МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
ДОНЕЦКОЙ НАРОДНОЙ РЕСПУБЛИКИ
ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ

для выполнения индивидуального задания
по дисциплине «Информатика»

Донецк
2022

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
ДОНЕЦКОЙ НАРОДНОЙ РЕСПУБЛИКИ
ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

КАФЕДРА «ПРИКЛАДНАЯ МАТЕМАТИКА
И ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ»

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ

для выполнения индивидуального задания
по дисциплине «Информатика»
для обучающихся по направлению подготовки
13.03.02 «Электроэнергетика и электротехника»
заочной формы обучения

РАССМОТРЕНО
на заседании кафедры
прикладной математики и
искусственного интеллекта
Протокол № 4 от 07.04.2022 г.

УТВЕРЖДЕНО
на заседании учебно-издательского
совета ДОННТУ
Протокол № 3 от 20.04.2022 г.

Донецк
2022

УДК 004:620.9:621.3(076)
М54

Составитель:
Ефименко Константин Николаевич – кандидат технических наук, доцент кафедры прикладной математики и искусственного интеллекта ГОУВПО «ДОННТУ»

М54 **Методические рекомендации для выполнения индивидуального задания по дисциплине «Информатика»** : для обучающихся по направлению подготовки 13.03.02 «Электроэнергетика и электротехника» заочной формы обучения / ГОУВПО «ДОННТУ», Каф. прикладной математики и искусственного интеллекта ; сост. К. Н. Ефименко. – Донецк : ДОННТУ, 2022. – Систем. требования: Acrobat Reader. – Загл. с титул. экрана.

В методических рекомендациях рассмотрена структура индивидуального задания (контрольной работы), его цель, задачи, содержание, объем и последовательность выполнения. Приведены примеры выполнения каждого задания контрольной работы. Прилагается список рекомендуемой литературы.

УДК 004:620.9:621.3(076)

СОДЕРЖАНИЕ

ЦЕЛЬ И ЗАДАЧИ ОСВОЕНИЯ ДИСЦИПЛИНЫ

Цель дисциплины «Информатика» – формирование у обучающихся навыков алгоритмического мышления, умения выполнять постановку задачи для разработки программного обеспечения и реализации алгоритмов в виде компьютерных программ.

Задачи дисциплины – изучение принципов организации вычислительных процессов, понятия алгоритмизации, основных типов алгоритмов, способов их представления; освоение этапов разработки программ на языке программирования С++.

В результате изучения дисциплины «Информатика» обучающиеся должны получить следующие навыки:

– владеть навыками работы с основными компонентами офисных пакетов;

– представлять алгоритм решения поставленной задачи в виде блок-схемы; использовать типовые структуры алгоритмов;

– реализовывать блок-схемы алгоритма в виде программы на языке программирования С++ с использованием компиляторов Dev-C++ или Microsoft Visual Studio;

– применять стандартные компьютерные технологии обработки и анализа технической информации;

– применять Internet при решении профессиональных задач.

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ К ВЫПОЛНЕНИЮ ИНДИВИДУАЛЬНОГО ЗАДАНИЯ

Индивидуальное задание по дисциплине учебным планом предусмотрено в 1-м семестре для заочной формы обучения, и реализуется в виде контрольной работы, которая выполняется в течение семестра.

Тематика индивидуального задания (контрольной работы) связана с самостоятельным выполнением работ по темам дисциплины, которые не рассматриваются на лекциях и лабораторных занятиях и изучаются обучающимся самостоятельно в соответствии с [1]. Объем учебной нагрузки при выполнении индивидуального задания – 9 часов.

Контрольная работа состоит из пяти заданий. При выполнении каждого задания необходимо составить блок-схему алгоритма решения задачи и программу на языке программирования С++ в соответствии с выбранным вариантом.

Отчет о выполнении контрольной работы (пояснительная записка) может быть выполнен письменно в отдельной тетради или распечатан на листах формата А4.

Пояснительная записка должна включать:

– титульный лист (рис. 1);

– лист с заданием на контрольную работу, содержащий таблицу с выбранными вариантами для 5 заданий и текст самих заданий (рис. 2);

– результаты выполнения каждого задания.

ГОУВПО «Донецкий национальный технический университет»
Кафедра прикладной математики и искусственного интеллекта

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к индивидуальному заданию по дисциплине «Информатика»

Выполнил: обучающийся гр. *<группа>*,
факультета *<название факультета>*
<Ф.И.О.>
зачетная книжка *<№ зачетки>*

Преподаватель:
<Ф.И.О. преподавателя>

Донецк
2022

Рисунок 1 – Образец оформления титульного листа

ЗАДАНИЕ НА КОНТРОЛЬНУЮ РАБОТУ

Составить блок-схему алгоритма решения задачи и программу на языке С++ в соответствии с выбранным вариантом задания.

ФИО					
Задание	1	2	3	4	5
Вариант					

ЗАДАНИЕ №1. Организация линейного и разветвляющегося вычислительных процессов

ЗАДАНИЕ №2. Организация циклов с известным числом повторений

ЗАДАНИЕ №3. Организация вложенных циклов

ЗАДАНИЕ №4. Обработка одномерных массивов

ЗАДАНИЕ №5. Обработка двумерных массивов

Задание	Вариант	Необходимо вычислить	Исходные данные	Выводимые значения
1				
2				
3				
4				
5				

Рисунок 2 – Лист с заданием на контрольную работу

Результаты выполнения каждого задания оформляются в следующей последовательности:

1. Исходные данные.
2. Постановка задачи (математическая модель).
3. Ограничения на решение задачи.
4. Выходные данные.
5. Блок-схема алгоритма.
6. Текст программы решения задачи на С++.
7. Screenshot консольного окна с результатами работы программы.

Блок-схема алгоритма и программа на языке С++ должны располагаться на отдельных страницах.

Рекомендуемый объем пояснительной записки по индивидуальному заданию – не более 15 страниц формата А4 (210×297 мм).

Выбор вариантов заданий для выполнения контрольной работы

Номер варианта в каждом задании выбирается по буквам фамилии студента в соответствии с таблицей 1.

Таблица 1. Кодировка вариантов заданий

№ варианта	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Буква	А	Б	В	Г	Д	Е	Ё	Ж	З	И	Й	К
	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц
	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я			

Например, для студента с фамилией ИВАНОВ варианты заданий будут следующими (рис. 3):

Фамилия	**И**	**В**	**А**	**Н**	**О**
№ задания	1	2	3	4	5
№ варианта	**10**	**3**	**1**	**3**	**4**

Рисунок 3 – Пример выбора вариантов заданий

Таким образом, при выполнении контрольной работы студент с указанной фамилией, должен решить следующие варианты заданий:

Задание 1 – вариант 10.
Задание 2 – вариант 3.
Задание 3 – вариант 1.
Задание 4 – вариант 3.
Задание 5 – вариант 4.

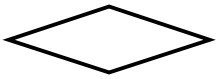
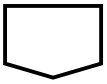
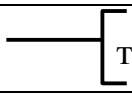
Если фамилия содержит меньше 5 букв, то в качестве недостающих взять первые буквы имени студента.

ЗАДАНИЕ №1. ОРГАНИЗАЦИЯ ЛИНЕЙНОГО И РАЗВЕТВЛЯЮЩЕГОСЯ ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ

1. Основные теоретические положения

Алгоритм – это строгая последовательность арифметических и логических действий, которая однозначно определяет процесс вычисления результата в зависимости от исходных данных [2,3,4]. Наиболее удобным и наглядным способом представления алгоритма является графический в виде **блок-схемы**. При этом каждый логически завершенный этап вычислительного процесса изображается в виде специального геометрического символа – **блока**. Наиболее часто используемые графические символы представлены в таблице 2.

Таблица 2. Графические символы, применяемые при составлении блок-схем

№ п/п	Наименование блока	Обозначение	Выполняемое действие
1	Начало или конец алгоритма		Показывает начало или конец алгоритма
2	Ввод или вывод		Обеспечивает ввод или вывод данных в алгоритме
3	Арифметический		Выполняет арифметические вычисления
4	Логический		Выполняет проверку заданного логического условия
5	Модификации		Заголовок цикла «Для»
6	Предопределенный процесс		Вызов подпрограммы
7	Линии потока		Указывают связь и направление движения между блоками
8	Соединитель		Указывает связь между прерванными линиями потока
9	Межстраничный соединитель		Указывает связь между частями блок-схемы, которые расположены на разных страницах
10	Комментарии		Запись пояснения к блоку или к линии потока

При составлении блок-схемы алгоритма блоки записываются последовательно друг за другом и соединяются линиями потока информации, которые показывают направление движения по блок-схеме. В блок-схеме любой путь движения из блока «Начало» алгоритма должен привести в блок «Конец» алгоритма.

В общем случае любой алгоритм может состоять из трех частей: ввод исходных данных, вычисление требуемых величин и вывод полученных результатов. Существует три основных типовых структуры алгоритма:

1. Линейный вычислительный процесс.
2. Разветвляющийся вычислительный процесс.

3. Циклический вычислительный процесс.

Любой алгоритм сложной структуры может быть получен путем комбинированного использования типовых структур.

В **линейном вычислительном процессе** все действия выполняются в строгой последовательности друг за другом. Таким образом, существует только один путь, по которому можно пройти из блока «Начало» в блок «Конец» алгоритма, т.е. выполнить алгоритм.

Разветвляющийся вычислительный процесс позволяет выбрать один из нескольких вариантов решения поставленной задачи в зависимости от выполнения некоторых условий. Таким образом, существует несколько различных путей, по которым можно пройти из блока «Начало» в блок «Конец» алгоритма, т.е. выполнить алгоритм.

Для реализации процесса выбора одного из двух вариантов решения используется логический блок (блок проверки условий) (рис. 4). При входе в блок выполняется проверка логического условия (обычно математического неравенства). Если результат проверки условия «Истина», т.е. условие выполняется, то происходит переход к выполнению блоков, стоящих по ветви «+». В противном случае, т.е. когда проверяемое условие не выполняется, происходит переход к выполнению блоков, стоящих по ветви «–».

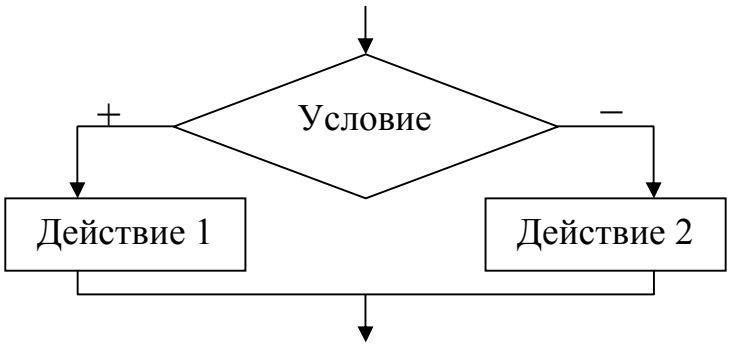


Рисунок 4 – Блок проверки условия

При выполнении вычислений необходимо учитывать область определения математических функций. Вначале необходимо проверить возможность вычисления данного математического выражения при текущих значениях исходных данных, т.е. проверить ограничения. К наиболее часто встречающимся ограничениям относятся: операция деления (на 0 делить нельзя), вычисление квадратного корня (подкоренное выражение должно быть ≥ 0), вычисление логарифма (выражение под знаком логарифма должно быть > 0), вычисление tg, ctg. В случае не выполнения ограничения (невозможно выполнить вычисления) необходимо пропустить все действия, которые зависят от вычисляемой величины, и перейти в ту часть алгоритма, где можно продолжить вычисления.

Задание №1. **Составить блок-схему алгоритма и программу на C++**, которая в соответствии с исходными данными вычисляет значения заданных выражений. Программу реализовать в виде консольного приложения (**Console Application**). Для ввода-вывода данных использовать функции ввода-вывода (**scanf** и **printf**). Варианты задания приведены в таблице 3.

2. Пример выполнения задания №1

Постановка задачи:

1. Исходные данные: *a, b*
2. Математическая модель:

$$y=\begin{cases}\sqrt{a^2+1}+\sin\dfrac{\pi}{2}x, & \text{если } x<1.5\\ |a+x|, & \text{если } 1.5\le x\le 3.5\\ \sqrt{x-a}, & \text{если } x>3.5\end{cases}\qquad x=\begin{cases}\ln ab-1, & \text{если } ab>1\\ \dfrac{b-1}{a}, & \text{если } ab\le 1\end{cases}$$

3. Ограничения:

а) подкоренное выражение $a^2+1 \ge 0$, **не проверять**, т.к. a^2+1 всегда больше 0;

б) подкоренное выражение $x - a \ge 0$;

в) выражение под знаком логарифма $ab > 0$, **не проверять**, т.к. это выражение для вычисления x используется только если $ab > 1$;

г) знаменатель $a \ne 0$.

4. Выходные данные: x, y

5. Блок-схема алгоритма (рис. 5):

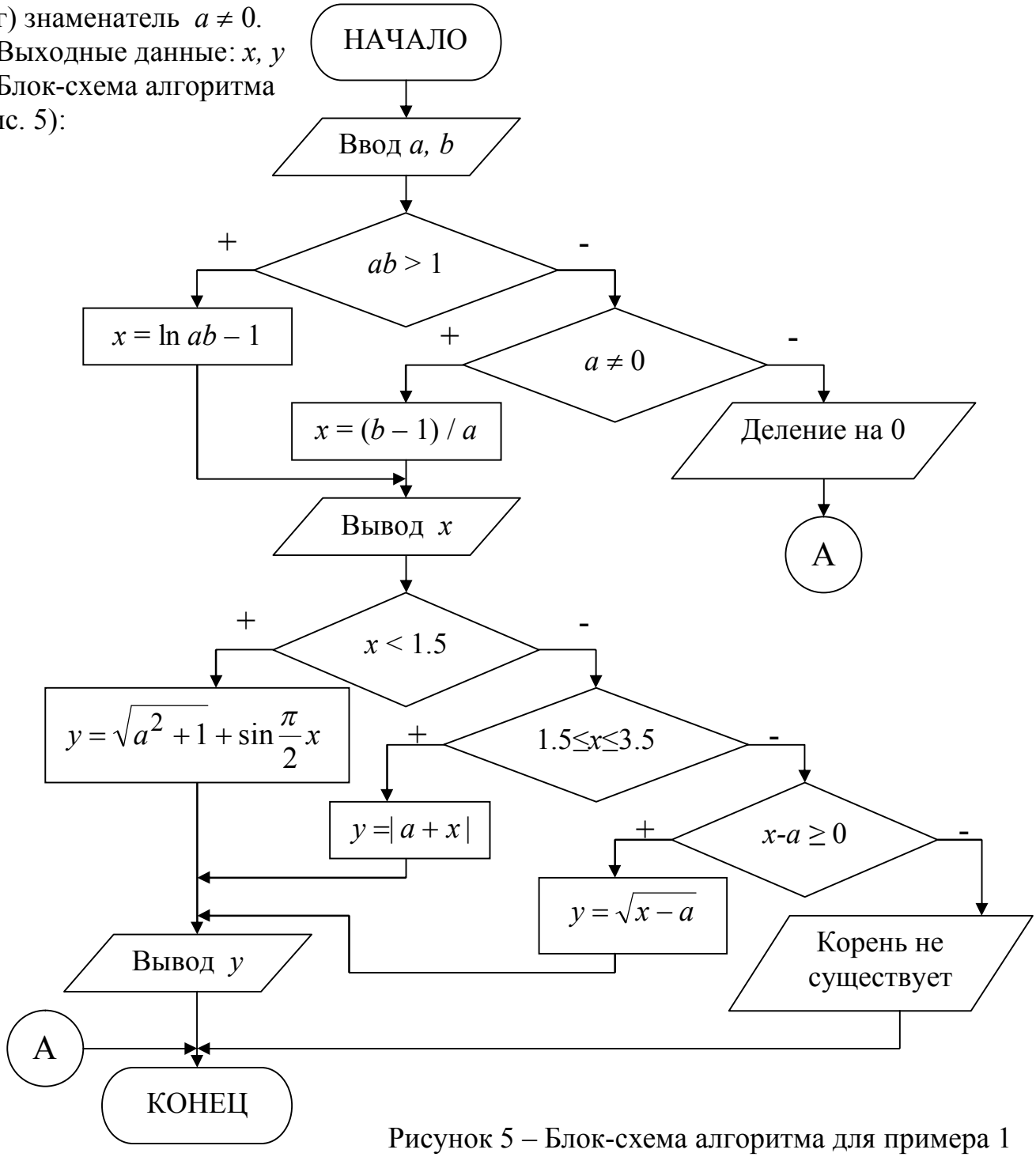


Рисунок 5 – Блок-схема алгоритма для примера 1

6. Программа решения задачи на C++, реализованная с помощью компилятора Dev-C++. Результаты работы программы показаны на рисунке 6.

```
//Директивы процессора
#include <cstdlib>
#include <iostream>
#include <math.h>
#include <stdio.h>
using namespace std;
//Начало главной функции программы
int main(int argc, char *argv[])
{ //Описание константы π и переменных
  const float Pi=3.14159;
  float a,b,x,y;
  //Ввод исходных данных a и b
  printf("Input a=");
  scanf("%f",&a);
  printf("Input b=");
  scanf("%f",&b);
  //Вычисление значения x
  if (a*b>1) x=log(a*b)-1;
  else
    if (a!=0) x=(b-1)/a;
    else
    { //Вывод сообщения о невыполнении ограничения г)
      printf("Error!\n"); goto m1;  }
  //Вывод результата x
  printf("x=%6.2f \n",x);
  //Вычисление значения y
  if (x<1.5) y=sqrt(a*a+1)+sin(Pi/2*x);
  else
    if ((x>=1.5)&&(x<=3.5)) y=fabs(a+x);
    else
      if (x-a>=0) x=sqrt(x-a);
      else
      { //Вывод сообщения о невыполнении ограничения б)
        printf("Error!\n"); goto m1;  }
  //Вывод результата y
  printf("y=%6.2f \n",y);
  m1:
  system("PAUSE");
  return EXIT_SUCCESS;
}
```

```
Input a=2
Input b=1
x= -0.31
y=  1.77
Для продолжения нажмите любую клавишу . . .
```

Рисунок 6 – Результаты работы программы

Таблица 3. Варианты для задания №1

№ вар.	Модель	№ вар.	Модель
1	$y=\begin{cases}x-ab, & \text{если } x<4\\ \ln x+ab, & \text{если } 4\le x\le 5\\ (x+a)/b, & \text{если } x>5\end{cases}$ $x=\begin{cases}a/\mathrm{b}, & \text{если } a<b\\ a-\mathrm{b}, & \text{если } a\ge b\end{cases}$	2	$y=\begin{cases}\ln a+cx, & \text{если } x<1\\ b+d/x, & \text{если } 1\le x<3\\ c-ax, & \text{если } x\ge 3\end{cases}$ $x=\begin{cases}\sqrt{ab}, & \text{если } ab\ge cd\\ \sqrt[3]{cd}, & \text{если } ab<cd\end{cases}$
3	$z=\begin{cases}\ln ax, & \text{если } \lvert x\rvert<3\\ bx^3, & \text{если } \lvert x\rvert=3\\ cx-1, & \text{если } \lvert x\rvert>3\end{cases}$ $x=\begin{cases}a+bc, & \text{если } ab\ge c\\ ab/c, & \text{если } ab<c\end{cases}$	4	$y=\begin{cases}\sqrt[3]{a+x}, & \text{если } x<1\\ \ln bx, & \text{если } 1\le x\le 5\\ \sqrt{a+bx}, & \text{если } x>5\end{cases}$ $x=\begin{cases}a^2b, & \text{если } a<b\\ ab^2, & \text{если } a\ge b\end{cases}$
5	$z=\begin{cases}\sin^2 x+1, & \text{если } x\le c\\ \cos x-1, & \text{если } c<x<d\\ e^x+1/a, & \text{если } x\ge d\end{cases}$ $x=\begin{cases}(a+c)d, & \text{если } a<c\\ (a-c)/d, & \text{если } a\ge c\end{cases}$	6	$y=\begin{cases}a\sqrt[3]{x}, & \text{если } x<1\\ tg(bx), & \text{если } 1\le x\le 3\\ cx^2, & \text{если } x>3\end{cases}$ $x=\begin{cases}ab+c, & \text{если } a\le b+1\\ a/b-c, & \text{если } a>b+1\end{cases}$
7	$y=\begin{cases}x^3+1, & \text{если } x<4\\ x^2+1, & \text{если } 4\le x<5\\ \ln(x+1), & \text{если } x\ge 5\end{cases}$ $x=\begin{cases}a^2/b^2, & \text{если } a\le b\\ a/b, & \text{если } a>b\end{cases}$	8	$y=\begin{cases}(a+b)x, & \text{если } x<3\\ (a-b)x, & \text{если } x=3\\ ax/b, & \text{если } x>3\end{cases}$ $x=\begin{cases}a^2+1, & \text{если } ab\ge 1\\ \sqrt{b^2-1}, & \text{если } ab<1\end{cases}$
9	$y=\begin{cases}\ln a+x, & \text{если } x<5\\ ax, & \text{если } 5\le x<7\\ x/a, & \text{если } x\ge 7\end{cases}$ $x=\begin{cases}\sqrt{a^2+1}, & \text{если } a\ge 2\\ \sqrt[3]{a^3+1}, & \text{если } a<2\end{cases}$	10	$y=\begin{cases}\sqrt{b+x^2}, & \text{если } x<1\\ abx, & \text{если } 1\le x\le 5\\ bx^3, & \text{если } x>5\end{cases}$ $x=\begin{cases}\cos ab+3, & \text{если } ab\le 3\\ a/b-3, & \text{если } ab>3\end{cases}$
11	$y=\begin{cases}a\sqrt{x}, & \text{если } x<2\\ bx^2, & \text{если } 2\le x<3\\ c\cdot e^x, & \text{если } x\ge 3\end{cases}$ $x=\begin{cases}(a+b)/c, & \text{если } a\le b\\ (a-b)c, & \text{если } a>b\end{cases}$	12	$y=\begin{cases}b\cdot e^x, & \text{если } x=2\\ \sin x, & \text{если } x>2\\ ax^2+\sqrt{b}, & \text{если } x<2\end{cases}$ $x=\begin{cases}ab+2, & \text{если } a\ge b\\ a/b+2, & \text{если } a<b\end{cases}$

ЗАДАНИЕ №2. ОРГАНИЗАЦИЯ ЦИКЛОВ С ИЗВЕСТНЫМ ЧИСЛОМ ПОВТОРЕНИЙ

1. Основные теоретические положения

В **алгоритмах циклической структуры** выполнение одних и тех же действий может повторяться несколько раз [2,3,4]. Этапы организации циклического вычислительного процесса:

I – **подготовка к выполнению цикла**: присваивание начальных значений параметру цикла и переменным, использующихся для хранения накапливающихся величин (сумма, количество или произведение вычисляемых величин).

II – **тело цикла**: арифметические и логические действия, которые могут повторяться определенное количество раз. В конце тела цикла обязательно должен быть блок, в котором изменяется значение параметра цикла.

III – **условие входа в цикл**: проверяется надо ли повторять вычисления, или выходить из цикла.

Параметр цикла – это переменная, на основе которой строится цикл. Она должна удовлетворять трем условиям: являться исходной величиной для выполнения вычислений; изменяться по определенному закону (чаще всего это закон арифметической прогрессии); оказывать влияние на условие завершения повторяющихся вычислений.

Существует три основных типа циклов: **цикл с постусловием** (рис. 7), **цикл с предусловием** (рис. 8) и **цикл «Для»** на основе блока модификации.

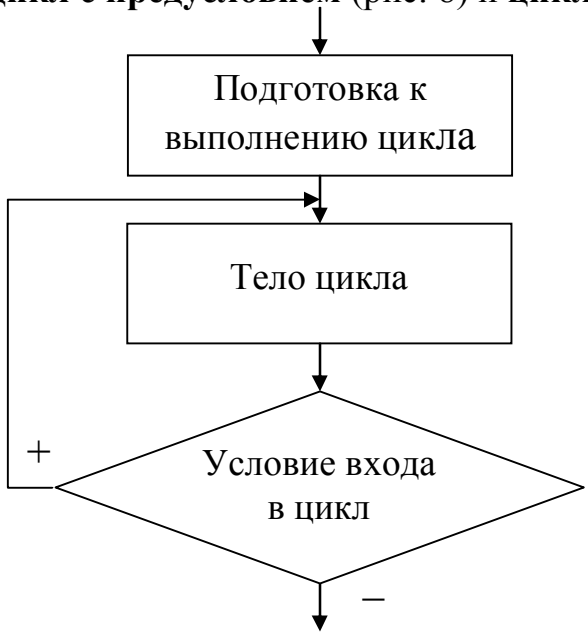


Рисунок 7 – Организация цикла с постусловием

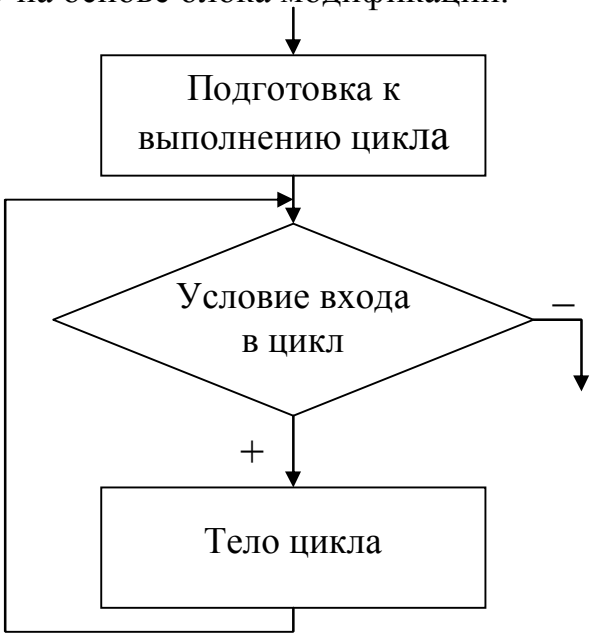


Рисунок 8 – Организация цикла с предусловием

В цикле с постусловием после каждого выполнения тела цикла, проверяется условие входа в цикл. Если оно выполняется, то происходит возврат на начало тела цикла и повторение вычислений. Когда условие входа не будет выполнено, произойдет завершение работы и выход из цикла. Наличие линии возврата в блок-схеме является основным признаком циклического вычислительного процесса.

Аналогично работает цикл с предусловием, в котором условие входа в цикл

проверяется в начале тела цикла. Таким образом, в цикле с постусловием в отличие от цикла с предусловием, тело цикла всегда выполнится хотя бы один раз.

Однократное выполнение тела цикла называется **шагом**. Циклические вычислительные процессы, для которых можно вычислить количество шагов цикла без выполнения алгоритма, называются **циклами с известным числом повторений**. Количество шагов выполнения цикла N можно вычислить до начала работы алгоритма по следующей формуле:

$$N = \left]\frac{xk - xn}{hx}\right[+ 1,$$

где] [обозначают целую часть выражения.

Задание исходных данных при решении поставленной задачи в виде *xn, xk,* Δx (начальное, конечное значения x из заданного интервала и шаг изменения x), обеспечивает соответствие блок-схемы двум свойствам алгоритма – определенность и массовость. Такой способ называется составлением алгоритма в «**общем виде**» (значения всех исходных данных вводятся, а не присваиваются в процессе выполнения алгоритма).

Для реализации циклов с известным числом повторений можно равноценно использовать любой из трех стандартных типов цикла (с постусловием, с предусловием, «Для»).

Если в теле цикла при выполнении вычислений не выполняется ограничение, то завершать выполнение алгоритма, как в разветвляющемся вычислительном процессе, не надо. В этом случае необходимо пропустить все операции, использующие переменную, значение которой невозможно вычислить, и выполнить переход к блоку, в котором изменяется значение параметра цикла, т.е. продолжить работу цикла. При следующем значении параметра цикла ограничение может выполниться, и работа цикла пойдет естественным путем.

При нахождении суммы, произведения или количества значений, полученных при каждом выполнении тела цикла, используется принцип пошагового накапливания величин. Для хранения таких «накапливающихся» величин используются дополнительные переменные, которым предварительно необходимо присвоить начальные значения. Обычно это делается на этапе подготовки к выполнению цикла. В качестве начального значения для суммы (S) и количества (k) используется ноль, для произведения (P) – единица.

```
S=0; k=0; P=1; //Присваивание начальных значений
…
//Накапливание значений в теле цикла
S=S+Имя_переменной;   //S+=Имя_переменной;
k=k+1;                //k++;
P=P*Имя_переменной;   //P*=Имя_переменной;
```

Задание №2. Используя ***цикл с постусловием***, составить блок-схему алгоритма и программу на языке С++, которая для каждого значения $\boldsymbol{x}$ из заданного интервала $\boldsymbol{xn \le x \le xk}$ с шагом $\boldsymbol{\Delta x}$ вычисляет соответствующие значения ***y=f(x)***. Программу реализовать в виде консольного приложения (**Console Application**).

Для ввода-вывода данных использовать функции ввода-вывода (**scanf** и **printf**). Варианты задания приведены в таблице 4.

2. Пример выполнения задания №2

Постановка задачи:

1. Исходные данные: *a, xn, xk,* Δx
2. Математическая модель:

$y = \ln(1/x) + \cos^2(x-a)$ Вычислить количество y≤*a*

$S = \sum_{y>a} y \qquad P = \prod_{y<a} y$

3. Ограничения: выражение под знаком логарифма $x > 0$
4. Выходные данные: *x, y, S, P* и *k*.
5. Блок-схема алгоритма (рис. 9):

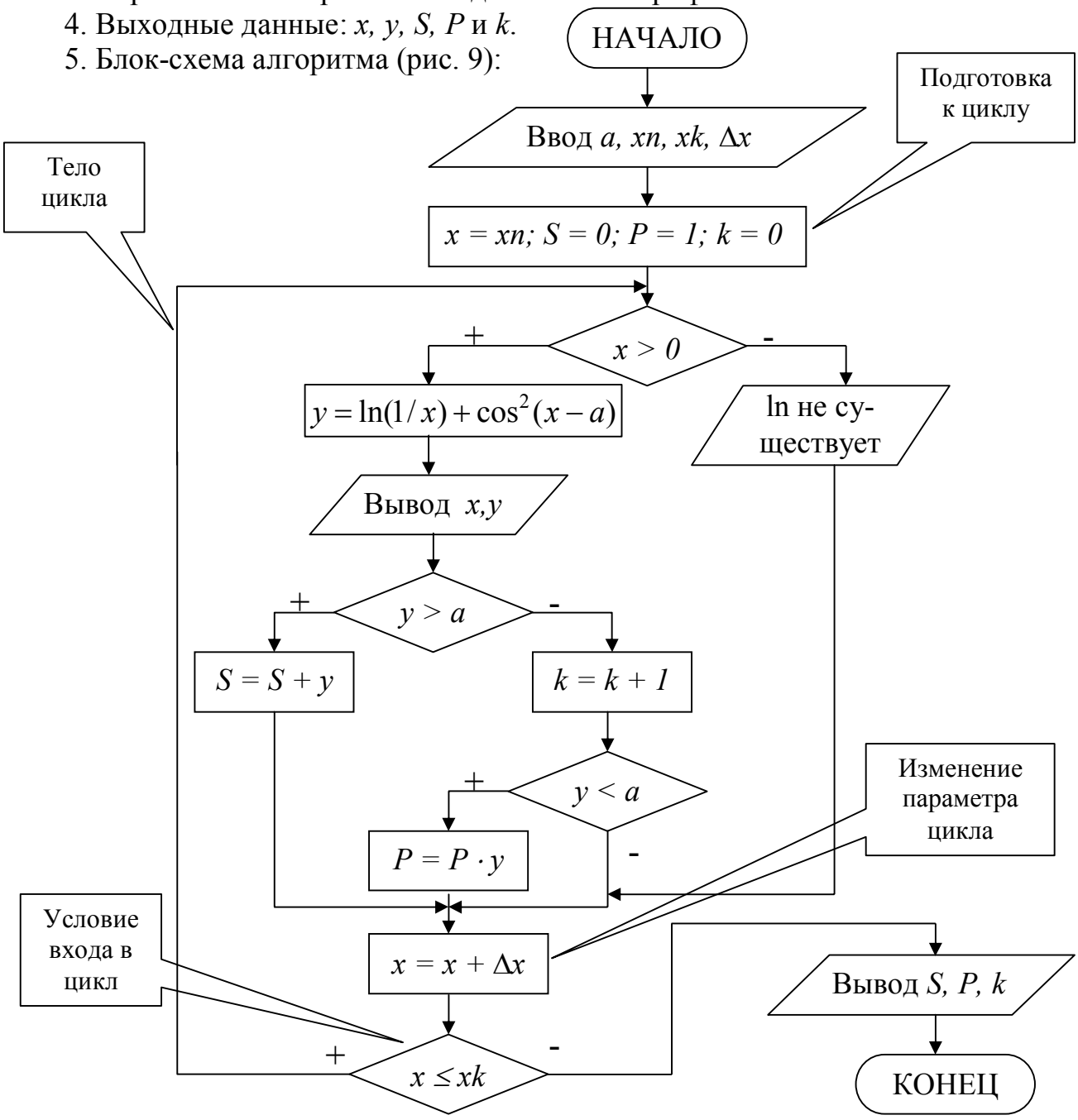


Рисунок 9 – Блок-схема алгоритма для примера 2

6. Программа решения задачи на C++. Результаты работы программы показаны на рисунке 10.

```
#include <cstdlib>
#include <iostream>
#include <math.h>
#include <stdio.h>
using namespace std;
int main(int argc, char *argv[])
{ float xn,xk,dx;
  float a,x,y,S,P;
  int k;
  //Ввод исходных данных
  printf("Input a=");   scanf("%f",&a);
  printf("Input xn=");  scanf("%f",&xn);
  printf("Input xk=");  scanf("%f",&xk);
  printf("Input dx=");  scanf("%f",&dx);
  x=xn; S=0; P=1; k=0;
  //Начало тела цикла с постусловием
  do
  { if (x>0)
    {
      y=log(1/x)+pow(cos(x-a),2);
      printf("For x=%5.1f\ty=%6.2f\n",x,y);
      if (y>a) S+=y;
      else
      {
        k++;
        if (y<a) P*=y;
      }
    }
    else
      printf("For x=%5.1f\ty=Error!\n",x);
    x+=dx;
  }
  while (x<=xk);  //Конец тела цикла
  printf("S=%6.2f\tP=%6.2f\tk=%d\n",S,P,k);
  system("PAUSE");
  return EXIT_SUCCESS;
}
```

```
Input a=0.4
Input xn=0
Input xk=5
Input dx=1
For x=  0.0     y=Error!
For x=  1.0     y=  0.68
For x=  2.0     y= -0.69
For x=  3.0     y= -0.36
For x=  4.0     y= -0.58
For x=  5.0     y= -1.60
S=  0.68        P=  0.23        k=4
Для продолжения нажмите любую клавишу . . .
```

Рисунок 10 – Результаты работы программы

Таблица 4. Варианты для задания №2

№ вар.	***y=f(x)***	№ вар.	***y=f(x)***
1	$y=1+\cos^2\sqrt{x+a}$ $S=\sum_{y\le 1} y \quad P=\prod_{y>1} y$ Количество $y\in[0.5;1.5]$	2	$y=\sin x+\sqrt{ax}$ $R=\sum_{y>0} y+\prod_{y<0} y$ Количество $y>a$
3	$y=e^{a+x}+\ln ax$ $S=\sum_{y>a} y \quad P=\prod_{y>a} y$ Количество $y<a$	4	$y=\cos^2\pi x+\sqrt{x+a}$ $S=\sum_{y>0.5a} y \quad P=\prod_{y\le 2a} y$ Количество $y<2\cdot a$
5	$y=\sqrt{b+x}+e^{bx}$ $S=\sum_{y\ge b} y \quad P=\prod_{y>b} y^2$ Количество y	6	$y=1/x+\sin^2(x+a)$ $S=\sum_{y<0.5} y \quad P=\prod_{y\ge 0.5} y$ Количество $y\notin[0.5;1]$
7	$y=(\ln a+e^x)/x$ $S=\sum_{y<0} y+\prod_{y\ge a} y$ Количество $y<a$	8	$y=\sqrt{\lvert x+b\rvert}\,/\,x-e^x$ $S=\sum_{y<b}(y+b) \quad P=\prod_{y\ge b} y$ Количество $y<2b$
9	$y=e^x+\sin^3 x/(x-a)$ $S=\sum_{y>5} 1/y \quad P=\prod_{1\le y<10} y$ Количество $y\in[-a;a]$	10	$y=\sin^2 x/(1+x)+a^{-x}$ $P=\prod_{y<a} y$ Среднее значение $y>a$
11	$y=\sin(\pi x)/\cos(ax)$ $P=\prod_{y\ne 0} y$ Процент $y<0$ и $y>0$	12	$y=x^2+\sqrt{x+b}$ $S=\sum_{y>x} y-\prod 1/y$ Процент $y>x$

ЗАДАНИЕ №3. ОРГАНИЗАЦИЯ ВЛОЖЕННЫХ ЦИКЛОВ

1. Основные теоретические положения

Вложенные циклы выполняют перебор значений нескольких переменных одновременно [2,3,4]. Каждый из них организовывается по стандартному принципу (может быть любого из трех типов) и осуществляет перебор только одного параметра. При этом первый цикл называется внешним, а вложенные в него – внутренними. Границы внутреннего цикла не могут выходить за границы внешнего по отношению к нему цикла.

Для каждого значения параметра внешнего цикла происходит перебор всех возможных значений параметра внутреннего цикла. Всегда выполняется в первую очередь самый внутренний цикл. Такая организация циклов дает возможность перебрать значения их параметров во всех возможных комбинациях.

Задание №3. Составить блок-схему алгоритма и программу на языке С++

для расчета указанной модели при всех комбинациях ***a*** и ***b***, значения которых заданы в виде интервалов $an \le a \le ak$ с шагом Δa и $bn \le b \le bk$ с шагом Δb соответственно. Циклы по параметрам ***a*** и ***b*** должны быть разного типа. Программу реализовать в виде консольного приложения (**Console Application**). Для ввода-вывода данных использовать функции ввода-вывода (**scanf** и **printf**). Варианты задания приведены в таблице 5.

2. Пример выполнения задания №3

Постановка задачи:

1. Исходные данные: *an, ak, Δa, bn, bk, Δb*.
2. Математическая модель:

$$x = \cos b + 0.2$$

$$y = \begin{cases} x^2 + |a - b|, & \text{если} \quad x < 0.6 \\ ab - x/(a - x), & \text{если} \quad x \ge 0.6 \end{cases}$$

3. Ограничения: знаменатель $a - x \neq 0$.
4. Выходные данные: *a, b, x, y*.
5. Блок-схема алгоритма (рис. 11).

Внешним рационально выбрать цикл по параметру *b*, так как *x* зависит только от *b*. Значение *x* необходимо вычислять только во внешнем цикле, это позволит избежать многократного вычисления одних и тех же значений *x*. Во внутреннем цикле по параметру *a* вычисляется значение *y*.

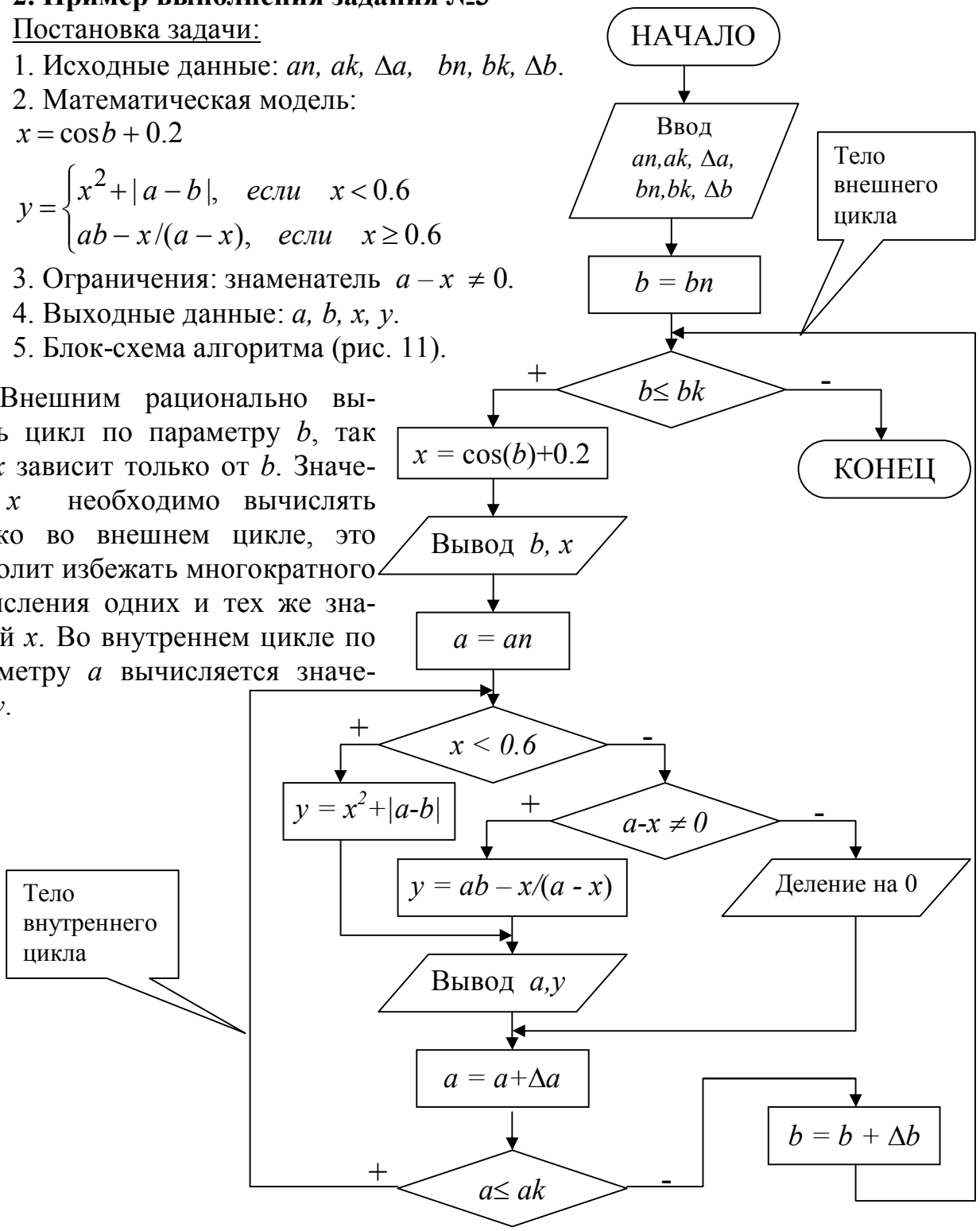


Рисунок 11– Блок-схема алгоритма для примера 3

6. Программа решения задачи на C++. Результаты работы программы показаны на рисунке 12.

```
#include <cstdlib>
#include <iostream>
#include <math.h>
#include <stdio.h>
using namespace std;
int main(int argc, char *argv[])
{ float a,b,x,y;
  float an,ak,da,bn,bk,db;
  printf("Input an="); scanf("%f",&an);
  printf("Input ak="); scanf("%f",&ak);
  printf("Input da="); scanf("%f",&da);
  printf("Input bn="); scanf("%f",&bn);
  printf("Input bk="); scanf("%f",&bk);
  printf("Input db="); scanf("%f",&db);
  b=bn;
  //Начало внешнего цикла с предусловием
  while (b<=bk)
  {
    x=cos(b)+0.2;
    printf("b=%5.1f\tx=%6.2f\n",b,x);
    a=an;
    //Начало внутреннего цикла с постусловием
    do
    {
      if (x<0.6) y=x*x+fabs(a-b);
      else
        if (a-x!=0) y=a*b-x/(a-x);
        else
        { printf("\ta=%5.1f\ty=Error!\n",a);
          goto m1; }
      printf("\ta=%5.1f\ty=%6.2f\n",a,y);
      m1:
      a+=da;
    }
    while (a<=ak);
   //Конец внутреннего цикла
    b+=db;
  } //Конец внешнего цикла
  system("PAUSE");
  return EXIT_SUCCESS;
}
```

```
Input an=1
Input ak=3
Input da=1
Input bn=2
Input bk=6
Input db=2
b=  2.0 x= -0.22
        a=  1.0 y=  1.05
        a=  2.0 y=  0.05
        a=  3.0 y=  1.05
b=  4.0 x= -0.45
        a=  1.0 y=  3.21
        a=  2.0 y=  2.21
        a=  3.0 y=  1.21
b=  6.0 x=  1.16
        a=  1.0 y= 13.24
        a=  2.0 y= 10.62
        a=  3.0 y= 17.37
Для продолжения нажмите любую клавишу .
```

Рисунок 12 – Результаты работы программы

Таблица 5. Варианты для задания №3

№ вар.	Модель	№ вар.	Модель
1	$z = ax - \frac{\sqrt{ab}}{0.2x + 0.5b},\ x = \frac{b}{a + 0.1}$	2	$z = \sqrt{a^2 + x^2} - \sqrt{\frac{a}{b + 2a}}, x = \frac{1}{b + 2}$
3	$z = y / a + \sqrt{\lvert a + 3b \rvert},\ y = \frac{\sin b}{2b}$	4	$z = ax - \sqrt{\frac{b}{e^x + 0.2}},\ x = 2a^2 + 0.3$
5	$z = \frac{x^2 + \sqrt{b^4 + 1}}{\sin(x + a)},\ x = \sqrt{2b + 0{,}2a}$	6	$z = \frac{\cos x - a}{\sqrt{ax}} + 2b,\quad x = 0.5a^2 - 2$
7	$z = \frac{ax - \sqrt{bx + 5}}{bx + 5},\quad x=3a+e^2$	8	$z = \frac{\sqrt{xb} - ax}{2 + \cos(bx + 3)},\quad x = \frac{a + 0.7}{3}$
9	$z = \frac{bx - a}{0.2 + x^2} - \sqrt{x + a},\ x = b^2 - 2$	10	$z = \frac{ax^3 + \sqrt{a + x}}{e^b + 1.5},\quad x = 2\sqrt{a} - \frac{1}{a}$
11	$z = \frac{\sqrt{a + 2} + b}{\sqrt{x^2 + 1.7}},\quad x = e^b - \cos^2 a$	12	$z = \frac{\sqrt{a^2 + \lvert b \rvert} - x}{x^2 + b},\quad x = \frac{a + 1.5}{a}$

ЗАДАНИЕ №4. ОБРАБОТКА ОДНОМЕРНЫХ МАССИВОВ

1. Основные теоретические положения

Массив – это последовательность однотипных элементов, которые имеют одно и тоже имя, но однозначно определяются своим номером (***индексом***) [2,3,4]. В одномерном массиве каждый элемент имеет один индекс, определяющий положение элемента в массиве. В С++ принято начинать нумерацию элементов массива с 0.

Основными характеристиками массива являются:

– размерность, т.е. количество элементов (обычно обозначается N);

– значения элементов (например, $X_0 = 2$; $X_5 = 1$ и т.д.).

Обработка массива обычно заключается в последовательном переборе его элементов и выполнении над ними однотипных операций, т.е. обработка массива является циклическим вычислительным процессом. Для этого достаточно организовать цикл по перебору индексов элементов массива от 0 до N-1 (N – размерность массива). Наиболее рационально использовать цикл «***Для***» на основе блока модификации (рис. 13).

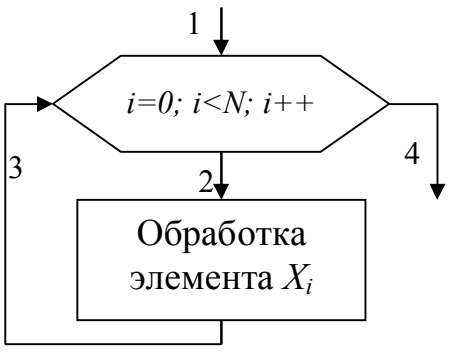


Рисунок 13 – Цикл «Для»

Рассмотрим принцип работы цикла «Для» на основе блока модификации.

При входе в блок модификации (линия 1) автоматически выполняются следующие действия. Параметру цикла *i* присваивает-

ся начальное значение 0 и проверяется, не превышает ли оно значения N–1 (номер последнего элемента в массиве). Если результатом проверки условия является истина, то происходит переход к телу цикла (линия 2). После выполнения тела цикла осуществляется возврат в блок модификации (линия 3), увеличение параметра цикла i на значение шага 1 и проверка условия продолжения цикла и т.д. Когда текущее значение параметра цикла i превысит значение N–1, цикл завершит свою работу (линия 4).

Ввод элементов массива выполняется в два этапа. Вначале указывается его размерность, а затем задаются значения для каждого элемента массива.

Алгоритм ввода массива X размерностью N (i=0÷N–1) следующий (рис. 14). На первом этапе вводится размерность массива N (блок 1). На втором этапе организовывается цикл «Для» на основе блока модификации, выполняющий поэлементный ввод произвольных однотипных значений компонент массива. Параметр цикла i в тоже время является индексом элементов массива X. Блок модификации (блок 2) перебирает значения i от 0 до N–1 и для каждого значения i выполняется ввод значения X_i-го элемента массива (блок 3), т.е. при i =0 будет введено значение для X_0, при i =1 – для X_1 и т.д. Таким образом, за один шаг выполнения цикла вводится один элемент массива X.

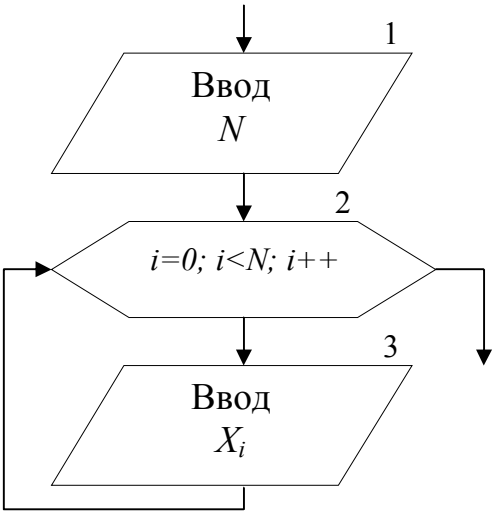


Рисунок 14 – Алгоритм ввода

Вывод массива также выполняется поэлементно с помощью цикла «Для».

Циклы по вводу или выводу элементов массивов не обязательно делать отдельно. Их можно объединять с циклами по обработке элементов массивов.

Задание №4. Составить блок-схему алгоритма и программу на языке С++, которая для заданного массива вещественных чисел X[N] решает поставленную задачу. Использовать статические массивы. Программу реализовать в виде консольного приложения (**Console Application**). Для ввода элементов массива использовать поток **cin**, для вывода результатов – **cout**. Варианты задания приведены в таблице 6.

2. Пример выполнения задания №4

Постановка задачи:

1. Исходные данные: массив X, размерностью $i = 0 \div N–1$.

2. Задача: на основе элементов массива X[N], вычислить значения элементов массива Y[N], по формуле $y_i = \sin x_i + 0.1e^{x_i}$. Определить сумму ($S$) всех и произведение ($P$) положительных элементов массива Y. Считая пары элементов массивов (x_i , y_i) координатами точек на плоскости X0Y, определить количество (k) точек расположенных в I квадранте.

3. Выходные данные: массив Y, S, P, k.

4. Ограничения: нет.

5. Блок-схема алгоритма (рис. 15):

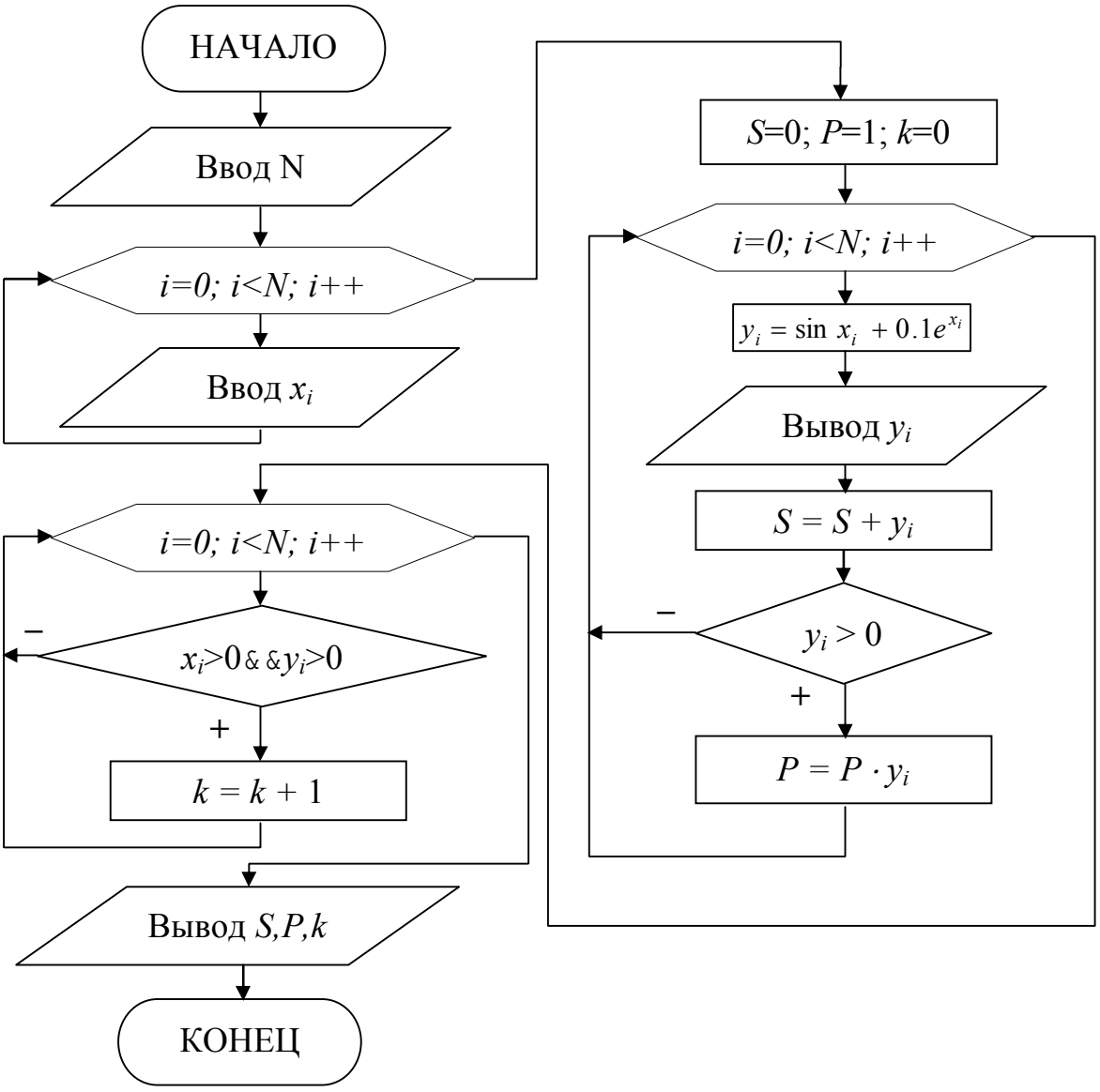


Рисунок 15– Блок-схема алгоритма для примера 4

6. Программа решения задачи на С++. Результаты работы программы показаны на рисунке 16.

```
#include <iostream>
#include <math.h>
int main()
{float x[100], y[100];  //Описание массивов
 float S, P;
 int i, N, k;
 cout<<"\nInput N="; cin>>N; //Ввод размерности массива
 //Ввод элементов исходного массива X
 for(i=0; i<N; i++)
 { cout<<"X["<<i<<"]="; cin>>x[i]; }
 cout<<endl;
```

```
 S=0; P=1; k=0;
 for(i=0; i<N; i++)
 { //Формирование массива Y и вывод его элементов
   y[i]=sin(x[i])+0.1*exp(x[i]);
   cout<<"Y["<<i<<"]="<<y[i]<<"\t";
   S=S+y[i];
   if (y[i]>0) P*=y[i];
 }
 cout<<endl;
 for(i=0; i<N; i++)
  if (x[i]>0 && y[i]>0) k++;
 cout<<"S="<<S<<endl<<"P="<<P<<endl<<"k="<<k<<endl;
}
```

```
Input N=5
X[0]=2
X[1]=6
X[2]=-4
X[3]=3
X[4]=-1

Y[0]=1.6482     Y[1]=40.0635    Y[2]=0.758634   Y[3]=2.14967    Y[4]=-0.804683

S=43.8153
P=107.687
k=3
Для продолжения нажмите любую клавишу . . .
```

Рисунок 16 – Результаты работы программы

Таблица 6. Варианты для задания №4

№ вар.	Задача
1	**2**
1	На основе элементов массива X[N], вычислить значения элементов массива Y[N], по формуле $y_i = 5\cos^2 x_i$. Определить сумму и произведение элементов массива Y, значения которых принадлежат интервалу [2.5; 3.5]. Считая пары элементов массивов (x_i, y_i) координатами точек на плоскости X0Y, определить количество точек расположенных в I квадранте.
2	На основе элементов массива X[N], вычислить значения элементов массива Y[N], по формуле $y_i = 2x_i - \cos x_i^2$. Определить сумму и произведение положительных элементов массива Y. Считая пары элементов массивов (x_i, y_i) координатами точек на плоскости X0Y, определить количество точек расположенных во II и III квадрантах.
3	На основе элементов массива X[N], вычислить значения элементов массива Y[N], по формуле $y_i = e^{2-x_i} - \sin x_i^3$. Определить сумму отрицательных и произведение положительных элементов массива Y. Считая пары элементов массивов (x_i, y_i) координатами точек на плоскости X0Y, определить количество точек расположенных в I и IV квадрантах.

Продолжение таблицы 6

1	2
4	На основе элементов массива X[N], вычислить значения элементов массива Y[N], по формуле $y_i = \ln(\sin^2 x_i + 0.5)$. Определить сумму и произведение элементов массива Y, значения которых больше 0.5. Считая пары элементов массивов (x_i, y_i) координатами точек на плоскости X0Y, определить долю точек расположенных в III квадранте.
5	На основе элементов массива X[N], вычислить значения элементов массива Y[N], по формуле $y_i = 1 + x_i \sin(x_i + \pi)$. Определить среднее арифметическое значение элементов массива Y. Считая пары элементов массивов (x_i, y_i) координатами точек на плоскости X0Y, определить количество точек расположенных во II и IV квадрантах.
6	На основе элементов массива X[N], вычислить значения элементов массива Y[N], по формуле $y_i = 2.5x_i - \sqrt[3]{x_i + 1}$. Определить сумму положительных и произведение отрицательных элементов массива Y. Считая пары элементов массивов (x_i, y_i) координатами точек на плоскости X0Y, определить количество точек расположенных в I квадранте.
7	На основе элементов массива X[N], вычислить значения элементов массива Y[N], по формуле $y_i = 2\sin(\cos(x_i + \pi/2))$. Определить среднее геометрическое значение положительных элементов массива Y. Считая пары элементов массивов (x_i, y_i) координатами точек на плоскости X0Y, определить количество точек расположенных в III квадранте.
8	На основе элементов массива X[N], вычислить значения элементов массива Y[N], по формуле $y_i = x_i - \sqrt{\cos^2 x_i + 1}$. Определить среднее арифметическое значение элементов массива Y, которые попадают в интервал [-2; 2]. Считая пары элементов массивов (x_i, y_i) координатами точек на плоскости X0Y, определить количество точек расположенных в I квадранте.
9	На основе элементов массива X[N], вычислить значения элементов массива Y[N], по формуле $y_i = x_i + \sin^3 x_i - 1$. Определить сумму и произведение элементов массива Y, значения которых не больше 1.5. Считая пары элементов массивов (x_i, y_i) координатами точек на плоскости X0Y, определить процент точек расположенных в I квадранте.
10	На основе элементов массива X[N], вычислить значения элементов массива Y[N], по формуле $y_i = \sqrt{x_i^2 + 1} - \cos x_i$. Определить среднее геометрическое значение элементов массива Y, которые попадают в интервал [0.5; 2.5]. Считая пары элементов массивов (x_i, y_i) координатами точек на плоскости X0Y, определить количество точек расположенных во II квадранте.
11	На основе элементов массива X[N], вычислить значения элементов массива Y[N], по формуле $y_i = \ln(\lvert x_i \rvert + \lvert \cos x_i \rvert)$. Определить сумму и произведение элементов массива Y, значения которых не меньше 1. Считая пары элементов массивов (x_i, y_i) координатами точек на плоскости X0Y, определить долю точек расположенных во II квадранте.
12	На основе элементов массива X[N], вычислить значения элементов массива Y[N], по формуле $y_i = \sqrt[5]{5x_i + 1} - \cos(2.5x_i)$. Определить сумму отрицательных и произведение положительных элементов массива Y. Считая пары элементов массивов (x_i, y_i) координатами точек на плоскости X0Y, определить количество точек расположенных в I квадранте.

ЗАДАНИЕ №5. ОБРАБОТКА ДВУМЕРНЫХ МАССИВОВ

1. Основные теоретические положения

Двумерный массив (матрица) представляет собой таблицу (рис. 17), на пересечении строк и столбцов которой располагаются элементы [2,3,4]. Каждый элемент имеет два индекса. Первый индекс обозначается буквой ***i*** и указывает номер строки, в которой расположен элемент. Второй индекс обозначается буквой ***j*** и указывает номер столбца, в котором расположен элемент. Размерность двумерного массива задается двумя числами: ***M*** – количество строк и ***N*** – количество столбцов.

Номера столбцов

Номера строк	$j=0$	$j=1$	$j=2$	...	$j=N-1$
$i=0$	$A_{0,0}$	$A_{0,1}$	$A_{0,2}$	...	$A_{0,N-1}$
$i=1$	$A_{1,0}$	$A_{1,1}$	$A_{1,2}$	...	$A_{1,N-1}$
...	…	...	...	$A_{i,j}$	...
$i=M-1$	$A_{M-1,0}$	$A_{M-1,1}$	$A_{M-1,2}$	...	$A_{M-1,N-1}$

Рисунок 17 – Двумерный массив

Двумерный массив, у которого количество строк равно количеству столбцов называется квадратной матрицей, в противном случае – прямоугольной.

Для обработки двумерного массива требуется два вложенных цикла «Для» на основе блока модификации (рис. 18). Первый цикл будет перебирать строки, второй – столбцы массива. Внешний цикл при ***i*** =0 «выбирает» 0-ю строку массива. Внутренний цикл перебирает все столбцы массива, т.е. поочередно выбираются элементы $A_{0,0}$, $A_{0,1}$, $A_{0,2}$ и т.д. до конца 0-й строки. После выхода из внутреннего цикла происходит возврат во внешний бок модификации, где выбирается 1-я строка массива, для которой внутренний цикл опять переберет поочередно все элементы $A_{1,0}$, $A_{1,1}$, $A_{1,2}$ и т.д. Таким образом, элементы двумерного массива будут перебираться по строкам. Если поменять местами параметры внешнего и внутреннего циклов, т.е. внешний цикл сделать по параметру ***j***, а внутренний – по параметру ***i***, то элементы массива будут перебираться по столбцам.

Рисунок 18 – Обработка массива

Задание №5. Составить блок-схему алгоритма и программу на языке С++, для решения поставленной задачи. Использовать статические массивы. Программу реализовать в виде консольного приложения (**Console Application**). Для ввода элементов массива использовать поток **cin**, для вывода результатов – **cout**. Варианты задания приведены в таблице 7.

2. Пример выполнения задания №5

Постановка задачи:

1. Исходные данные: двумерный массив A, размерностью $i = 0 \div M-1$, $j = 0 \div N-1$.

2. Задача: сформировать одномерный массив $B = (b_0, b_2, \ldots, b_{N-1})$, каждый элемент которого равен среднему арифметическому значению элементов соответствующего столбца двумерного массива A.

3. Выходные данные: массив B.

4. Блок-схема алгоритма (рис .19):

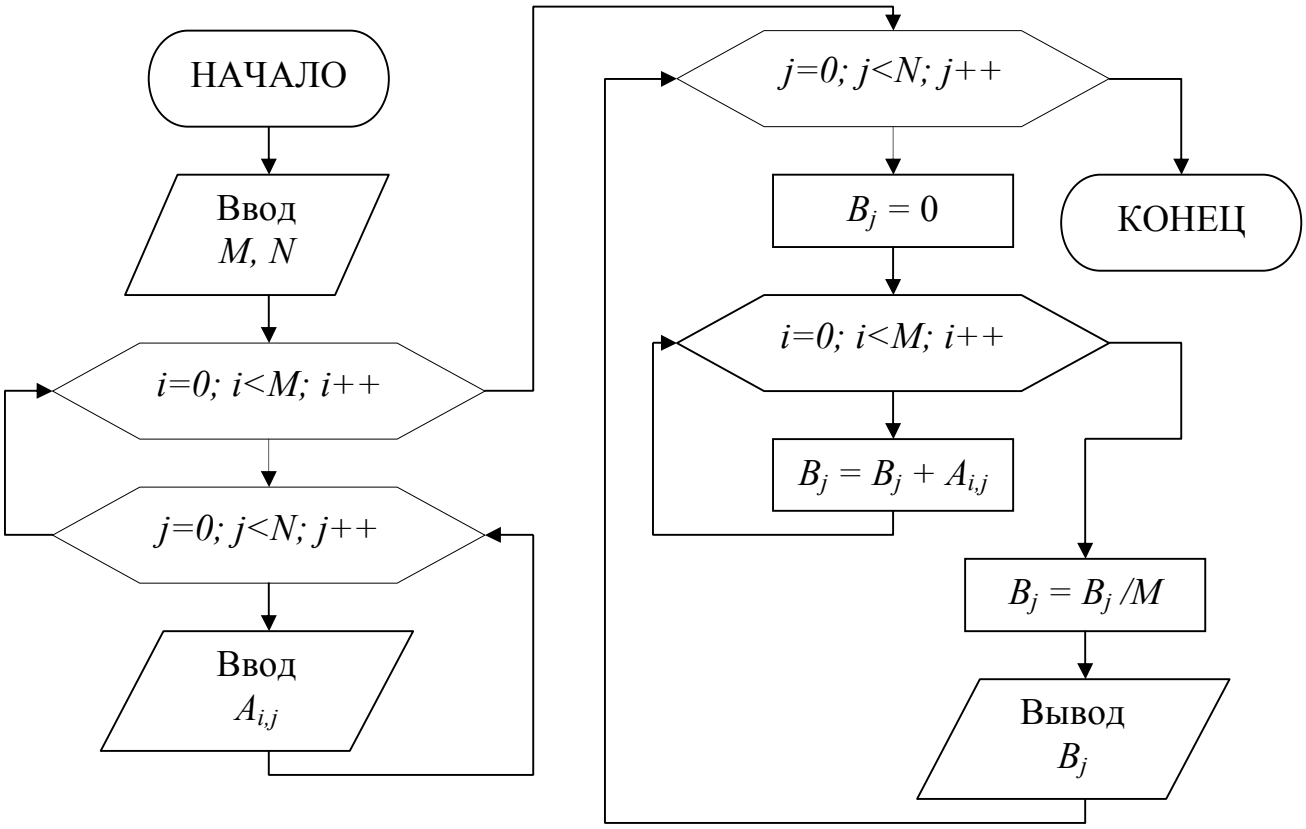


Рисунок 19– Блок-схема алгоритма для примера 5

5. Программа решения задачи на С++.

```
#include <cstdlib>
#include <iostream>

int main(int argc, char *argv[])
{
 float A[10][10], B[10];  //Описание массивов
 int i, j, M, N, k;
 //Ввод размерности массива
 cout<<"Input M="; cin>>M;
 cout<<"Input N="; cin>>N;
 //Ввод элементов исходного массива A
 for(i=0; i<M; i++)
 for(j=0; j<N; j++)
```

```
 { cout<<"A["<<i<<"]["<<j<<"]=";  cin>>A[i][j]; }
 cout<<endl;
 for(j=0; j<N; j++)
 { B[j]=0;
   //Формирование массива B и вывод его элементов
   for(i=0; i<M; i++)
    B[j]=B[j]+A[i][j];
   B[j]=B[j]/M;
   cout<<"B["<<j<<"]="<<B[j]<<"\t";
 }
  system("PAUSE");
  return EXIT_SUCCESS;
}
```

Таблица 7. Варианты для задания №5

№ вар.	Задача
1	Найти в каждом столбце матрицы P(N,N) наименьший элемент и поменять его местами с соответствующим элементом побочной диагонали.
2	Сформировать вектор D(M), каждый элемент которого равен среднему арифметическому ненулевых значений элементов строк матрицы C(M,N).
3	В матрице C(M,L) заменить максимальный элемент каждого столбца произведением положительных элементов этого столбца.
4	Сформировать вектор B(N), каждый элемент которого равен разности максимального и минимального элементов соответствующего столбца матрицы A(M,N).
5	В матрице A(K,L) максимальный элемент заменить количеством нулевых элементов матрицы, минимальный элемент заменить средним арифметическим значением положительных элементов матрицы.
6	В каждом столбце матрицы X(M,N) найти среднее арифметическое значение элементов и определить номер столбца с максимальным значением среднего арифметического.
7	В каждой строке матрицы B(K,L) минимальный элемент заменить средним арифметическим значением отрицательных элементов этой же строки.
8	Сформировать вектор G(K), каждый элемент которого равен отношению количества положительных к количеству отрицательных элементов соответствующей строки матрицы A(K,M).
9	Определить количество нулевых элементов матрицы A(M,N) расположенных вне главной и побочной диагоналей. Вывести индексы последнего по порядку такого элемента.
10	Преобразовать матрицу A(K,L) так, чтобы последний элемент каждой строки был заменен суммой предыдущих элементов строки уменьшенной на значение максимального элемента этой же строки.
11	Сформировать вектор B(K), каждый элемент которого содержит номер столбца максимального отрицательного элемента соответствующей строки матрицы C(K,N).
12	Получить вектор B, каждый элемент которого равен среднему геометрическому положительных значений элементов соответствующей строки матрицы A(M,K).

СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

1. Методические рекомендации для самостоятельной работы по дисциплине «Информатика» : для обучающихся по направлению подготовки 13.03.02 «Электроэнергетика и электротехника» всех форм обучения / ГОУВПО «ДОННТУ», Каф. прикладной математики и искусственного интеллекта ; сост. К. Н. Ефименко. – Донецк : ДОННТУ, 2022. – Систем. требования: Acrobat Reader. – Загл. с титул. экрана.

2. Прата С. Язык программирования С [Электронный ресурс] : лекции и упражнения / С. Прата ; С. Прата ; пер. с англ., под ред. Ю.Н. Артеменко. - 6-е изд. - 15 Мб. - Москва : Вильямс, 2015. - 1 файл. - Систем. требования: Acrobat Reader. http://ed.donntu.ru/books/19/cd9100.pdf

3. Мейерс С. Эффективный и современный С++ [Электронный ресурс] : 42 рекомендации по использованию С++11 и С++14 / С. Мейерс ; С. Мейерс ; пер. с англ., ред. И.В. Красикова. - 13 Мб. - Москва ; Санкт-Петербург ; К. : Вильямс, 2016. - 1 файл. - (O'Reilly). - Систем. требования: Acrobat Reader. http://ed.donntu.ru/books/19/cd9081.pdf

4. Ефименко, К. Н. Основы алгоритмизации: методическое пособие/ К. Н. Ефименко, Ю. Н. Добровольский, В. С. Ильяшенко. – Донецк : ДОННТУ, 2006. – 75 с. http://ea.donntu.ru:8080/jspui/bitstream/123456789/7553/1/%d0%9c%d0%b5%d1%82%d0%be%d0%b4%d0%b8%d1%87%d0%ba%d0%b0_7.pdf

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ
для выполнения индивидуального задания
по дисциплине «Информатика»

Составитель:

Ефименко Константин Николаевич – кандидат технических наук, доцент кафедры прикладной математики и искусственного интеллекта ГОУВПО «ДОННТУ».

Ответственный за выпуск:

Павлыш Владимир Николаевич – заведующий кафедрой прикладной математики и искусственного интеллекта ГОУВПО «ДОННТУ», доктор технических наук, профессор.