуса окружности с центром в точке (x_1, y_1), на которой лежит точка с координатами (x_3,y_3).

2.2 Примеры проверки вхождения точки с заданными координатами в фигуру

Пример 1. Напишите алгоритм, проверяющий вхождение точки в область, изображенную на рисунке 2.3.

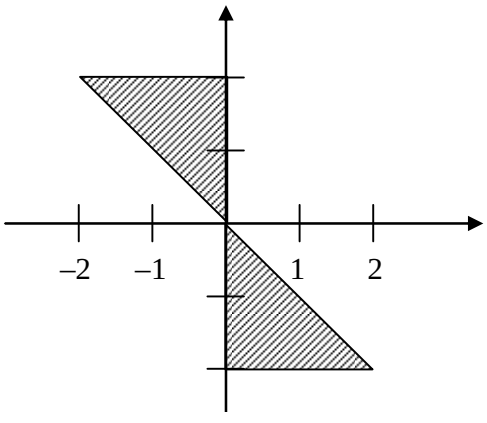


Рис. 2.3 – Пример 1

Построим условия вхождения точки в заданную область:

Уравнение прямой, на которой лежат гипотенузы прямоугольных треугольников, образующих фигуру:

$$y = -x.$$

Разобьем фигуру на две части. Точка будет считаться принадлежащей фигуре, если она попадет в первую или вторую часть.

Первую (верхнюю часть) можно ограничить следующими условиями:

$$(y >= -x) \text{ и } (x <= 0) \text{ и } (y <= 2).$$

Первое условие описывает гипотенузу, второе и третье условие описывают катеты. Условия связаны между собой связками **И** (логическое умножение).

Вторую (нижнюю часть) можно ограничить условиями:

$$(y <= -x) \text{ и } (x >= 0) \text{ и } (y >= -2).$$

Общее условие для двух частей будет выглядеть следующим образом:

если $(y >= -x)$ **и** $(x <= 0)$ **и** $(y <= 2)$ **или**

$(y <= -x)$ **и** $(x >= 0)$ **и** $(y >= -2)$,

то «Точка принадлежит заданной области»,

иначе «Точка не принадлежит заданной области».

Тогда алгоритм проверки вхождения точки с заданными координатами будет выглядеть следующим образом:

```
алг Пример 1 нач
вещ x, y
ввод x, y
если (y >= –x) и (x <= 0) и (y <= 2 ) или
(y <= –x) и (x >= 0) и (y >= –2),
то «Точка принадлежит заданной области»,
иначе «Точка не принадлежит заданной области»
кон
```

Пример 2. Рассмотрим еще один пример, заданная область изображена на рис. 2.4. В этом случае:

- уравнение прямой $y = x$;
- уравнение окружности $1 = (x - 1)^2 + (y - 1)^2$.

Ограниченная область находится правее прямой ($y < x$) и внутри окружности ($1 > (x - 1)^2 + (y - 1)^2$).

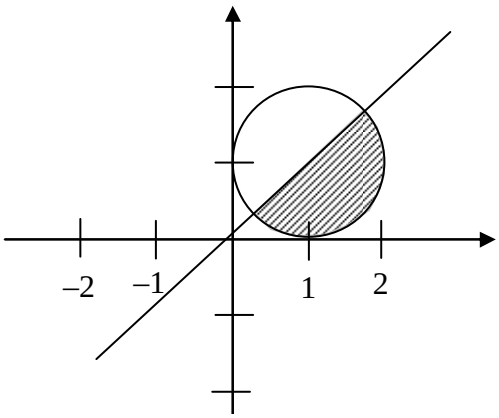


Рис. 2.4 – Пример 2

Тогда общее условие будет выглядеть следующим образом:

если $y < x$ **и** $1 > (x - 1)^2 + (y - 1)^2$,

то «Точка принадлежит заданной области»,

иначе «Точка не принадлежит заданной области».

Тогда алгоритм проверки вхождения точки с заданными координатами будет выглядеть следующим образом:

алг Пример 2 нач

вещ x, y

ввод x, y

если $y < x$ и $1 > (x - 1)^2 + (y - 1)^2$,

то «Точка принадлежит заданной области»,

иначе «Точка не принадлежит заданной области»

кон

2.3 Порядок выполнения работы

1. Выбрать индивидуальный вариант.

2. По выбранному варианту определить условия вхождения точки в заданную область.

3. Составить и записать алгоритм решения задачи

4. Составить программу, реализующую записанный алгоритм.

5. Отладить программу.

6. Написать отчет о проделанной работе.

2.4 Содержание отчета

Отчет по лабораторной работе № 1 должен содержать:

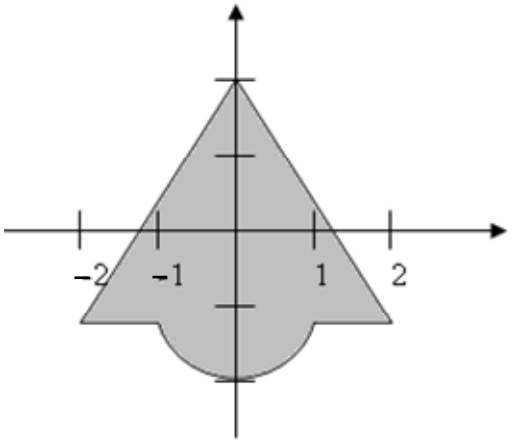
- титульный лист,
- текст индивидуального варианта,
- алгоритм решения задачи,
- текст программы,
- результаты тестирования программы.

Пример оформленного отчета приведен в приложении А.

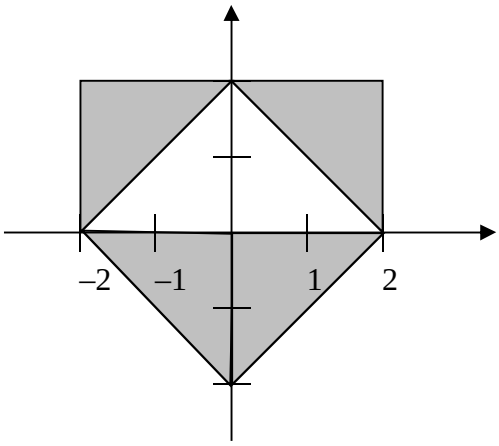
2.5 Индивидуальные варианты заданий на лабораторную работу № 1

Проверить, принадлежит ли точка с заданными координатами (x, y) заштрихованной области.

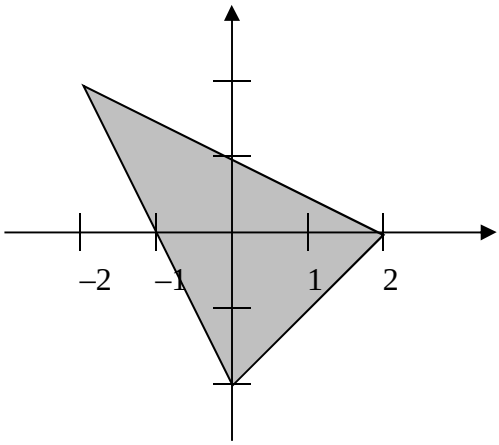
Вариант 1



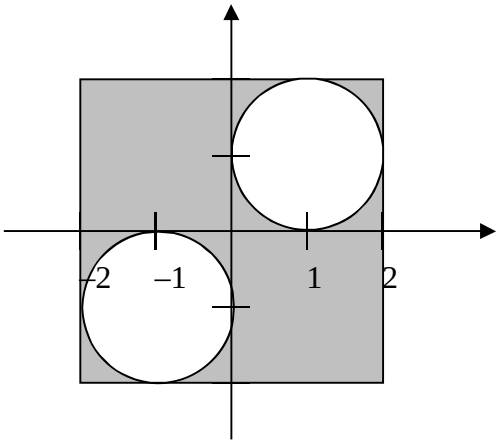
Вариант 2



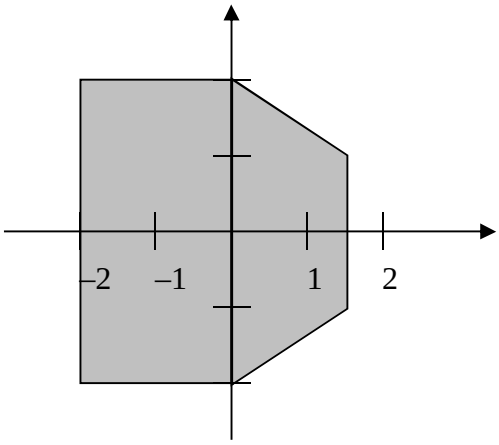
Вариант 3



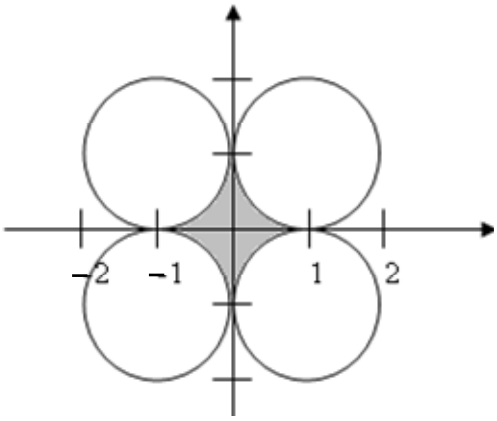
Вариант 4



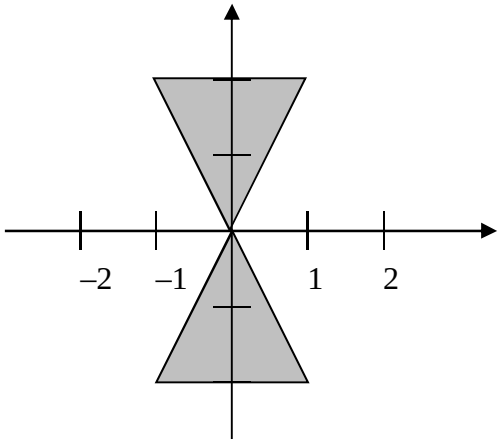
Вариант 5



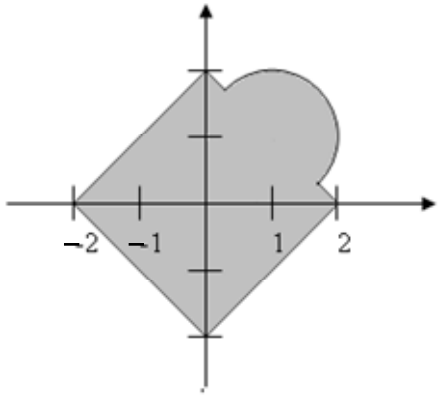
Вариант 6



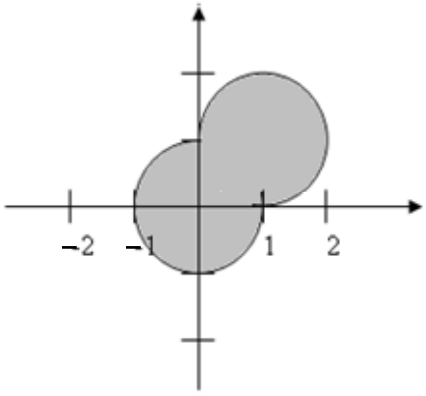
Вариант 7



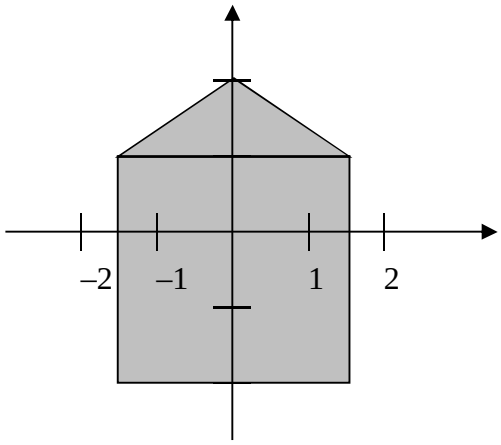
Вариант 8



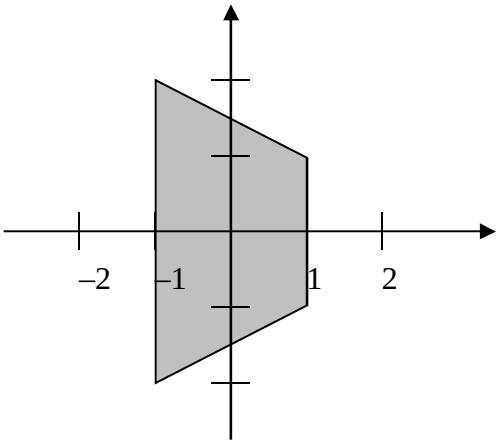
Вариант 9



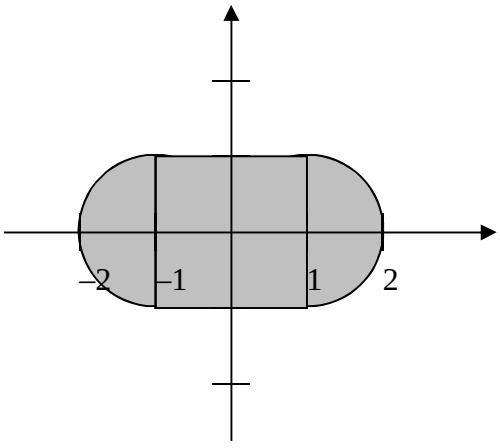
Вариант 10



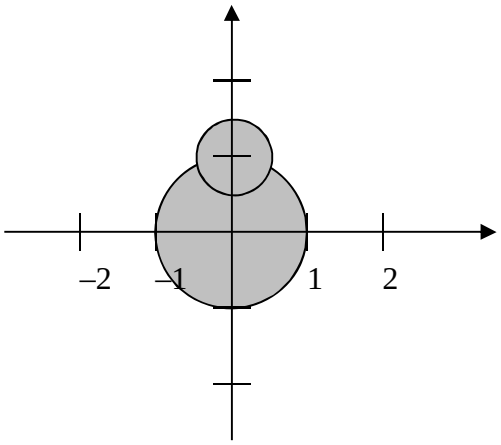
Вариант 11



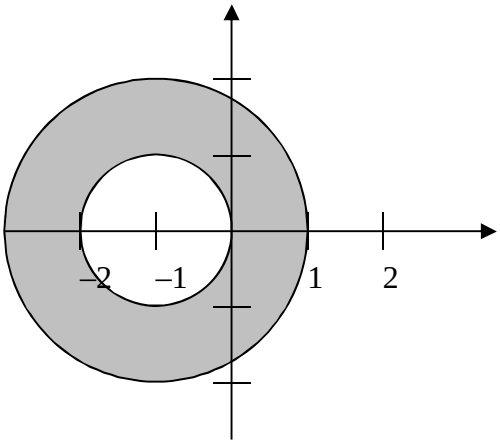
Вариант 12



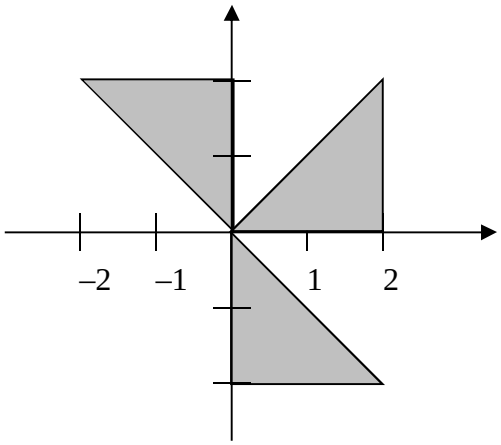
Вариант 13



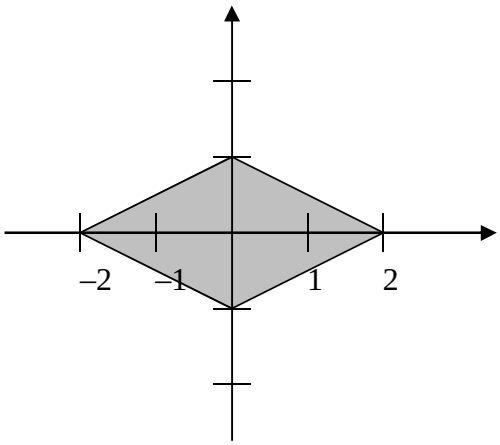
Вариант 14



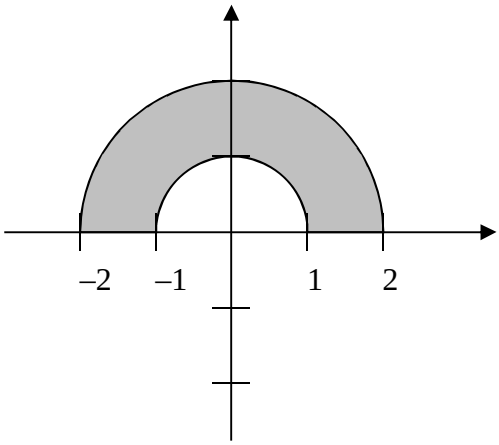
Вариант 15



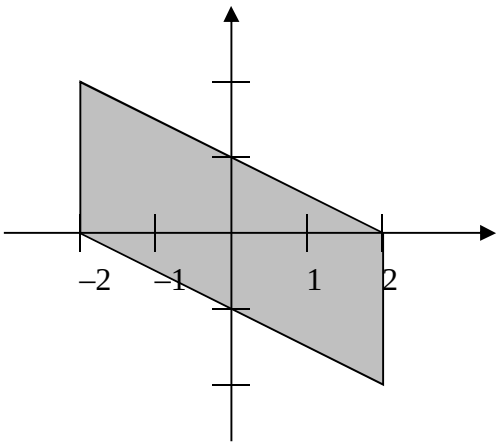
Вариант 16

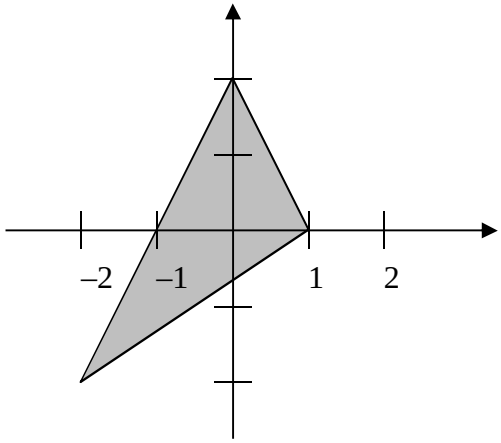
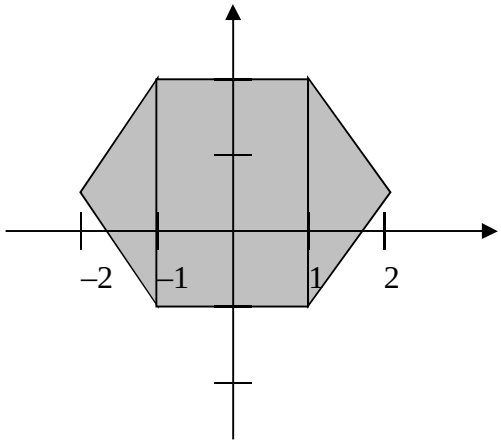


Вариант 17



Вариант 18



Вариант 19	Вариант 20
	

3 ЛАБОРАТОРНАЯ РАБОТА № 2 «ОБРАБОТКА МАССИВОВ»

Цель работы: получить навыки обработки одномерных массивов, навыки практической реализации циклических процессов.

3.1 Конструкции циклов в языке Си

Цикл с фиксированным числом операций for

Синтаксис:

for (секция инициализации значения; секция проверки условия; секция коррекции) {тело цикла}

Пример:

```
…
int i;
for(i=1;i<5;i++)
 {printf(“%d ”,2*i+1);}
```

В начале работы цикла единственный раз выполняется секция инициализации, переменной *i* присваивается значение 1. После этого управление передается в секцию проверки условия, если условие истинно, то управление передается в тело цикла, на экран выводится число 3. Далее управление передается в секцию коррекции и значение переменной *i* увеличивается на единицу. После этого вновь выполняется проверка условия, и если условие истинно, управление вновь передается в тело цикла. Таким образом, при выполнении вышеописанного цикла на экран выведутся числа 3 5 7 9. Цикл выполнится четыре раза, при *i* = 5 условие станет ложным.

Цикл while

Синтаксис:

while (условное выражение)

{ тело цикла

}

Запишем приведенный выше пример с помощью цикла *while*.

…

int i=1;

while(*i*<5) { *printf*(“%*d* ”,2**i*+1);

i++;

}

Тело цикла выполняется до тех пор, пока условие истинно.

Цикл do while

Синтаксис:

do {

тело цикла

} *while* (условное выражение);

Пример вывода на экран чисел 3, 5, 7, 9 с помощью цикла *do while* может быть записан следующим образом:

int i=0;

do{

i++;

printf(“%*d* ”,2**i*+1);} *while*(*i*<4);

Операторы безусловной передачи управления continue и break

Оператор *break* досрочно завершает выполнение цикла. Управление передается первому оператору, следующему за циклом. Например:

…

int i;

while(1) {

```
scanf(“%d”, &i);
if (i>10) break;
}
i++; …
```

Условное выражение описанного цикла предписывает ему выполняться бесконечно, но цикл закончит свою работу, если с клавиатуры будет введено число, большее 10.

Оператор *continue* пропускает все последующие операторы тела цикла и передает управление на начало цикла.

```
…
int n=0,i;
while (n<10) {
n++;
scanf(“%d”, &i);
if (i%2==1) continue;
printf(“%d ”, i);
}
```

Цикл *while* выполнится 10 раз, все введенные в цикле четные значения будут выведены на экран, а при вводе нечетных значений вызов функции *printf*() будет пропущен.

3.2 Массивы в языке Си

Массивами называется набор однотипных элементов, каждый из которых имеет собственный номер – индекс. Нумерация элементов массива начинается с 0.

В Си можно задать массивы двумя способами. Статические массивы описываются следующим образом:

тип идентификатор [количество элементов] ;

Например, массив из ста целочисленных элементов может быть задан следующим образом: *int x*[100].

Динамические массивы описываются в Си следующим образом:

указатель идентификатор;

Например: *int* y*;

После того как динамический массив описан, необходимо выделить память для хранения элементов массива с помощью функции *malloc*.

Выделим для массива *y* память для хранения десяти элементов:

y=(*int**)*malloc*(*sizeof*(*int*)*10);

В качестве аргумента функции *malloc* используется значение, возвращаемое функцией *sizeof* (в примере возвращается размер, требуемый для хранения целочисленной переменной), умноженное на количество элементов. То есть размер памяти, требуемый для хранения элементов массива.

Элементы массива можно задать с клавиатуры, определить по заданной формуле или задать случайным образом, исходя из постановки задачи. Но в любом случае для обращения к элементам массива необходимо использовать конструкцию цикла, для того чтобы перебрать все элементы массива. Синтаксис обращения к элементу массива: идентификатор [индекс].

Приведем пример, определяющий элементы массива *x*, описанного выше случайным образом:

```
… int i;
for(i=0;i<100;i++) {
x[i]=rand()%20;
printf(“%d ”,x[i]);
}…
```

Записанный цикл задает значения всем элементам массива в интервале от [0;19] и выводит их на экран.

А элементы массива *y* зададим с клавиатуры:

```
for(i=0;i<10;i++) {
scanf("%d",&y[i]);
}…
```

Для вывода элементов массива так же, как и для ввода необходимо использовать цикл.

3.3 Пример выполнения индивидуального варианта

Алгоритмы работы с массивами подробно описаны в гл. 8 учебного пособия [1]. Используя эти алгоритмы, решим следующую задачу.

В одномерном массиве, состоящем из n вещественных элементов, вычислить:

А) *количество элементов массива, меньших среднего арифметического значения элементов массива*;

Б) *сумму индексов максимального и минимального элементов массива, значения которых попали в интервал* [*a*, *b*]. *Значения a* и *b задавать с клавиатуры*;

В) *выполнить циклический сдвиг влево элементов массива на n позиций*.

Количество элементов массива и значения элементов массива задавать с клавиатуры.

Запишем блок-диаграммы пунктов А, Б и В, выделив отдельно ввод элементов массива и вывод элементов массива (рис. 3.1).

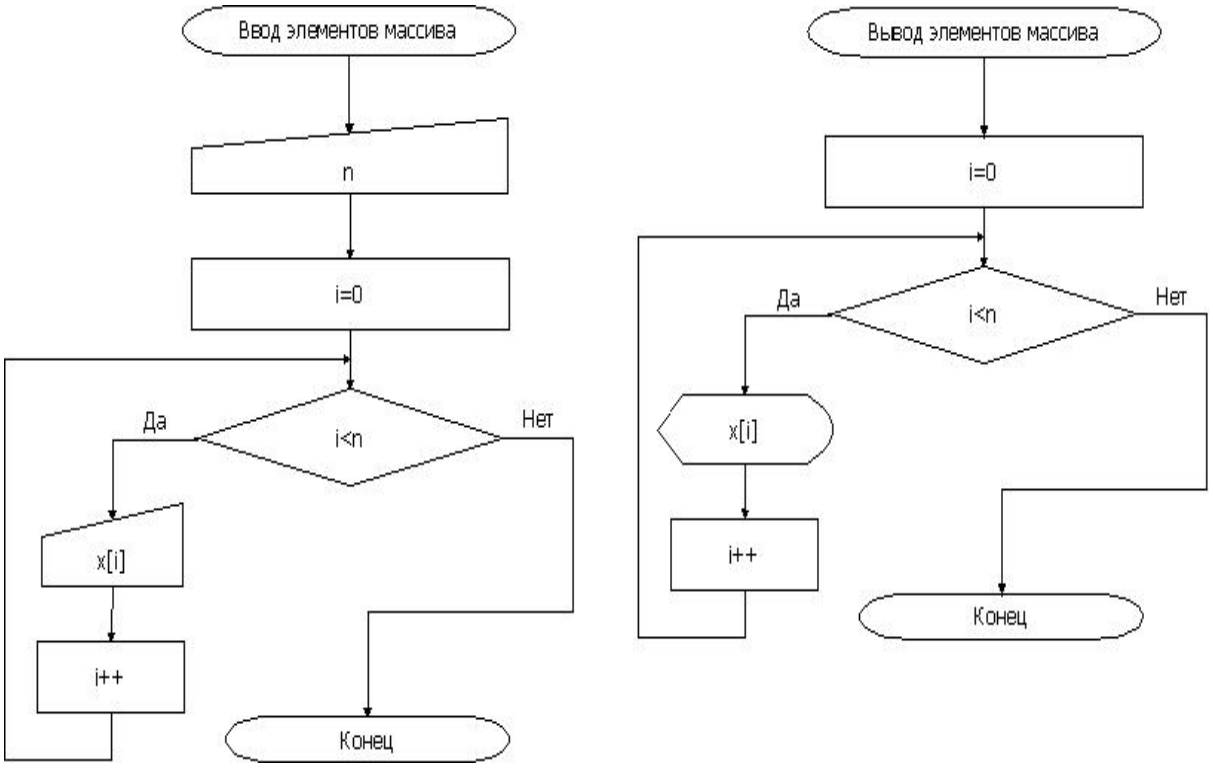


Рис. 3.1 – Ввод и вывод элементов массива

Задание А. Для решения задачи необходимо выполнить следующие действия: найти среднее арифметическое элементов массива, просуммировав все элементы массива и разделив полученную сумму на количество элементов. Затем поочередно сравнить все элементы массива с найденным значением, при этом увеличивая счетчик в том случае, если сравниваемый элемент меньше, чем среднее арифметическое (рис. 3.2).

Задание Б. Для решения задачи необходимо задать с клавиатуры значения границ интервала a и b. Для нахождения минимального и максимального значений воспользуемся классическим алгоритмом поиска: найдем первый встреченный элемент массива, значение которого $\in [a,b]$, и будем считать этот элемент и максимальным и минимальным элементом (рис. 3.3).

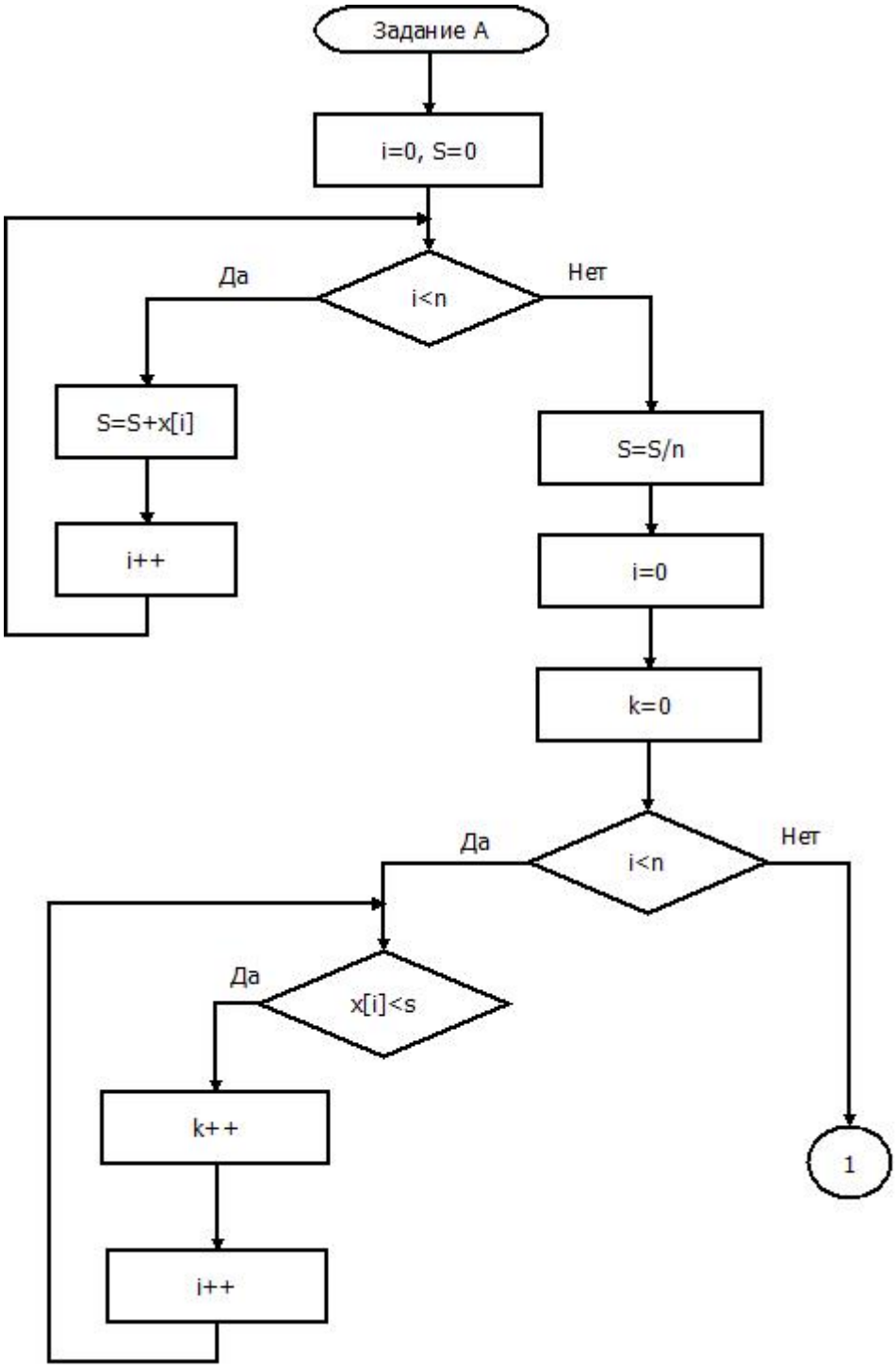


Рис. 3.2 – Алгоритм задания А

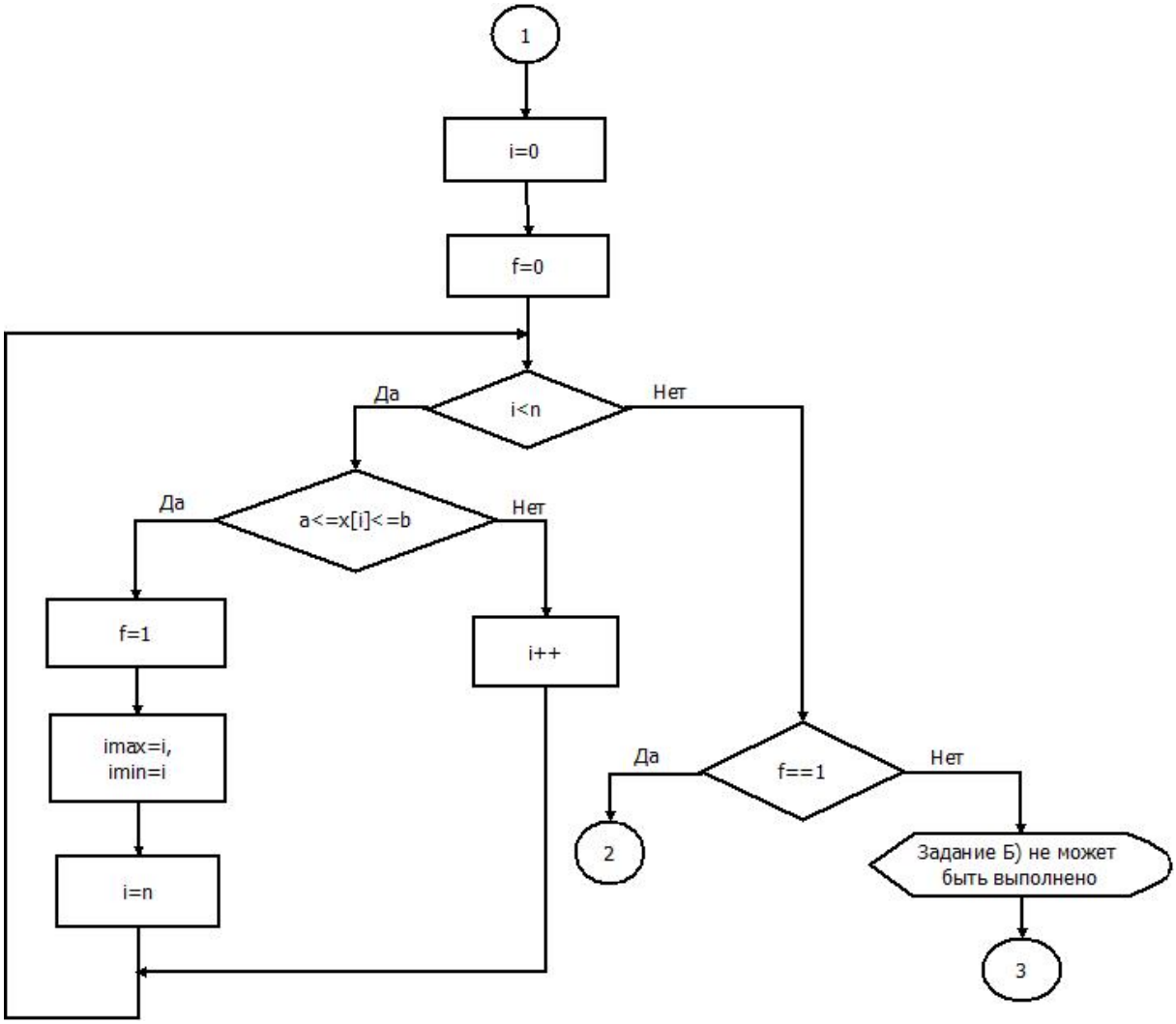


Рис. 3.3 – Нахождение первого встреченного элемента массива $x \in [a,b]$

Алгоритм поиска максимального и минимального значения просматривает элементы массива, значения которых принадлежат интервалу [a, b]. Если просматриваемый элемент с индексом i больше элемента с индексом $imax$, то изменяется значение индекса текущего максимального элемента. Аналогичным образом ищется индекс текущего минимального элемента (рис. 3.4).

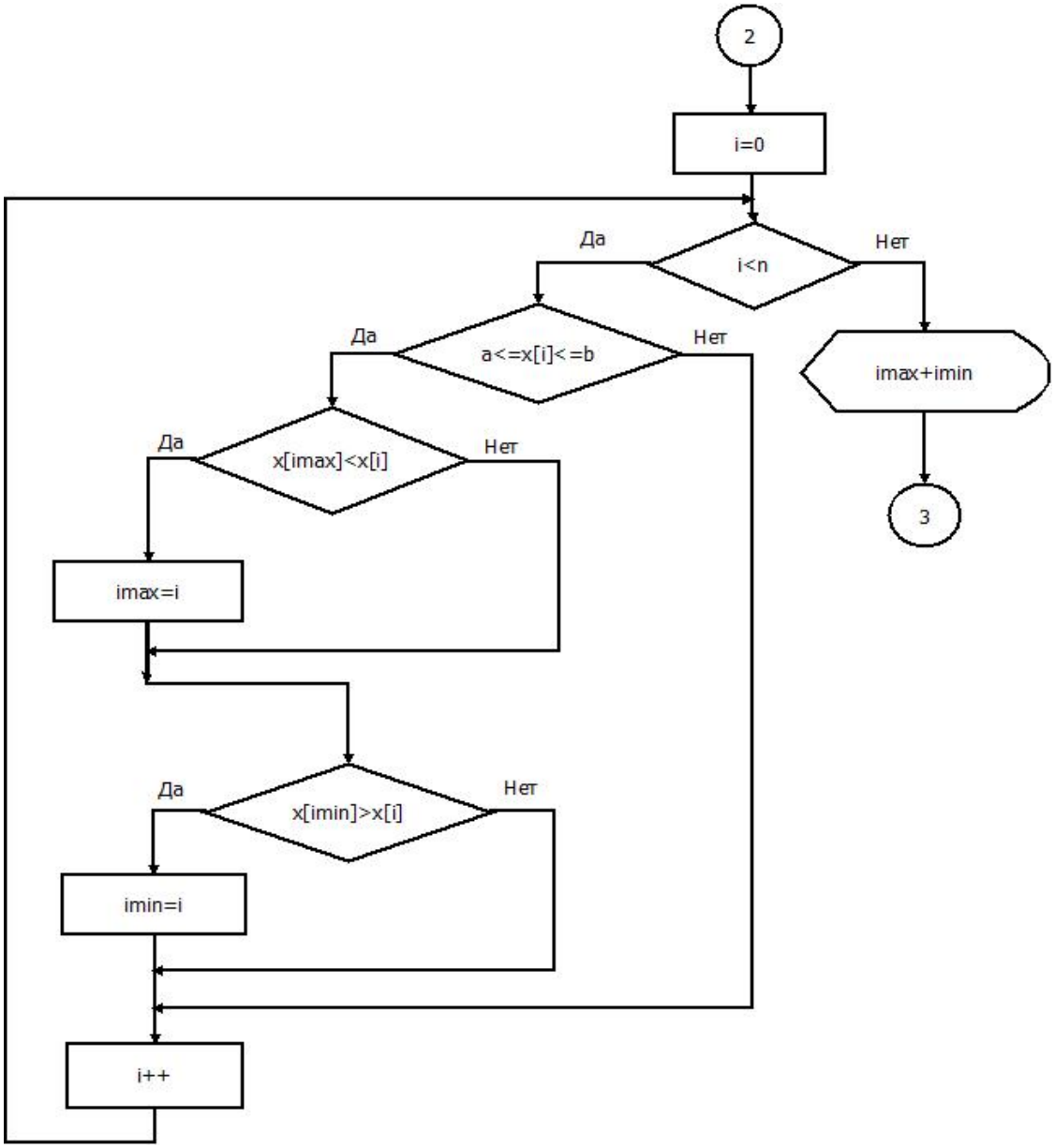


Рис. 3.4 – Поиск минимального и максимального значения

Задание В. Для наглядности, продемонстрируем циклический сдвиг элементов массива *x* на *k* элементов.

Пусть массив *x* = {1, 2, 3, 4, 5, 6}.

- *k* = 1. *x* = {2, 3, 4, 5, 6, 1}.
- *k* = 2. *x* = {3, 4, 5, 6, 1, 2}.
- *k* = 3. *x* = {4, 5, 6, 1, 2, 3}.
- *k* = 4. *x* = {5, 6, 1, 2, 3, 4}…

Очевидно, что сдвиги на 1 и на 7, 13, … дадут один и тот же результат, точно так же, как сдвиги на 3 и 9, 15, 21, … Тогда значение переменной k можно заменить на $k\%n$ (остаток от деления), где n – размерность массива.

Алгоритм можно реализовать, повторив k раз передвижение всех элементов массива влево на одну позицию. Таким образом, алгоритм выполнит $n*k$ перемещений.

Можно ограничиться только n перемещениями, но при этом необходимо ввести дополнительный массив. Попробуем найти закономерность расположения элементов массива в зависимости от k.

Пусть $k = 4$. Тогда, если $x_{исх} = \{1, 2, 3, 4, 5, 6\}$, то $x_{рез} = \{5, 6, 1, 2, 3, 4\}$. Закономерность изменения индексов показана в табл. 3.1.

Таблица 3.1 – Изменение индексов элементов

Индекс элемента до сдвига	Индекс элемента после сдвига
0	2
1	3
2	4
3	5
4	0
5	1

Нетрудно заметить, что если $i < k$ то $x[i] \rightarrow x[i + n – k]$, в противном случае $x[i] \rightarrow x[i – k]$. Блок-диаграмма алгоритма приведена рис. 3.5

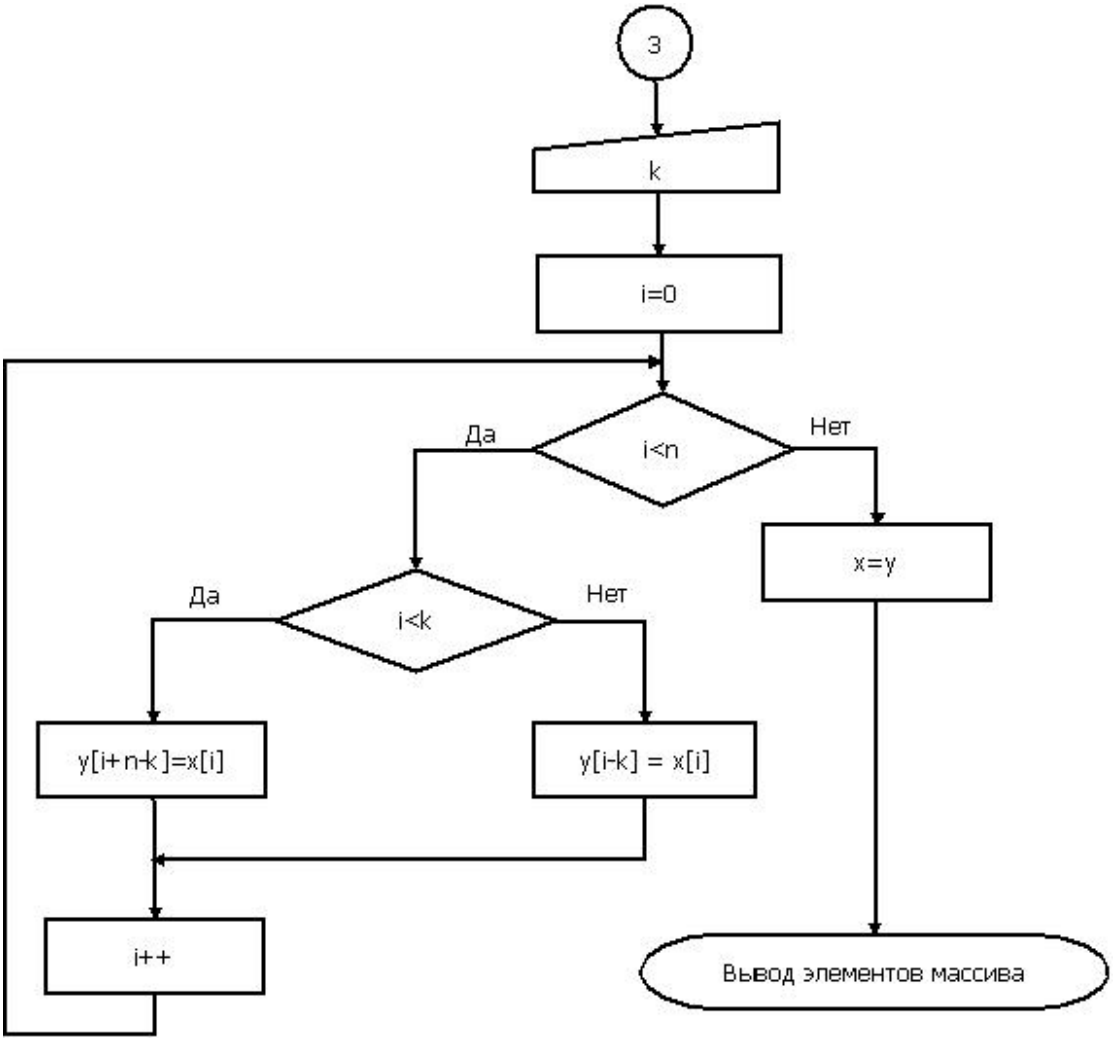


Рис. 3.5 – Циклический сдвиг элементов массива

Код программы, решающий поставленную задачу, может быть следующим:

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[]) {
int x[100];
int y[100];
int n,a,b,imax,imin;
system("chcp 1251");
```

```
printf("Введите размерность массива: ");
scanf("%d",&n);
float s = 0;
   int k=0, i;
   for(i=0;i<n;i++) {scanf("%d",&x[i]);
                          s = s+x[i];}
   s=s/n;
   for(i=0;i<n;i++){
        if(x[i]<s) k++;}
   printf("Задание А) Количество элементов, меньших
           среднего арифметического (%f) %d\n",s,k);
   int f=0;
   printf("Введите значения a и b: ");
   scanf("%d%d",&a,&b);
   for(i=0;i<n;i++)
   {   if (x[i]>=a&&x[i]<=b) { imax=i; imin = i;
                                  i=n; f= 1;}
   }
   if (f==1) {
        for(i=0;i<n;i++)
        { if(x[i]>=a&&x[i]<=b) { if(x[i]>x[imax]) imax=i;
                          if(x[i]<x[imin]) imin=i;
        }}
   printf("Задание Б) Сумма индексов  %d\n",imin+imax); }
   else printf("Задание Б не может быть выполнено.\n");
   printf("Введите величину сдвига: ");
   scanf("%d",&k);
   k = k%n;
   for(i=0;i<n;i++)
```

```
{if(i<k) y[i+n–k]=x[i];
        else y[i–k]=x[i];}
printf("Задание В) \n");

for(i=0;i<n;i++) { x[i] = y[i];          printf("%d ",x[i]);}
system(“pause”);
return 0; }
```

3.4 Порядок выполнения работы

1. Выбрать индивидуальный вариант.
2. Составить и записать алгоритм решения задачи.
3. Составить программу, реализующую записанный алгоритм.
4. Отладить программу.
5. Написать отчет о проделанной работе.

3.5 Содержание отчета

Отчет по лабораторной работе № 2 должен содержать:

- титульный лист,
- текст индивидуального варианта,
- алгоритмы решения пунктов А, Б, В,
- текст программы,
- результаты тестирования программы.

3.6 Индивидуальные варианты заданий лабораторной работы №2

Вариант 1

В одномерном массиве, состоящем из *n* целых элементов, вычислить:

А) сумму положительных элементов массива, стоящих на четных позициях;

Б) произведение индексов максимального и минимального элементов;

В) поменять местами первый положительный элемент с последним отрицательным.

Вариант 2

В одномерном массиве, состоящем из *n* целых элементов, вычислить:

А) сумму индексов четных (по значению) элементов массива;

Б) произведение элементов массива, расположенных между первым и вторым нулевыми элементами;

В) «перевернуть массив» – поменять первый элемент с последним, второй с предпоследним и т. д.

Вариант 3

В одномерном массиве, состоящем из *n* целых элементов, вычислить:

А) длину максимальной последовательности равных по значению элементов;

Б) количество элементов массива, являющихся *k*-й степенью двойки (k = 1, 2, 3, 4, 5);

В) найти сумму индексов отрицательных элементов.

Вариант 4

В одномерном массиве, состоящем из n целых элементов, вычислить:

А) произведение элементов, одновременно кратных 3 и 5;

Б) количество элементов массива, расположенных между минимальным и максимальным элементами;

В) выполнить циклический сдвиг вправо элементов массива на заданное число k.

Вариант 5

В одномерном массиве, состоящем из n целых элементов, вычислить:

А) количество нулевых элементов массива;

Б) сумму индексов максимального и минимального элементов массива, значения которых не попали в интервал $[a, b]$, значения a, b задавать с клавиатуры;

В) произведение элементов массива, равных заданному k.

Вариант 6

В одномерном массиве, состоящем из n целых элементов, вычислить:

А) сумму элементов массива с индексами, кратными 3;

Б) количество положительных элементов массива, значения которых не попали в интервал $[a, b]$;

В) произведение минимального и максимального элементов.

Вариант 7

В одномерном массиве, состоящем из n целых элементов, вычислить:

А) количество элементов массива, значения которых не превышают значение квадрата их индексов;

Б) произведение элементов массива, расположенных между первым и последним отрицательными элементами;

В) поменять местами последний нулевой элемент и первый максимальный элементы.

Вариант 8

В одномерном массиве, состоящем из n целых элементов, вычислить:

А) количество элементов массива, больших по значению, чем среднее арифметическое максимального и минимального элементов;

Б) произведение элементов массива, расположенных между первым и последним отрицательными элементами;

В) количество четных элементов, значения которых вошли в интервал $[a, b]$, значения a, b задавать с клавиатуры.

Вариант 9

В одномерном массиве, состоящем из n целых элементов, вычислить:

А) количество элементов массива, больших по значению, чем среднее арифметическое первого и последнего элементов;

Б) произведение элементов массива, расположенных между первым и последним положительными элементами;

В) количество четных элементов, значения которых вошли в интервал $[a, b]$, значения a, b задавать с клавиатуры.

Вариант 10

В одномерном массиве, состоящем из n целых элементов, вычислить:

А) вывести на печать элементы массива, значения которых больше их индексов;

Б) количество элементов массива, не меньших заданного элемента C, значение C вводить с клавиатуры;

В) заменить все положительные элементы массива квадратами индексов.

Вариант 11

В одномерном массиве, состоящем из *n* целых элементов, вычислить:

А) количество пар $x[i]$ и $x[i + 1]$, таких, что $x[i] < x[i + 1]$;

Б) произведение элементов массива, расположенных после последнего нулевого элемента;

В) заменить все отрицательные элементы максимальными.

Вариант 12

В одномерном массиве, состоящем из *n* целых элементов, вычислить:

А) количество пар $x[i]$ и $x[i + 1]$, таких, что $x[i] > x[i + 1]$;

Б) сумму элементов массива, расположенных после минимального элемента массива;

В) произведение элементов массива, расположенных до первого элемента массива, кратного 5.

Вариант 13

В одномерном массиве, состоящем из *n* целых элементов, вычислить:

А) количество элементов массива, кратных 7;

Б) количество пар одинаковых рядом стоящих элементов;

В) сумму элементов массива, расположенных между первым и последним нулевым элементами.

Вариант 14

В одномерном массиве, состоящем из *n* целых элементов, вычислить:

А) произведение элементов массива, не больших заданного элемента *C*, значение *C* вводить с клавиатуры;

Б) количество элементов массива, расположенных после максимального элемента массива;

В) сумму элементов массива, расположенных до первого нулевого элемента.

Вариант 15

В одномерном массиве, состоящем из n целых элементов, вычислить:

А) сумму отрицательных элементов массива;

Б) количество элементов массива, не меньших заданного элемента C, значение C вводить с клавиатуры;

В) заменить все отрицательные элементы массива их индексами.

Вариант 16

В одномерном массиве, состоящем из n целых элементов, вычислить:

А) количество четных (по значению) элементов массива;

Б) произведение элементов массива, расположенных между первым и последним отрицательными элементами;

В) количество элементов, значения которых вошли в интервал $[a, b]$, значения a, b задавать с клавиатуры.

Вариант 17

В одномерном массиве, состоящем из n целых элементов, вычислить:

А) сумму элементов массива с нечетными индексами;

Б) количество положительных элементов массива, расположенных между элементами с индексами a и b, значения a, b задавать с клавиатуры;

В) произведение минимального и максимального элементов.

Вариант 18

В одномерном массиве, состоящем из n целых элементов, вычислить:

А) количество нечетных (по значению) элементов массива;

Б) произведение элементов массива, расположенных между первым и последним отрицательными элементами;

В) поменять местами последний нулевой элемент и первый минимальный элементы.

Вариант 19

В одномерном массиве, состоящем из n целых элементов, вычислить:

А) сумму отрицательных элементов;

Б) количество элементов массива, расположенных между первым и последним четными положительными элементами;

В) поменять местами последний минимальный элемент и первый максимальный.

Вариант 20

В одномерном массиве, состоящем из вещественных элементов, вычислить:

А) произведение элементов с нечетными значениями;

Б) количество элементов массива, расположенных между минимальным и максимальным элементами;

В) поменять порядок следования элементов в массиве (поменять местами i-й элемент с $n - i$-м элементом).

4 ЛАБОРАТОРНАЯ РАБОТА № 3 «ОБРАБОТКА МАТРИЦ. ФУНКЦИИ»

Цель работы: закрепить навыки работы с двумерными массивами, научиться оформлять функции.

4.1 Матрицы в языке Си

В языке Си двумерные массивы можно определять двумя способами. Для описания статической матрицы используется следующее обозначение:

тип идентификатор [количество строк][количество столбцов].

Например, для описания целочисленной матрицы *X*, содержащей 10 строк и 5 столбцов:

```
int X[10][5];
```

Для описания динамической матрицы, размеры которой может задавать пользователь, в программе на Си выполняются следующие описания и действия:

- описывается указатель на указатель;
- выделяется память для хранения указателей на строки;
- выделяется память для хранения элементов матрицы.

Например, для описания динамической целочисленной матрицы *Y*, которая содержит *n* строк и *m* столбцов, в программе необходимо выполнить следующие действия:

```
int **Y;
Y=(int**)malloc(sizeof(int*)*n);
for(i=0;i<n;i++)
Y[i]=(int*)malloc(sizeof(int)*m);
```

Предполагается, что значения переменных *n* и *m* уже определены (например, ранее введены с клавиатуры).

Для обращения к элементу матрицы необходимо указать номер строки и номер столбца. Например, обращение к элементу матрицы *X*, расположенному в нулевой строке и третьем столбце, будет выглядеть как *x*[0][3], а обращение к элементу матрицы *Y*, расположенному в строке с номером *i* и в столбце с номером *j* выглядит как *Y*[*i*][*j*].

Для перебора всех элементов матриц необходимо выполнить два вложенных цикла, при этом если внешний цикл перебирает номера строк, а внутренний – номера столбцов, то просмотр матрицы осуществляется по строкам. Если же внешний цикл перебирает номера столбцов, а внутренний – номера строк, то матрица просматривается по столбцам. Пример, рассмотренный ниже, определяет элементы матрицы *Y* случайным образом и выводит их на экран.

```
for(i=0;i<n;i++) {
 for(j=0;j<m;j++){
   Y[i][j]=rand()%10;
   printf("%4d", Y[i][j]); }
printf("\n");}
```

Примеры работы с матрицами рассмотрены в гл. 8 учебного пособия [1].

4.2 Квадратные матрицы

Матрицы, имеющие одинаковое количество строк и столбцов, называются квадратными матрицами. В квадратной матрице, состоящей из *n* строк и столбцов, главной диагональю называется массив из элементов матрицы, которые имеют номера [*i*][*i*], при этом переменная *i* изменяется в диапазоне от 0 до $n-1$. Побочной диагональю называется массив

из элементов матрицы с номерами [*i*][*n* – *i* – 1], значение переменной *i* так же изменяется от 0 до *n* – 1.

Используя эту информацию, можно рассмотреть различные части квадратной матрицы. Например, для того чтобы обратиться к элементам матрицы, расположенным выше главной диагонали, необходимо изменять счетчики циклов так, как показано в примере ниже:

```
for(i=0;i<n-1;i++)
   for(j=i+1;j<n;j++).
```

Для того чтобы обратиться к элементам матрицы, расположенным ниже побочной диагонали, циклы просмотра матрицы будут записаны следующим образом:

```
for(i=1;i<n;i++)
   for(j=n-i;j<n;j++).
```

4.3 Функции в языке Си

Для осуществления принципов структурного программирования в Си существует возможность создания пользовательских функций. Основными целями использования функций являются уменьшение кода и возможность использования уже написанного кода в других программах. Синтаксис описания функции в языке Си:

тип идентификатор (список формальных аргументов) { тело функции}

Например, функция печати матрицы может выглядеть следующим образом:

```
void show (int **x, int n, int m){
int i,j;
```

```
for(i=0;i<n;i++){
for(j=0;j<m;j++)
printf("%4d", x[i][j]); }
printf("\n"); }
```

Функция, написанная таким образом, может использоваться для вывода на экран элементов любой целочисленной матрицы.

Все функции в языке Си пишутся вне функции *main*. Если функция описывается после *main* либо располагается в отдельном файле, необходимо до ее использования описать так называемый прототип функции. Синтаксис описания прототипа выглядит следующим образом:

тип идентификатор (перечисление типов формальных аргументов);

Прототип функции *show*, описанной выше, будет выглядеть следующим образом:

```
void show(int*, int, int);
```

В большинстве случаев функции должны вернуть какое-либо рассчитанное в теле функции значение в вызывающую функцию (например, в *main*). Для возврата значения из функции используется оператор *return*. Например, необходимо написать функцию, возвращающую сумму элементов заданной строки матрицы, в этом случае функция может быть написана следующим образом:

```
// x – матрица, k – номер заданной строки, m – количество столбцов
int SumI(int **x, int k, int m){
int S = 0,i;
for(i=0;i<m;i++)
S+=x[k][i];
return S; }
```

Любая функция заканчивает свою работу по достижении оператора *return*, либо, если функция не возвращает значения в вызывающую функцию, по достижении завершающей фигурной скобки.

Для вызова функции указывают имя функции и в круглых скобках перечисляют имена фактических аргументов. Например, вызов функции *show* может быть следующим: *show*(*p*,5,7). При условии, что в программе определена матрица *p*, состоящая из пяти строк и семи столбцов. А вызов функции *SumI* может выглядеть следующим образом:

*int i*1 = *SumI*(*Y*,2,*m*);

Значение, возвращаемое оператором *return*, запишется в переменную *i*1.

Более подробно правила работы с функциями описаны в гл. 7 учебного пособия [1].

4.4 Пример выполнения индивидуального варианта

Используемые обозначения:

*A*10(*X*) – сумма элементов матрицы *X*;

*A*11(*X*) – максимальный элемент матрицы *X*;

*A*12(*X*) – минимальный элемент матрицы *X*.

Напишите программу, которая вычисляет значение по заданной формуле:

$$p = \begin{cases} A10(X) + A10(Y) + A12(X), \text{ если } A11(X) < A11(Y), \\ A10(X) - A10(Y) - A12(Y), \text{ если } A11(X) > A11(Y), \\ A10(Y) - A10(X) + A12(X), \text{ в противном случае.} \end{cases}$$

где *X* – матрица размерности [5x10], *Y* – матрица размерности [6x8]. Элементы матриц целые числа, задаются случайным образом.

Для решения задачи необходимо написать три функции:

- функцию, возвращающую сумму элементов матрицы;
- функцию, возвращающую минимальный элемент матрицы;
- функцию, возвращающую максимальный элемент матрицы.

Дадим функциям имена, совпадающие с используемыми обозначениями. Для начала необходимо определить список аргументов для каждой функции. Так как в ходе решения задачи необходимо найти суммы элементов обеих заданных матриц, а также найти их максимальные и минимальные значения, аргументами функций будут матрицы и их размерности. Алгоритмы функций поиска суммы элементов, минимального значения и максимального значения матрицы представлены на рисунках 4.1–4.3.

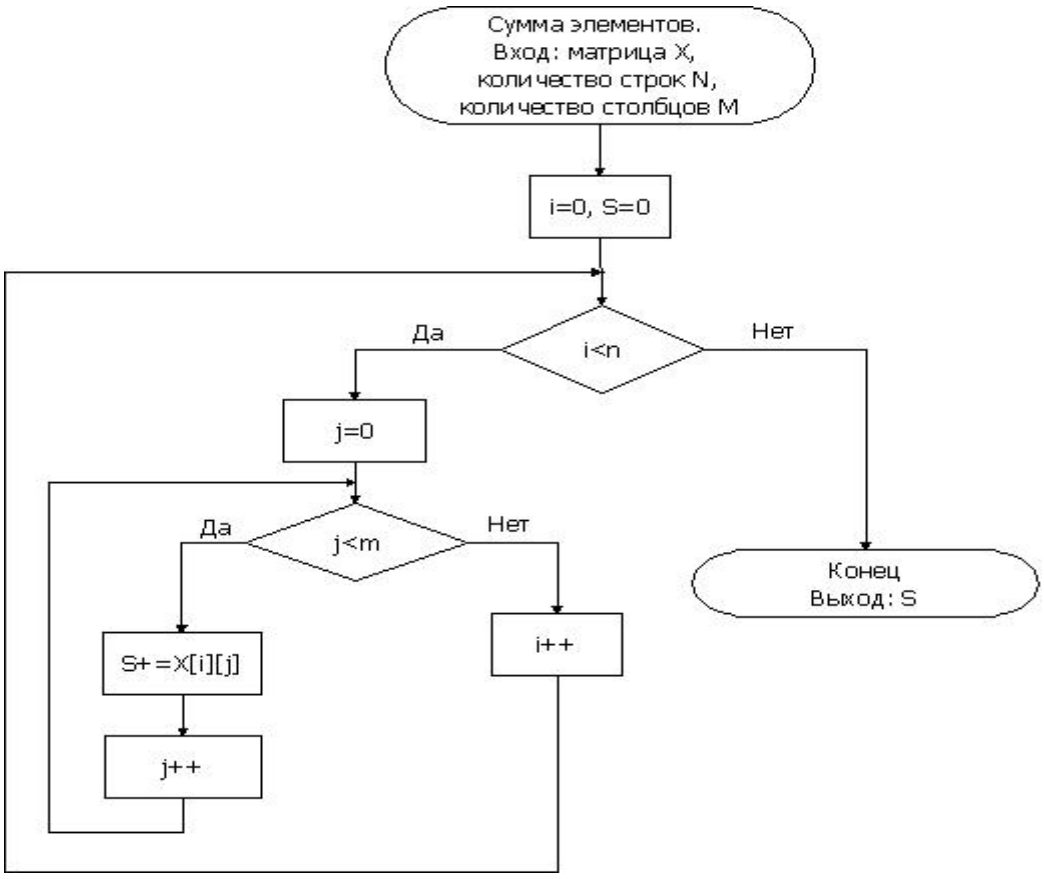


Рис. 4.1 – Поиск суммы элементов матрицы

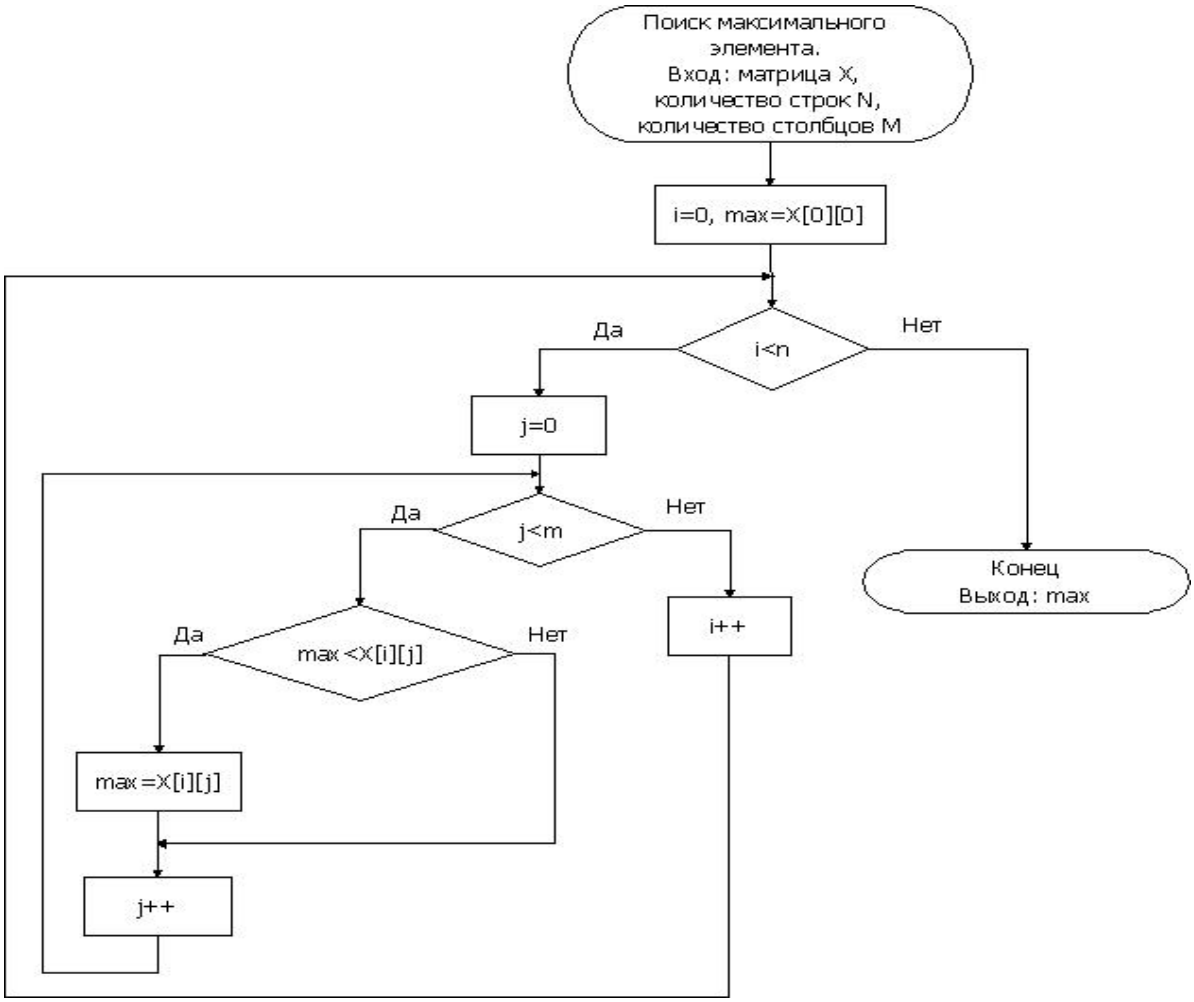


Рис. 4.2 – Поиск максимального элемента матрицы

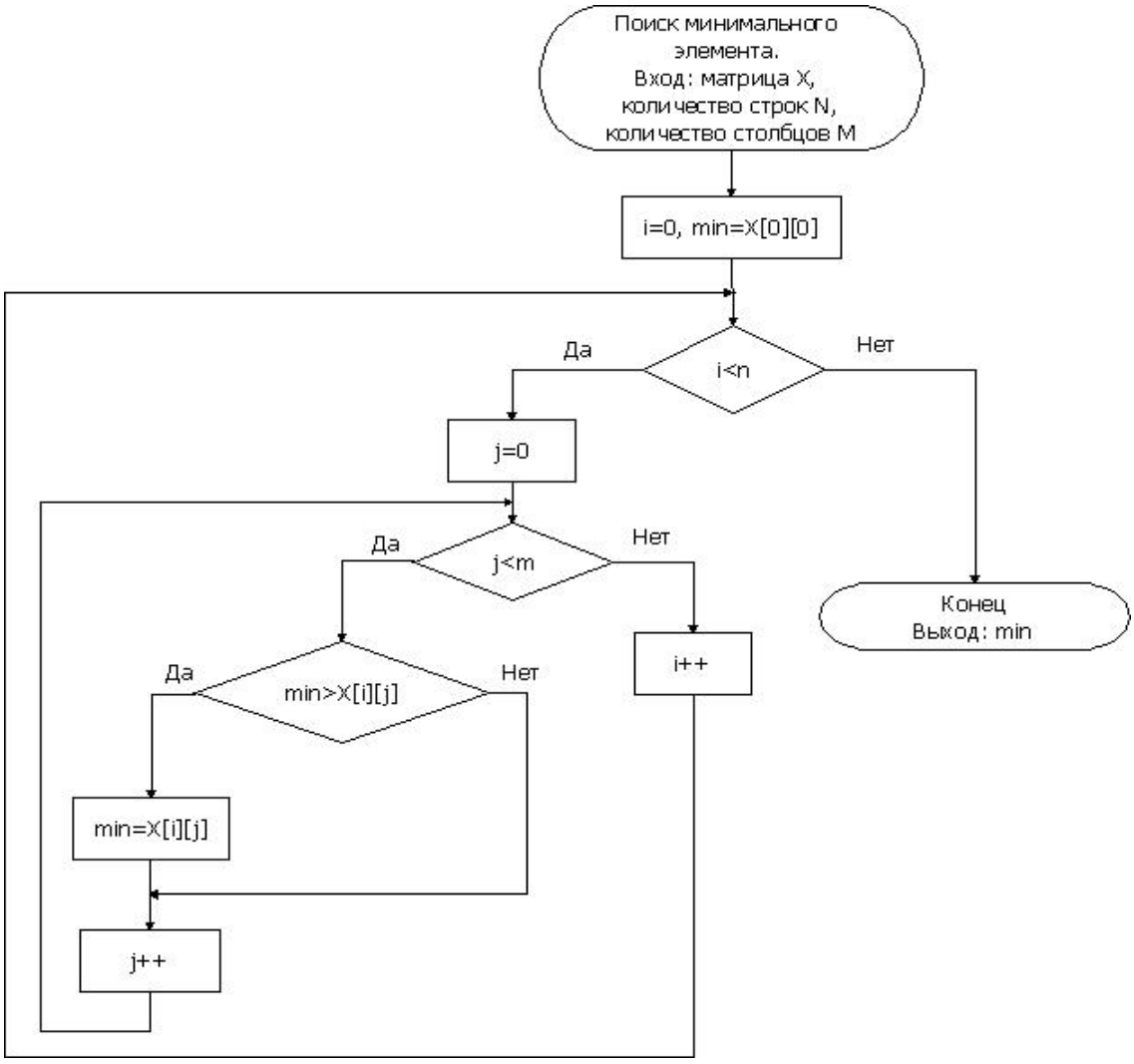


Рис. 4.3 – Поиск минимального элемента

Текст программы по индивидуальному варианту приведен ниже:

```
#include <stdio.h>
#include <stdlib.h>
int A10(int** z, int n, int m)
{int s =0;
int i,j;
for(i=0;i<n;i++)
 for(j=0;j<m;j++)
```

```
   s+=z[i][j];
return s; }
int A11(int** z, int n, int m)
{int max =z[0][0];
int i,j;
for(i=0;i<n;i++)
 for(j=0;j<m;j++)
   if (z[i][j]>max) max = z[i][j];
return max; }
int A12(int** z, int n, int m)
{int min =z[0][0];
int i,j;
for(i=0;i<n;i++)
 for(j=0;j<m;j++)
   if (z[i][j]<min) min = z[i][j];
return min; }
int main(int argc, char *argv[]) {
  int **x, **y;
  int nx = 5, mx = 10, ny = 6, my = 8;
  int i,j;
  x = (int**) malloc(sizeof(int*)*nx);
  for(i=0;i<nx;i++) x[i] = (int*) malloc(sizeof(int)*mx);
  y = (int**) malloc(sizeof(int*)*ny);
  for(i=0;i<ny;i++) y[i] = (int*) malloc(sizeof(int)*my);
  system("chcp 1251");
  srand(time(NULL));
  printf("Матрица X \n");
  for(i=0;i<nx;i++){
        for(j=0;j<mx;j++){
```

```
                x[i][j]=rand()%55;
                printf("%3d",x[i][j]);  }
    printf("\n");    }
printf("Матрица Y \n");
for(i=0;i<ny;i++){
    for(j=0;j<my;j++){
                y[i][j]=rand()%55;
                printf("%3d",y[i][j]);   }
     printf("\n");  }
int a10x, a11x, a12x, a10y, a11y, a12y;
a10x = A10(x,nx,mx);
a10y = A10(y,ny,my);
a11x = A11(x,nx,mx);
a12x = A12(x,nx,mx);
a11y = A11(y,ny,my);
a12y = A12(y,ny,my);
printf("Сумма элементов матрицы X %d\n",a10x);
printf("Сумма элементов матрицы Y %d\n",a10y);
printf("Максимальный элемент матрицы X %d\n",a11x);
printf("Минимальный элемент матрицы  X %d\n",a12x);
printf("Максимальный элемент матрицы Y %d\n",a11y);
printf("Минимальный элемент матрицы  Y %d\n",a12y);
int p;
if (a11x<a11y)
   p =  a10x+a10y+a12y;
if (a11x>a11y)
   p =  a10x - a10y - a12y;
    else p = a10y - a10x+a12x;
 printf("Значение  p %d\n",p);
```

```
system("PAUSE");
return 0; }
```

4.5 Порядок выполнения работы

1. Выбрать индивидуальный вариант.
2. Составить и записать алгоритм решения задачи.
3. Составить программу, реализующую записанный алгоритм.
4. Отладить программу.
5. Написать отчет о проделанной работе.

4.6 Содержание отчета

Отчет по лабораторной работе № 3 должен содержать:

- титульный лист,
- текст индивидуального варианта,
- алгоритмы написанных функций,
- текст программы,
- результаты тестирования программы.

Пример оформленного отчета приведен в приложении А.

4.7 Индивидуальные варианты заданий лабораторной работы № 3

Используемые обозначения:

$A1(X)$ – сумма элементов матрицы X с нечетными значениями.

$A2(X)$ – сумма элементов матрицы X с четными значениями.

$A3(X)$ – количество нулевых элементов матрицы X.

$A4(X)$ – среднее арифметическое элементов матрицы X.

$A5(X)$ – количество элементов матрицы X с нечетными значениями.

$A6(X)$ – количество элементов матрицы X с четными значениями.

$A7(X,i)$ – сумма элементов строки с номером i матрицы X.

$A8(X,i)$ – сумма элементов столбца с номером i матрицы X.

$A9(X)$ – количество положительных элементов матрицы X.

$A10(X)$ – количество отрицательных элементов матрицы X.

$A11(X)$ – сумма положительных элементов матрицы X.

$A12(X)$ – сумма отрицательных элементов матрицы X.

Вариант 1

Вычислите значение переменной p, если X – целочисленная матрица размерности [5 x 7], Y – целочисленная матрица размерности [7 x 5]. Переменная p определяется следующим образом:

$$p = \begin{cases} \dfrac{A2(X) + A3(Y)}{A2(X)}, \text{ если } A4(X) < A4(Y), \\ \dfrac{A2(X) - A3(Y)}{A2(X)}, \text{ если } A4(X) > A4(Y), \\ \dfrac{A2(X) + A3(Y)}{A2(X)}, \text{ в противном случае.} \end{cases}$$

Вариант 2

Вычислите значение переменной p, если X – целочисленная матрица размерности [7 x 4], Y – целочисленная матрица размерности [4 x 7]. Переменная p определяется следующим образом:

$$p = \begin{cases} \sqrt{A5(X) + A5(Y)} + A6(X), \text{ если } A4(X) < A4(Y), \\ \sqrt{A6(X) + A6(Y)} + A5(X), \text{ если } A4(X) > A4(Y), \\ \sqrt{A5(X) + A5(Y) + A6(X)}, \text{ в противном случае.} \end{cases}$$

Вариант 3

Вычислите значение переменной p, если X – целочисленная матрица размерности [8 x 9], Y – целочисленная матрица размерности [9 x 8]. Переменная p определяется следующим образом:

$$p = \begin{cases} \dfrac{A7(X,2) + A7(Y,2)}{A8(X,0)}, \text{ если } A9(X) < A9(Y), \\ \dfrac{A7(X,1) + A7(Y,1)}{A8(X,2)}, \text{ если } A9(X) > A9(Y), \\ \dfrac{A7(X,3) + A7(Y,3)}{A8(X,4)}, \text{ в противном случае.} \end{cases}$$

Вариант 4

Вычислите значение переменной *p*, если *X* – целочисленная матрица размерности [5 x 6], *Y* – целочисленная матрица размерности [3 x 10]. Переменная *p* определяется следующим образом:

$$p = \begin{cases} A11(X) - A12(Y), \text{ если } A10(X) < A10(Y), \\ A12(X) - A11(Y), \text{ если } A10(X) > A10(Y), \\ A11(X) + \sqrt{|A12(Y)|}, \text{ в противном случае.} \end{cases}$$

Вариант 5

Вычислите значение переменной *p*, если *X* – целочисленная матрица размерности [5 x 7], *Y* – целочисленная матрица размерности [7 x 5]. Переменная *p* определяется следующим образом:

$$p = \begin{cases} \dfrac{A1(X) + A1(Y)}{A2(X)}, \text{ если } A3(X) < A3(Y), \\ \dfrac{A1(X) - A1(Y)}{A2(X)}, \text{ если } A3(X) > A3(Y), \\ \dfrac{A1(X) + A2(Y)}{A2(X)}, \text{ в противном случае.} \end{cases}$$

Вариант 6

Вычислите значение переменной *p*, если *X* – целочисленная матрица размерности [7 x 4], *Y* – целочисленная матрица размерности [4 x 7]. Переменная *p* определяется следующим образом:

$$p = \begin{cases} \sqrt{|A7(X,0) + A7(Y,0)|} + A6(X), \text{если } A1(X) < A1(Y), \\ \sqrt{|A6(X) + A6(Y)|} + A7(X,1), \text{если } A1(X) > A1(Y), \\ \sqrt{|A7(X,3) + A7(Y,3) + A6(X)|}, \text{в противном случае.} \end{cases}$$

Вариант 7

Вычислите значение переменной *p*, если *X* – целочисленная матрица размерности [8 x 9], *Y* – целочисленная матрица размерности [9 x 8]. Переменная *p* определяется следующим образом:

$$p = \begin{cases} \dfrac{A9(X) + A9(Y)}{A8(X,0)}, \text{если } A7(X,0) < A8(Y,0), \\ \dfrac{A9(X) + A9(Y)}{A8(X,2)}, \text{если } A7(X,0) > A8(Y,0), \\ \dfrac{A9(X) + A9(Y)}{A8(X,4)}, \text{в противном случае.} \end{cases}$$

Вариант 8

Вычислите значение переменной *p*, если *X* – целочисленная матрица размерности [5 x 6], *Y* – целочисленная матрица размерности [3 x 10]. Переменная *p* определяется следующим образом:

$$p = \begin{cases} A10(X) - A12(Y), \text{если } A2(X) < 0, \\ A10(X) - A11(Y), \text{если } A2(X) > 0, \\ A10(X) + \sqrt{|A12(Y)|}, \text{в противном случае.} \end{cases}$$

Вариант 9

Вычислите значение переменной *p*, если *X* – целочисленная матрица размерности [5 x 7], *Y* – целочисленная матрица размерности [7 x 5]. Переменная *p* определяется следующим образом:

$$p = \begin{cases} \dfrac{A10(X) + A10(Y)}{A11(X)}, \text{ если } A2(X) < A2(Y), \\ \dfrac{A10(X) - A10(Y)}{A11(X)}, \text{ если } A2(X) > A2(Y), \\ \dfrac{A10(Y) - A10(X)}{A11(X)}, \text{ в противном случае.} \end{cases}$$

Вариант 10

Вычислите значение переменной *p*, если *X* – целочисленная матрица размерности [7 x 4], *Y* – целочисленная матрица размерности [4 x 7]. Переменная *p* определяется следующим образом:

$$p = \begin{cases} \sqrt{|A1(X) + A1(Y)|} + A6(X), \text{ если } A5(X) < A5(Y), \\ \sqrt{|A1(X) + A1(Y)|} + A5(X), \text{ если } A5(X) > A5(Y), \\ \sqrt{|A1(X) + A1(Y) + A6(X)|}, \text{ в противном случае.} \end{cases}$$

Вариант 11

Вычислите значение переменной *p*, если *X* – целочисленная матрица размерности [8 x 9], *Y* – целочисленная матрица размерности [9 x 8]. Переменная *p* определяется следующим образом:

$$p = \begin{cases} \dfrac{A4(X) + A4(Y)}{A8(X,5)}, \text{ если } A6(X) < A6(Y), \\ \dfrac{A4(X) + A4(Y)}{A8(X,6)}, \text{ если } A6(X) > A6(Y), \\ \dfrac{A4(X) + A4(Y)}{A8(X,7)}, \text{ в противном случае.} \end{cases}$$

Вариант 12

Вычислите значение переменной *p*, если *X* – целочисленная матрица размерности [5 x 6], *Y* – целочисленная матрица размерности [3 x 10]. Переменная *p* определяется следующим образом:

$$p = \begin{cases} A7(X,0) - A8(Y,0), \text{ если } A9(X) < 10, \\ A7(X,1) - A8(Y,1), \text{ если } A9(X) > 10, \\ A7(X,2) + \sqrt{|A7(Y,2)|}, \text{ в противном случае.} \end{cases}$$

Вариант 13

Вычислите значение переменной *p*, если *X* – целочисленная матрица размерности [5 x 7], *Y* – целочисленная матрица размерности [7 x 5]. Переменная *p* определяется следующим образом:

$$p = \begin{cases} \dfrac{A3(X) + A3(Y)}{A4(X)}, \text{ если } A5(X) < 15, \\ \dfrac{A3(X) - A3(Y)}{A4(X)}, \text{ если } A5(X) > 15, \\ \dfrac{A4(X) + A4(Y)}{A3(X)}, \text{ в противном случае.} \end{cases}$$

Вариант 14

Вычислите значение переменной *p*, если *X* – целочисленная матрица размерности [7 x 4], *Y* – целочисленная матрица размерности [4 x 7]. Переменная *p* определяется следующим образом:

$$p = \begin{cases} \sqrt{|A7(X,0) + A7(Y,0)|} + A10(X), \text{ если } A7(X,0) < A7(X,1), \\ \sqrt{|A10(X) + A10(Y)|} + A7(X,1), \text{ если } A7(X,0) > A7(X,1), \\ \sqrt{|A7(X,0) + A7(Y,0) + A10(X)|}, \text{ в противном случае.} \end{cases}$$

Вариант 15

Вычислите значение переменной *p*, если *X* – целочисленная матрица размерности [8 x 9], *Y* – целочисленная матрица размерности [9 x 8]. Переменная *p* определяется следующим образом:

$$p = \begin{cases} \dfrac{A11(X) + A11(Y)}{A8(X,0)}, \text{ если } A8(X,0) < A8(X,1), \\ \dfrac{A11(X) + A12(Y)}{A8(X,2)}, \text{ если } A8(X,0) > A8(X,1), \\ \dfrac{A12(X) + A11(Y)}{A8(X,1)}, \text{ в противном случае.} \end{cases}$$

Вариант 16

Вычислите значение переменной *p*, если *X* – целочисленная матрица размерности [5 x 6], *Y* – целочисленная матрица размерности [3 x 10]. Переменная *p* определяется следующим образом:

$$p = \begin{cases} A1(X) - A1(Y), \text{ если } A9(Y) < 7, \\ A1(X) - A2(Y), \text{ если } A9(Y) > 7, \\ A1(X) + \sqrt{|A2(Y)|}, \text{ в противном случае.} \end{cases}$$

Вариант 17

Вычислите значение переменной *p*, если *X* – целочисленная матрица размерности [7 x 4], *Y* – целочисленная матрица размерности [4 x 7]. Переменная *p* определяется следующим образом:

$$p = \begin{cases} (A3(X) + A3(Y))^2 + A5(X), \text{ если } A6(X) < A6(Y), \\ (A3(X) + A3(Y))^2 + A5(Y), \text{ если } A6(X) > A6(Y), \\ (A3(X) + A3(Y) + A6(X))^2, \text{ в противном случае.} \end{cases}$$

Вариант 18

Вычислите значение переменной *p*, если *X* – целочисленная матрица размерности [8 x 9], *Y* – целочисленная матрица размерности [9 x 8]. Переменная *p* определяется следующим образом:

$$p = \begin{cases} \dfrac{A4(X) + A5(Y)}{A7(X,5)}, \text{ если } A7(X,2) < A7(X,3), \\ \dfrac{A4(X) + A5(Y)}{A7(X,6)}, \text{ если } A7(X,2) > A7(X,3), \\ \dfrac{A4(X) + A5(Y)}{A7(X,7)}, \text{ в противном случае.} \end{cases}$$

Вариант 19

Вычислите значение переменной *p*, если *X* – целочисленная матрица размерности [5 x 6], *Y* – целочисленная матрица размерности [3 x 10]. Переменная *p* определяется следующим образом:

$$p = \begin{cases} A8(X,0) - A7(Y,0), \text{ если } A4(X) < 0, \\ A8(X,3) - A7(Y,1), \text{ если } A4(X) > 0, \\ A8(X,3) + \sqrt{|A7(Y,2)|}, \text{ в противном случае.} \end{cases}$$

Вариант 20

Вычислите значение переменной *p*, если *X* – целочисленная матрица размерности [5 x 7], *Y* – целочисленная матрица размерности [7 x 5]. Переменная *p* определяется следующим образом:

$$p = \begin{cases} \dfrac{A2(X) + A2(Y)}{A3(X)}, \text{ если } A7(X,0) < 0, \\ \dfrac{A2(X) - A3(Y)}{A2(X)}, \text{ если } A7(X,0) > 0, \\ \dfrac{A3(X) + A3(Y)}{A2(X)}, \text{ в противном случае.} \end{cases}$$

5 ЛАБОРАТОРНАЯ РАБОТА № 4 «РАБОТА СО СТРОКАМИ»

Цель работы: закрепить навыки работы со строковыми переменными.

5.1 Строки в языке Си

Строки в языке Си являются массивами символов. В программе строка может быть задана статически следующим образом: *char A*[15]. При описании динамической строки выполняются такие же действия, как при работе с динамическими массивами:

- описывается указатель на символ;
- выделяется память для хранения строки.

Например, это может быть выполнено следующим образом:

char **B*;

B = (*char**) *malloc*(*sizeof*(*char*)*15);

Любая строка в Си заканчивается символом ‘\0’, поэтому строки *A* и *B*, описанные ранее, могут содержать только 14 символов. Если строка задана статически, символ ‘\0’ устанавливается автоматически. В случае работы с динамической строкой о наличии символа ‘\0’ должен позаботиться программист.

Стандартные функции работы со строками описаны в гл. 8 учебного пособия [1].

5.2 Пример выполнения индивидуального варианта

Дана строка символов. Группы символов, разделенные пробелами (одним или несколькими) и не содержащие пробелов внутри себя, будем называть словами. Удалить из строки слова, содержащие символы, не являющиеся буквами.

Для выделения слов строки воспользуемся функцией *strtok*, ее работа подробно описана в учебном пособии. Далее, в каждом выделенном слове проверим, являются ли символы слова буквами с помощью функции *isalpha*.

Если при проверке будет встречен хотя бы один символ, не являющийся буквой, слово будет удалено из строки. Так как в языке Си нет стандартных функций удаления символов, то по мере нахождения слов, содержащих не буквы, будем переписывать конец строки, замещая удаляемое слово. При реализации алгоритма:

- в переменной *j* хранится номер символа, с которого нужно начинать перемещения;
- в переменной *k* хранится величина, на значение которой нужно передвинуть символы;
- в переменной *j*1 накапливается сумма длин удаляемых слов.

Алгоритм решения задачи представлен на рис. 5.1–5.3.

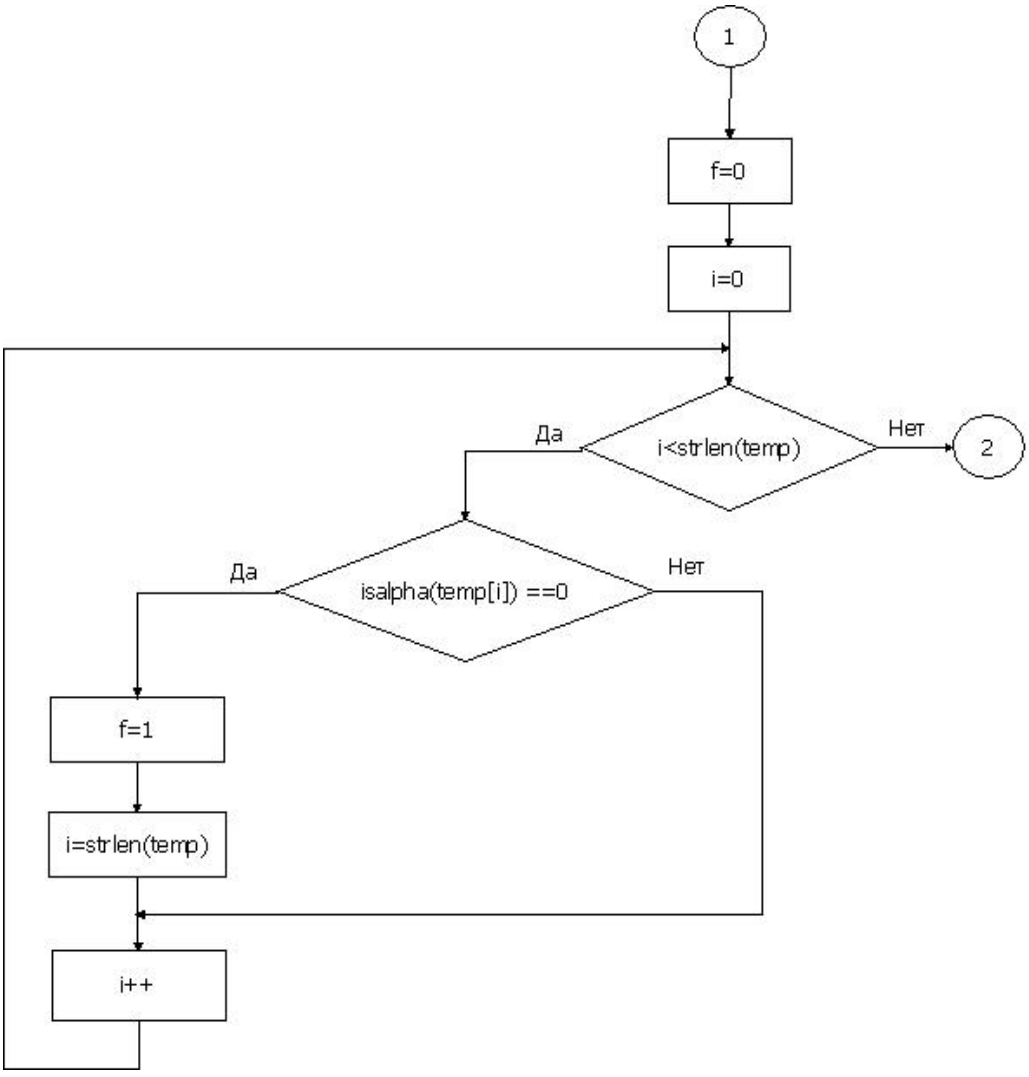


Рис. 5.1 – Алгоритм проверки символов

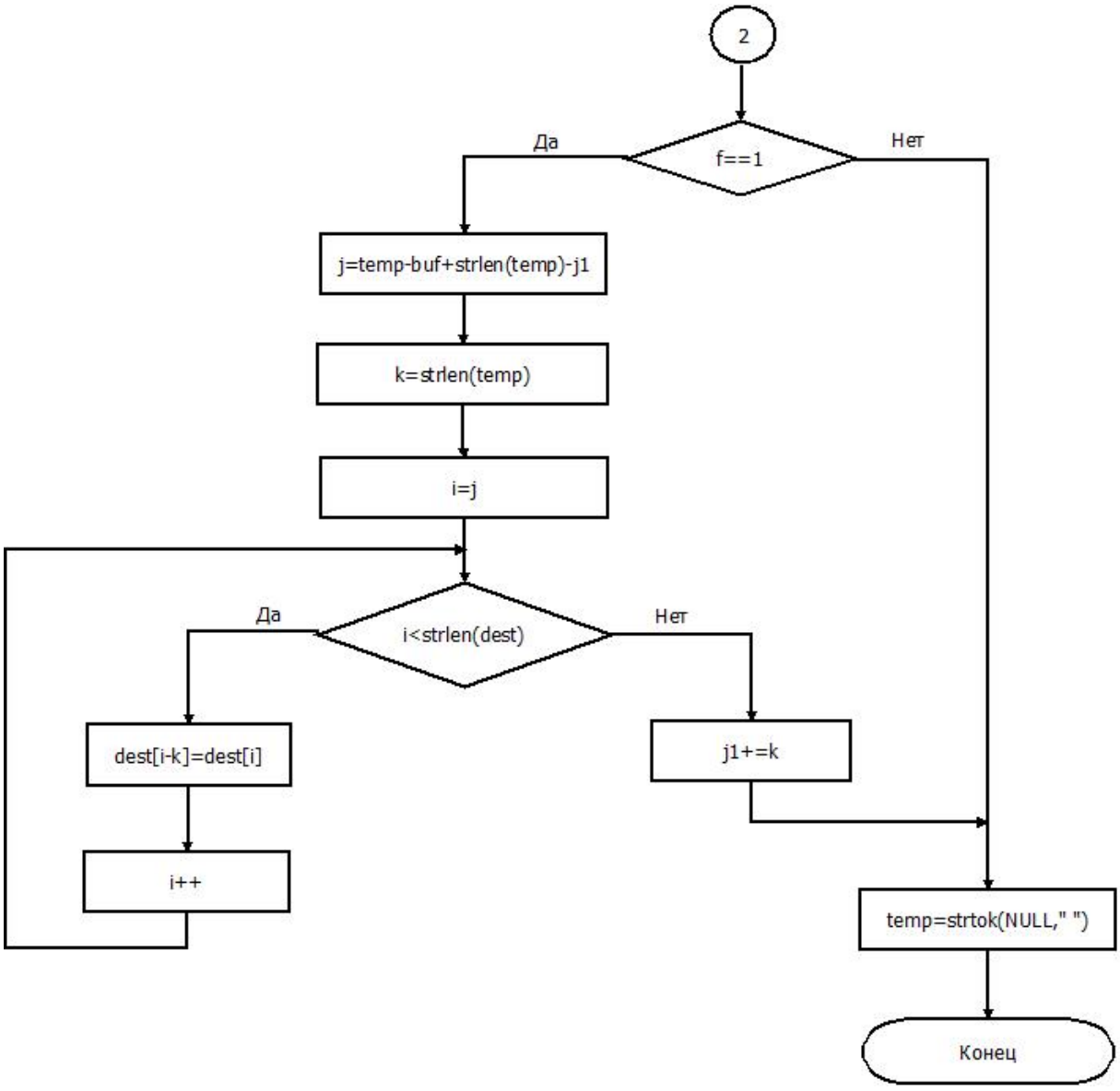


Рис. 5.2 – Алгоритм перемещения символов в начало строки

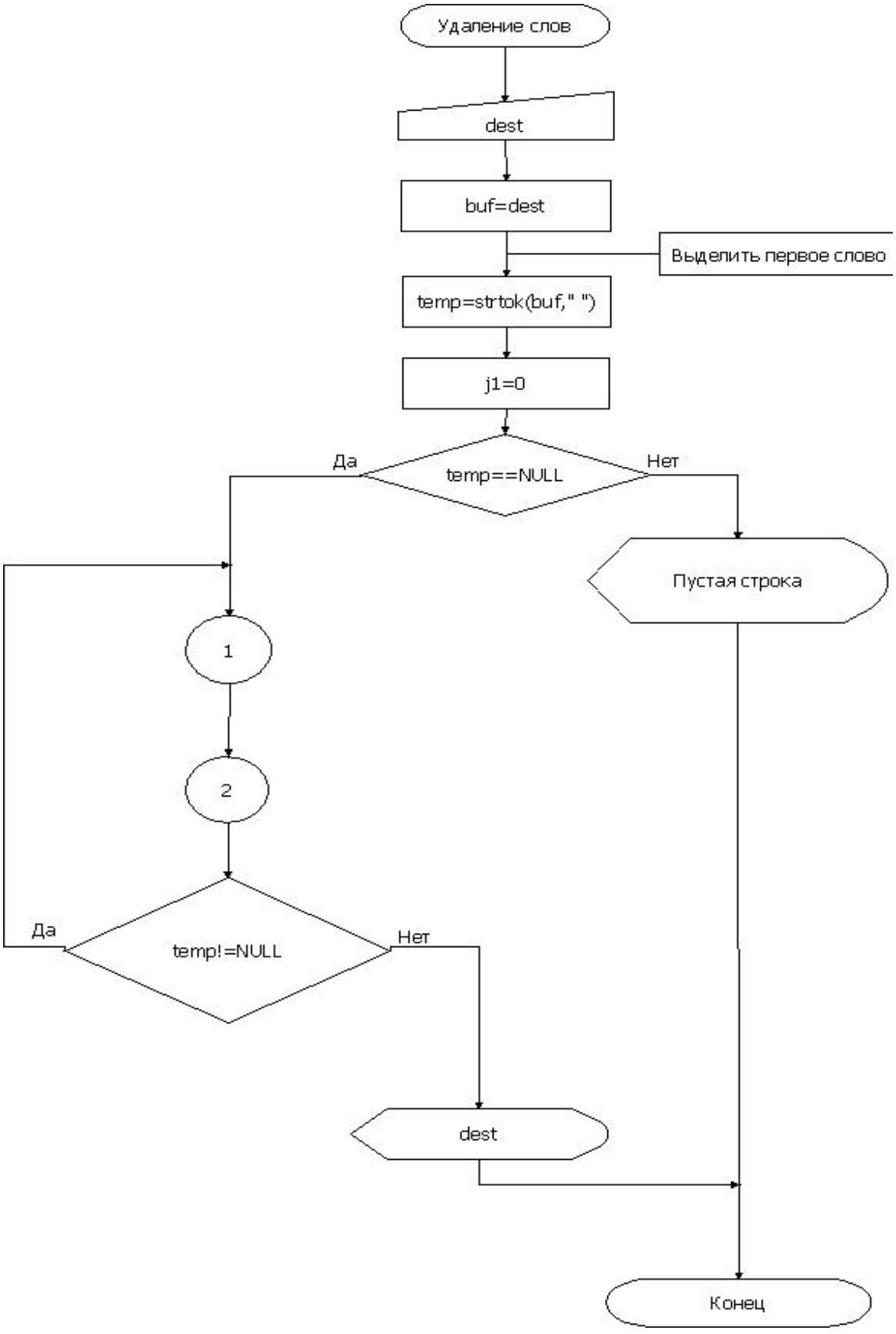


Рис. 5.3 – Алгоритм удаления слов

Ниже представлен код программы на языке Си.

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[]){
 char dest[200];
 printf("Введите произвольную строку: ");
 gets(dest);
 char *buf = (char*)malloc(sizeof(char)*(strlen(dest)+1));
 strcpy(buf,dest);
 int k;
 char *temp = strtok(buf," ");
 int i,j1=0,j,f;
 if (temp==NULL) {printf ("Пустая строка");
                  system("pause");
                  return 0;}
 do {
   f=0;
   for(i=0;i<strlen(temp);i++)
      if(isalpha(temp[i])==0) {f=1;break;}
   if (f==1)
   { j= temp-buf+strlen(temp)-j1;
     k = strlen(temp);
    for(i=j;i<=strlen(dest);i++)
        dest[i-k] = dest[i];
    j1 += k;          }
   temp = strtok(NULL," ");   }
 while (temp!=NULL);
   puts(dest);
 system("PAUSE");
 return 0; }
```

5.3 Порядок выполнения работы

1. Выбрать индивидуальный вариант.
2. Составить и записать алгоритм решения задачи.
3. Составить программу, реализующую записанный алгоритм.
4. Отладить программу.
5. Написать отчет о проделанной работе.

5.4 Содержание отчета

Отчет по лабораторной работе № 4 должен содержать:

- титульный лист,
- текст индивидуального варианта,
- алгоритм решения задачи,
- текст программы,
- результаты тестирования программы.

Пример оформленного отчета приведен в приложении А.

5.5 Индивидуальные варианты заданий лабораторной работы № 4

Вариант 1

Дана строка символов. Преобразовать данную строку, удалив из нее каждую пару символов ‘*ab*’ и повторив (вставив еще раз) каждую пару символов ‘*ba*’. После преобразования полученную строку вывести на печать.

Вариант 2

Даны натуральное число *n* и символы *S*1, …, *Sn* (строка), среди которых есть двоеточия. Получить все символы строки, расположенные между первым и вторым двоеточием. Если второго двоеточия нет, то получить все символы, расположенные после первого двоеточия.

Вариант 3

Дана строка символов. Исключить из этой строки группы символов, расположенные между скобками [ ]. Сами скобки тоже должны быть исключены. Предполагается, что внутри каждой пары скобок нет других скобок.

Вариант 4

Задана строка символов, каждое слово – последовательность, состоящая из одних цифр. Рассматривая каждое слово как число, определить сумму четных и нечетных значений элементов массива.

Вариант 5

Задана строка символов, каждое слово – последовательность, состоящая из одних цифр. Рассматривая каждое слово как число, найдите разность максимального и минимального элемента и среднее значение всех элементов.

Вариант 6

Дана строка символов, состоящая из нулей, единиц и пробелов. Группы нулей и единиц, разделенные пробелами (одним или несколькими) и не содержащие пробелов внутри себя, будем называть словами. Требуется подсчитать количество слов в данной строке. Рассматривая слова как числа, определить количество слов, делящихся на 5 без остатка.

Вариант 7

Дана строка символов, состоящая из нулей, единиц и пробелов. Группы нулей и единиц, разделенные пробелами (одним или несколькими) и не содержащие пробелов внутри себя, будем называть словами. Требуется подсчитать количество слов в данной строке. Вывести каждое слово на отдельной строчке с сообщением, делится ли это число на 10.

Вариант 8

Дана строка символов, состоящая из цифр и пробелов. Требуется подсчитать количество слов в данной строке. Вывести на экран слово с максимальной длиной и напечатать его в перевернутом виде.

Вариант 9

Дана строка символов, напечатать все символы, входящие в строку в алфавитном порядке. Если в исходной строке символ встречается несколько раз, то удалить все его повторные вхождения.

Вариант 10

Разработать программу, которая предназначена для зашифровки текстов. Принять следующий тривиальный алгоритм шифрования: все буквы А в исходном тексте заменяются на букву «У», буквы «П» – на «Ж», буквы «О» – на «Ю» и т. д. (по вашему усмотрению). Вывести на печать исходный текст и результат шифрования. Произвести дешифрование.

Вариант 11

Задана строка символов, каждое слово – последовательность, состоящая из одних цифр и знаков «–». Рассматривая каждое слово как число, определить сумму положительных и отрицательных значений элементов массива.

Вариант 12

Дана строка символов. Группы символов, разделенные пробелами (одним или несколькими) и не содержащие пробелов внутри себя, будем называть словами. Найти количество слов строки. Вывести все слова строки, длина которых превышает количество слов.

Вариант 13

Дана строка символов. Группы символов, разделенные пробелами (одним или несколькими) и не содержащие пробелов внутри себя, будем называть словами. Удалить из каждого слова символы, не являющиеся буквами. Напечатать каждое слово строки на отдельной строчке.

Вариант 14

Дана строка символов, состоящая из нулей, единиц и пробелов. Группы нулей и единиц, разделенные пробелами (одним или несколькими) и не содержащие пробелов внутри себя, будем называть словами.

Найти количество слов с заданной длиной. Вывести найденные слова «в столбик»

Вариант 15

Дана строка символов. Группы символов, разделенные пробелами (одним или несколькими) и не содержащие пробелов внутри себя, будем называть словами. Найти длину самого короткого слова и вывести это слово на экран.

Вариант 16

Дана строка символов. Группы символов, разделенные пробелами (одним или несколькими) и не содержащие пробелов внутри себя, будем называть словами. Подсчитать количество слов, у которых первый и последний символы не совпадают.

Вариант 17

Дана строка символов. Группы символов, разделенные пробелами (одним или несколькими) и не содержащие пробелов внутри себя, будем называть словами. Подсчитать количество слов, не содержащих заданный символ.

Вариант 18

Дана строка символов. Группы символов, разделенные пробелами (одним или несколькими) и не содержащие пробелов внутри себя, будем называть словами. Подсчитать количество заданного символа в первом и последнем слове данной строки.

Вариант 19

Дана строка символов. Группы символов, разделенные пробелами (одним или несколькими) и не содержащие пробелов внутри себя, будем называть словами. Подсчитать количество слов, длина которых не превышает заданного числа *n*.

Вариант 20

Задана строка символов, каждое слово – последовательность, состоящая из одних цифр. Рассматривая каждое слово как число, найти такие числа, цифры которых составляют неубывающую последовательность.

СПИСОК ЛИТЕРАТУРЫ

1. Пермякова Н. В. Информатика и программирование : учеб. пособие / Н. В. Пермякова. – Томск : ФДО ТУСУР, 2016. – 188 с.

2. Павловская Т. А. C/C++. Программирование на языке высокого уровня : учебник для вузов. – СПб. : Питер, 2005. – 460 с.

3. Павловская Т. А. C/C++. Структурное программирование. Практикум: учеб. пособие для вузов. – СПб. : Питер, 2005. – 238 с.

4. Костюк Ю. Л. Основы алгоритмизации : учеб. пособие – Томск : [б. и.], 1999. – 122 с.

ПРИЛОЖЕНИЕ

Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Факультет систем управления

Кафедра АОИ

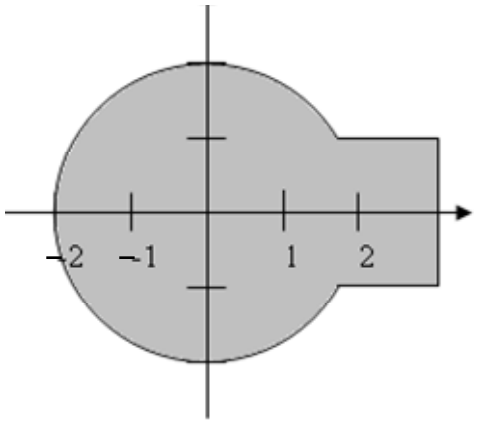
Лабораторная работа № 1
по дисциплине «Информатика и программирование»

Выполнил:
студент ФДО ТУСУР
специальность 231000.62
ФИО

Томск 2016

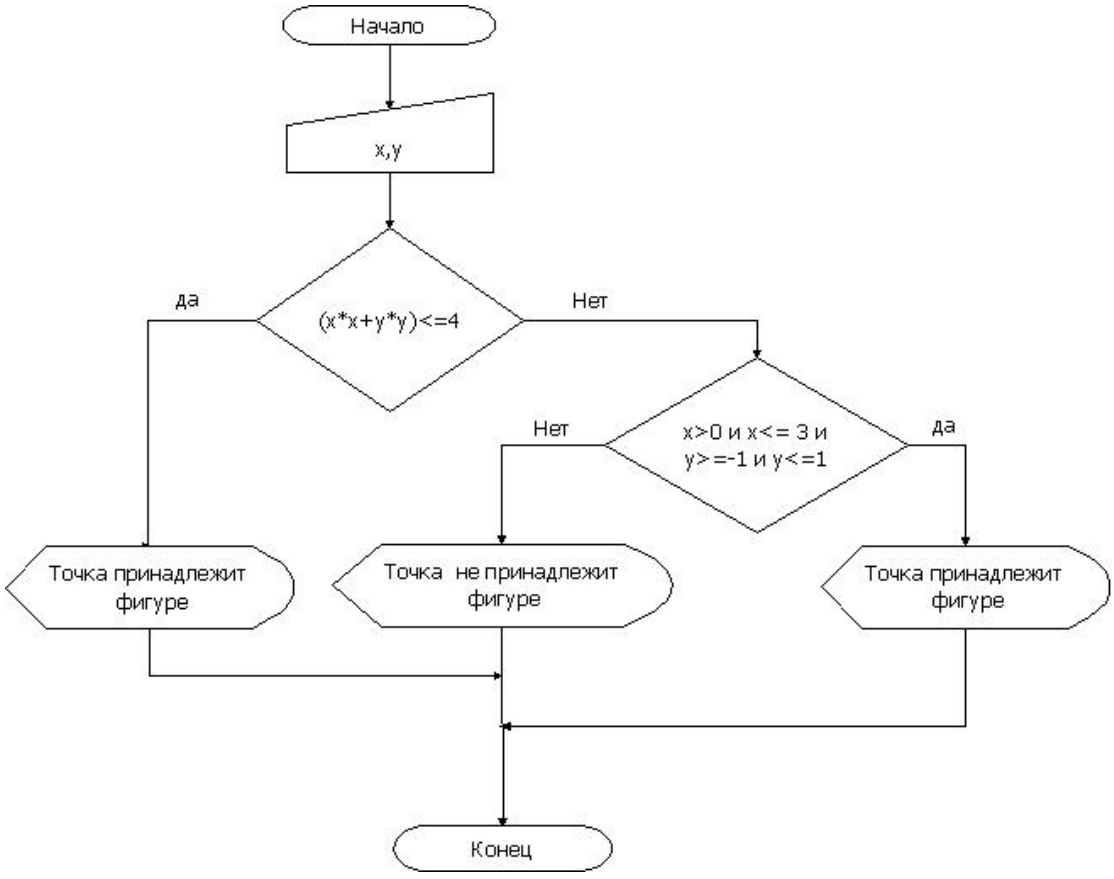
Индивидуальный вариант

Проверить, принадлежит ли точка с заданными координатами (*x*, *y*) заштрихованной области:



Алгоритм решения задачи

Алгоритм решения задачи представлен в виде блок-диаграммы.



Текст программы

```
int main(int argc, char *argv[])
{
   system("chcp 1251");
   printf("Введите координаты x и y: ");
   float x,y;
   scanf("%f%f",&x,&y);
   if((x*x+y*y)<=4) printf("Точка принадлежит фигуре \n");
     else  if (x>0&&x<=3&&y>=-1&&y<=1)
                        printf("Точка принадлежит фигуре \n");
           else        printf("Точка не принадлежит фигуре \n");
   system("PAUSE");
   return EXIT_SUCCESS;
}
```

Результаты тестирования программы

На рисунке 1 представлены экранные формы работы программы

- при введенных значениях координат (0,0) точка с такими координатами принадлежит окружности, входящей в фигуру;
- при введенных значениях координат (2.5,0.7) точка с такими координатами принадлежит прямоугольнику, входящему в фигуру;
- при введенных значениях координат (4,1) точка с такими координатами не принадлежит фигуре.

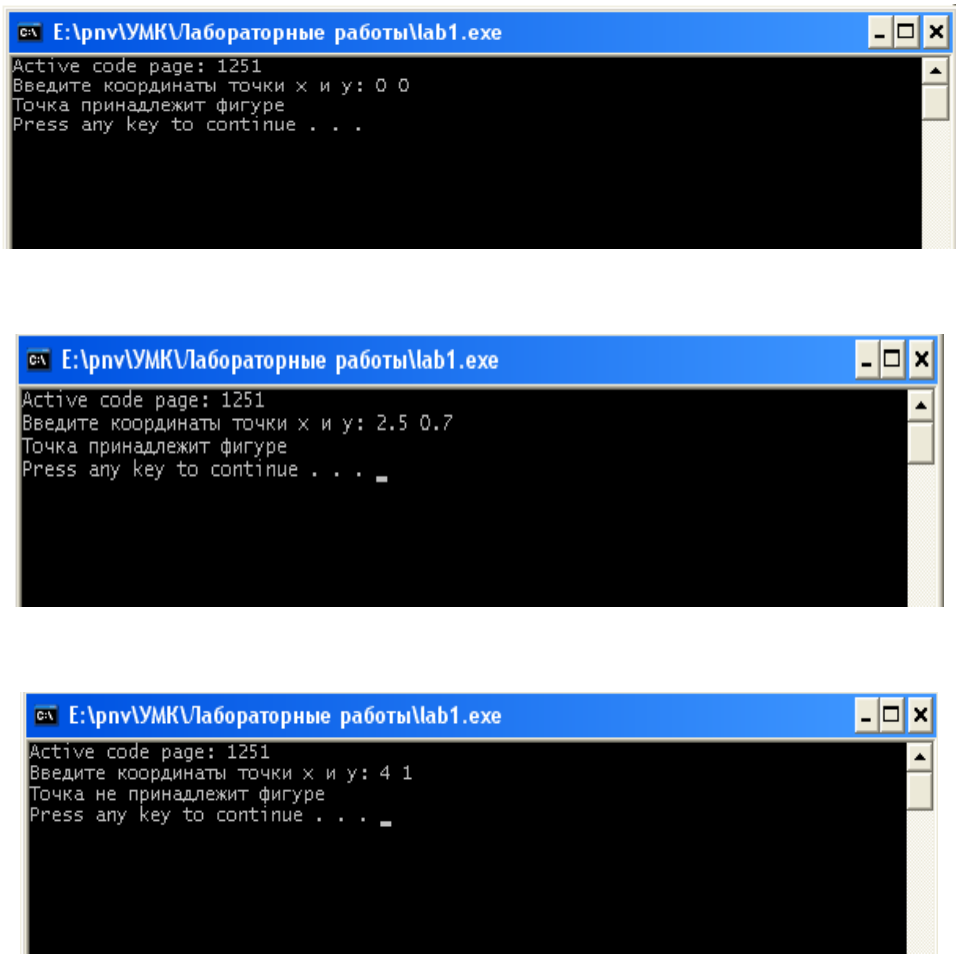


Рис. 1 – Результаты тестирования