МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ

НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ХАРЬКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ»

ОСНОВЫ ПРОГРАММИРОВАНИЯ НА ЯЗЫКЕ С++. ОДНОМЕРНЫЕ МАССИВЫ И ЦИКЛЫ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
для выполнения практических и лабораторных работ

по курсу
«ИНФОРМАТИКА»

для студентов факультета «Автоматика и приборостроение»
дневной и заочной форм обучения

Утверждено
редакционно-издательским
советом университета,
протокол № 2 от 12.12.2013 г.

Харьков
НТУ «ХПИ»
2014

Основы программирования на языке С++. Одномерные массивы и циклы: методические указания для выполнения практических и лабораторных работ по курсу «Информатика» для студентов факультета «Автоматика и приборостроение» дневной и заочной форм обучения / сост.: Е. Е. Тверитникова, В.А. Крылова, М. И. Опрышкина – Х.: НТУ «ХПИ», 2014. – 72 с.

Составители: Е. Е. Тверитникова
В. А. Крылова
М. И. Опрышкина

Рецензент А. В. Ивашко

Кафедра автоматики и управления в технических системах

Вступление

Развитие современных технологий невозможно без использования компьютерной техники и программного обеспечения. Подготовка специалистов в области автоматики, измерительной и медицинской техники требует обширных знаний и навыков владения вычислительной техникой, а также знания основ алгоритмизации и программирования на языках высокого уровня таких как С/С++.

Методические указания предназначены для изучения языка программирования С++ на практических занятиях и лабораторных работах, а также для самостоятельного освоения. В методических указаниях на примерах рассматриваются основные средства программирования, а также создания блок-схем алгоритмов программ на языке С++, а именно: цикл с предусловием *while,* цикл с постусловием *do … while*, параметрический цикл *for,* использование одномерных массивов. Приведены индивидуальные задания для выполнения лабораторных работ и задание для самостоятельного изучения основ программирования. Рассмотрены примеры и задания, которые помогут эффективному освоению курса «Информатика».

ЛАБОРАТОРНАЯ РАБОТА 5

РАЗРАБОТКА ПРОГРАММ НА ЯЗЫКЕ С++ С ИСПОЛЬЗОВАНИЕМ ЦИКЛА С ПРЕДУСЛОВИЕМ *while*

Цель работы – приобретение и закрепление практических навыков при составлении циклических программ на языке С++ с использованием цикла с предусловием ***while***.

Цикл с предусловием *while*

Цикл – это специальный оператор языка программирования, с помощью которого то или иное действие можно выполнить нужное количество раз, в зависимости от некоего условия. Действия, которые повторяются, называются <*тело цикла*> (набор повторяющихся операторов). Условие, в течение которого действие выполняется, называется <*условие выполнения цикла*>. При написании любого цикла надо иметь в виду, что в нем всегда явно или неявно присутствуют четыре элемента: начальные установки, тело цикла, модификация параметра цикла и проверка условия продолжения цикла.

На языке программирования С++ цикл с предусловием реализован оператором ***while*** (приложение А, рис. А.1). В цикле с предусловием ***while*** перед началом выполнения тела цикла проверяется *условие* выполнения цикла. Параметр *условие* – выражение логического типа, либо будет к нему приведено. Сначала проверяется параметр *условие,* если значение *true*, то выполняется оператор, описанный в теле цикла. Если *false* ,то цикл не выполняется и происходит выход из цикла. Если условие всегда *true,* то оператор в теле цикла будет выполняться всегда, и цикл станет бесконечным. Необходимо создать ситуацию, когда параметр *условие* принимает значение *false*, для того, чтобы обеспечить выход из циклического алгоритма. Один из способов – использование изменений переменных внутри цикла и проверка изменений при проверке условия. Таким образом, проверка *условия* повторяется при каждом выполнении цикла. Как только оно перестает быть верным, цикл завершается. Если условие ложно (*false*) с самого начала, действие внутри цикла (*тело цикла*) не будет выполнено ни разу.

Пример.

Написать программу (используя цикл ***while***), которая находит сумму всех целых чисел от 1 до 10 включительно.

```
int main ( )
{
    int  i, s;
    s = 0; // переменная для накопления суммы
    i = 1; // управляющая переменная цикла
    while  (i <= 10)  // проверка условия, сравнение управляющей переменной с окончанием диапазона
    {
      s = s + i; // накапливание суммы
      i++; // изменение управляющей переменной
    }
    cout<< "s = " << s << '\n'; // вывод результата на экран
    return 0;
}
```

В начале программы объявляется переменная ***i* = 1,** а также переменная для накапливания суммы ***s = 0***. Далее в условии цикла проверяется условие – значение переменной ***i*** меньше или равно **10**. Поскольку именно от этой переменной зависит, будет ли цикл выполняться или нет, то такая переменная называется управляющей переменной цикла. В теле цикла выполняется накапливание суммы ***s = s + i*** – к предыдущему значению переменной ***s*** прибавляется текущее значение переменной ***i*** и новое значение перезаписывается в переменную ***s.*** Также в теле цикла выполняется увеличение значения переменной ***i*** на **1**. Данное действие является обязательным, так как если не изменять значение переменной управляющей циклом, результат проверки условия тоже никогда не изменится. Это может привести к тому, что цикл будет выполняться бесконечно (вечный цикл). Во избежание таких ошибок нужно внимательно следить, чтобы внутри тела цикла происходило изменение управляющей переменной. Далее программа возвращается на проверку условия цикла (***while (i* <= 10*)***): если условие принимает значение *true*, то опять выполняется тело цикла, если условие – *false*, то происходит

выход из цикла и выполнение оператора, следующего за циклом, а именно вывод на экран значения переменной ***s.***

ПРИМЕРЫ РЕШЕНИЯ ЗАДАНИЙ

Задание 5.1

Составить программу вычисления выражения $y = 2 \cdot x!$. Результат вывести на экран.

I. *Выбор метода*

Факториал числа вычисляется как произведение предыдущего числа на последующее (увеличенное на единицу). Для данного выражения необходимо воспользоваться формулой $x! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot (x-1) \cdot x$. Таким образом, для вычисления факториала числа x в программе необходимо организовать цикл с начальными установками, равными единице, а также условием выхода из цикла, когда начальные установки достигнут значения x. Тело цикла – произведение факториала и увеличение на единицу начальных установок.

II. *Описание решения задачи на псевдокоде*

1. Начало.
2. Ввести с клавиатуры значение x.
3. Вычислить значение $f = x!$.
4. Вычислить значение $y = 2 \cdot f$.
5. Вывести результат y на экран.
6. Конец.

III. *Схема алгоритма работы программы*

Блок–схема алгоритма программы представлена на рисунке 1.1.

IV. *Разработка текста программы*

1. Подключаем в файле ***stdafx.h*** необходимые для работы программы библиотеки:

 # include <iostream> – для работы операторов ввода/вывода;

using namespace std;

2. Разработка раздела описания переменных

int x, y, f, i;

x – целое число, вводимое с клавиатуры;

f – вычисление факториал $x!$;

y – вычисление $y = 2 \cdot f$.

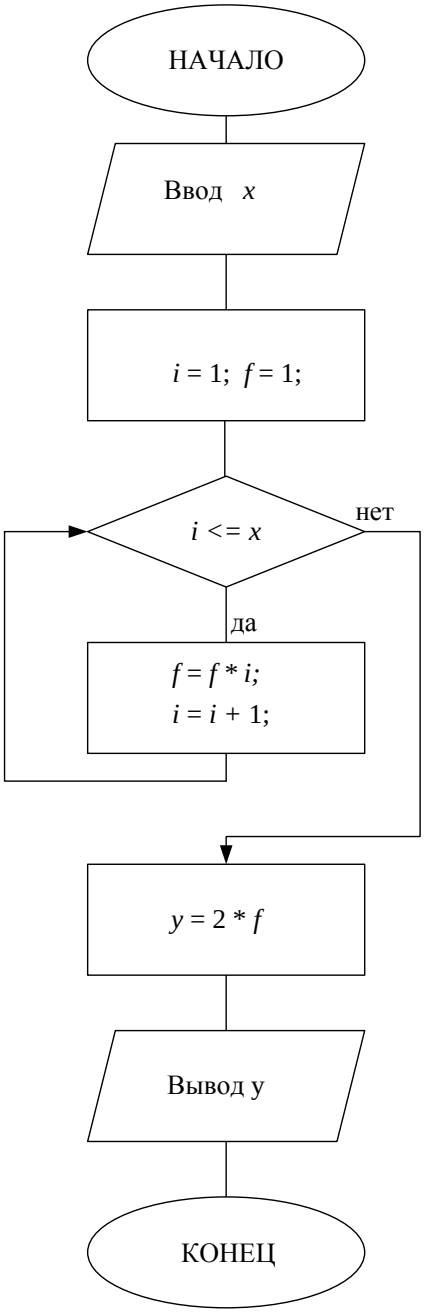

Рисунок 1.1 – Блок-схема алгоритма программы задания 5.1

3. Разработка тела программы

Вводим с клавиатуры исходные данные – целое число *x*. Для указания пользователю что вводить, используем оператор вывода ***cout***<< с соответствующим текстом. Затем используем оператор ввода ***cin***>> с указанием необходимых переменных.

cout*<< *" Введите целое число x \n";

cin*>> *x;

Для вычисления факториала необходимо установить

```
i = 1;
f = 1;
```

Цикл с предусловием для вычисления факториала числа *x*

```
while (i <= x)
{
f = f * i;
i = i + 1;
}
```

Синтаксис программы

```
# include<iostream>
using  namespace std;
int main( )
{
    int x, i, f, y; // описание переменных целого типа
    cout<<" Введите целое число x \n";
    cin>>x; // ввод значения переменной x
    i = 1;
    f = 1;
    while (i <= x) // условие выполнения цикла
    {
    f = f * i; // произведение предыдущего значения на следующее
    i = i + 1; // увеличение на единицу предыдущего значения
    }
    y =2 * f;
    cout<< " y= "<<y<<'\n'; // вывод значения y на экран
    return 0;
}
```

4. Отладка и запуск программы

Для отладки программы используем клавишу *F7*, убеждаемся в отсутствии ошибок и запускаем программу на исполнение с помощью комбинации клавиш *Ctrl+F5*. Ниже приведены результаты работы программы.

Введите целое число x

5

y = 240.

Задание 5.2

Написать программу, которая выводит на экран сумму цифр целого числа *n*, введенного с клавиатуры.

I. *Выбор метода*

Целое число *n* вводится с клавиатуры пользователем. Заведомо неизвестно, какое количество цифр в этом числе, т.к. вводимое пользователем число может быть – двухзначным, трёхзначным, четырёхзначным и т.д. Таким образом, для нахождения суммы цифр числа *n*, необходимо организовать цикл, в котором каждая цифра числа *n*, начиная с младшей, будет находиться как остаток от деления числа *n* на 10. При этом текущее значение переменной *n* в цикле, каждый раз после нахождения остатка от деления, должно менять своё значение с помощью выражения: $n = n/10$. Условием выхода из цикла является обнуление значения переменной *n*.

Например: целое число $n = 543$.

1. $a = n \% 10;$ $\quad a = 3;$
   $n = n / 10;$ $\quad n = 54;$
2. $a = n \% 10;$ $\quad a = 4;$
   $n = n / 10;$ $\quad n = 5;$
3. $a = n \% 10;$ $\quad a = 5;$
   $n = n / 10;$ $\quad n = 0.$

Для нахождения суммы цифр числа *n* необходимо организовать в теле цикла выражение для суммы, которое будет увеличиваться на значение цифры, и перезаписывать новое значение.

II. *Описание решения задачи на псевдокоде*

1. Начало.
2. Ввести значение *n*.
3. Вычислить значение выражения $n \% 10$.
4. Увеличить значение суммы *s*.
5. Вычислить значение $n = n / 10$.
6. Если *n* не равно 0, то повторить пп. 3, 4 и 5, иначе – пп. 7.
7. Вывести на экран значения суммы *s*.
8. Конец.

III. *Схема алгоритма решения задачи*

Блок–схема алгоритма программы показана на рисунке 1.2.

IV. *Разработка текста программы*

1. Подключаем в файле ***stdafx.h*** необходимые для работы программы библиотеки:

 # include <iostream>– для работы операторов ввода/вывода;

using namespace std;

2. Разработка раздела описания переменных

int n, a, s;

n – целое число, вводимое с клавиатуры;

a – значение цифры;

s – значение суммы.

3. Разработка тела программы

Исходные данные вводим с клавиатуры. Для подсказки, что пользователь должен ввести целое число, используем оператор вывода ***cout<<*** с соответствующим текстом. Затем используем оператор ввода ***cin>>*** с указанием необходимых переменных

cout<< " Введите целое число n \n ";

cin>>n;

Для вычисления суммы начальное значение переменной необходимо обнулить

s = 0;

Нахождение и суммирование цифр числа *n* выполняется в цикле с предусловием

```
while (n != 0)
    {
        a = n % 10;
        s = s + a;
        n = n / 10;
    }
```

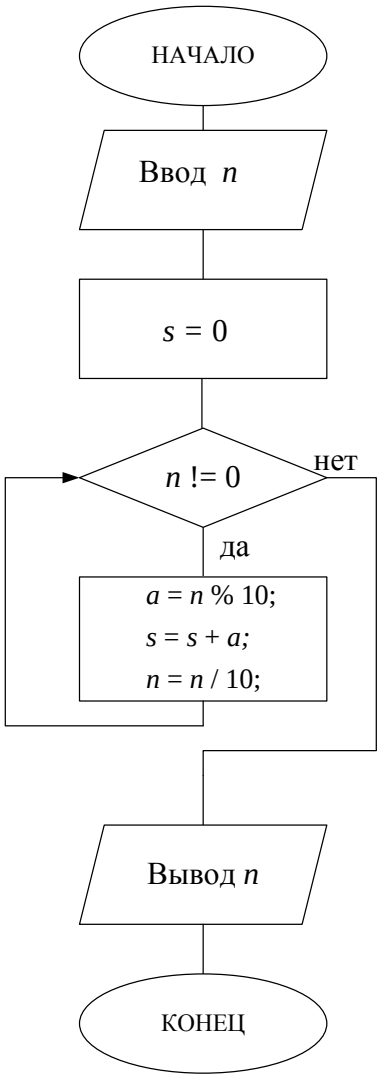

Рисунок 1.2 – Блок-схема алгоритма программы задания 5.2

Синтаксис программы

```
#include <iostream>
using  namespace std;

int main ( )
{
        int n, a, s; // описание переменных целого типа
        cout<< " Введите целое число n \n ";
        cin>>n; // ввод с клавиатуры целого числа n
        s = 0; // обнуление суммы
        while (n != 0) // условие выхода из цикла с предусловием
```

```
{
        a = n % 10; // вычисление цифры числа n
        s = s + a; // вычисление суммы
        n = n / 10; // изменение значения числа n
}
cout<< " Сумма цифр числа "<<n<< " = " <<s<<'\n'; // вывод на экран результата суммы
}
```

4. Отладка и запуск программы

Для отладки программы используем клавишу *F7*, убеждаемся в отсутствии ошибок и запускаем программу на исполнение с помощью комбинации клавиш *Ctrl+F5*. Ниже приведены результаты работы программы.

Введите целое число

849

Сумма цифр числа 849 = 21

Порядок выполнения лабораторной работы

1. В соответствии с номером по журналу выберите индивидуальное задание.
2. Разработайте алгоритм решения задачи.
3. Составьте текст программы.
4. Создайте проект в интегрированной среде разработки *Microsoft Visual Studio*.(*lab5*_фамилия.*cpp)*
5. Введите текст программы.
6. Скомпилируйте программу. Если в программе есть ошибки, их необходимо исправить. Если ошибок нет, то появится сообщение об успешной компиляции.
7. Запустите программу на выполнение, проанализируйте результаты и убедитесь в правильности решения задачи.
8. Выполните отчет по лабораторной работе, который должен содержать:

− титульный лист;

− цель работы;

− индивидуальное задание;

− алгоритм работы программы;

− текст программы;

− результаты работы программы;

− выводы.

Индивидуальные задания

1. Подсчитать количество цифр в десятичной записи целого неотрицательного числа n и вывести результат на экран.

2. Определить, является ли заданное натуральное число n палиндромом, т.е. таким, которое читается одинаково слева направо и справа налево.

3. Вывести на экран все делители заданного натурального числа n.

4. Определить число, полученное выписыванием в обратном порядке цифр заданного натурального числа n. Результат вывести на экран.

5. Составить программу нахождения суммы всех делителей целого числа n, вводимого с клавиатуры. Результат вывести на экран.

6. Составить программу нахождения произведения всех делителей целого числа n, вводимого с клавиатуры. Результат вывести на экран.

7. Составить программу нахождения среднего арифметического всех делителей числа целого числа n, вводимого с клавиатуры. Результат вывести на экран.

8. Составить программу вычисления суммы квадратов нечетных чисел от a до b, где a и b – целые числа, введенные с клавиатуры.

9. Составить программу вычисления выражения $y = (3 + n)!$. Результат вывести на экран.

10. Последовательность чисел a_0, a_1, a_2, ... образуется по закону: $a_0 = 1$; $a_k = k \cdot a_{k-1} + 1 / k$, где $(k = 1, 2, \ldots n)$. Натуральное число n, вводится с клавиатуры. Составить программу получения a_1, a_2, ..., a_n. Результат вывести на экран.

11. Составить программу нахождения наименьшего общего кратного (НОК) двух натуральных чисел.

12. Введите последовательность вещественных чисел, пока не введете 0. Составить программу определения суммы введенных чисел.

13. Составить программу, определяющую, является ли заданное целое число n степенью числа 5. Результат вывести на экран.

14. Составить программу, определяющую, все целые числа из промежутка от 100 до 300, у которых сумма делителей равна 50. Результат вывести на экран.

15. Составить программу поиска первых 20 натуральных чисел, делящихся нацело на 13 или 17 и больших 500. Результат вывести на экран.

16. Составить программу для нахождения всех натуральных решений (x и y) уравнения $x^2 + y^2 = n$, где x, y и n лежат в интервале от 1 до 30. Решения, которые получаются перестановкой x и y, считать совпадающими. Результат вывести на экран.

17. Составить программу, определяющую, количество точек с целочисленными координатами, принадлежащих кругу радиуса R с центром в начале координат.

18. Составить программу поиска количества «счастливых» билетов, у которых сумма первых трех цифр и последних трех чисел равны.

19. Составить программу поиска количества трехзначных натуральных чисел, сумма квадратов которых равна заданному целому числу n. Результат вывести на экран.

20. С клавиатуры вводится целое число n. Составить программу, которая выводит на экран цифры данного числа n.

Контрольные вопросы

1. Что будет на экране после выполнения следующего фрагмента кода?

```
int a = 0;
while (a = 1)
{
    if (a % 2 == 1)
    {
      break;
    }
    a++;
}
cout<<a;
```

Вариатны ответов: **А.** 0; **Б.** 1; **В.** 2; **Г.** Ошибка на этапе компиляции; **Д.** Ошибка на этапе выполнения.

2. Чему будет равно значение переменной *i* после выполнения следующего фрагмента кода?

```
int i = –3, a = –3;
while (a – i) {
        a = i++;
}
cout<<i;
```

Варианты ответов: **А.** *i* = –6; **Б.** *i* = –3; **В.** *i* = –2; **Г.** *i* = 1.

3. Что будет на экране после выполнения следующего фрагмента кода?

```
int i = 3;
while (i > 0) {
        i ––;
        while (i == 2) {
                break;
                i = –1;
        }
}
cout<<i;
```

Варианты ответов: **А.** *i* = 1; **Б.** *i* = 2; **В.** *i* = 0; **Г.** Ошибка на этапе компиляции; **Д.** Вечный цикл.

4. Что будет на экране после выполнения следующего фрагмента кода?

```
int i = 1, a = 2;
while (i > a)
{
        i++; a++;
}
cout<<i<<a;
```

Варианты ответов: **А.** *i* = 1, *a* = 2; **Б.** *i* = 2, *a* = 1; **В.** *i* = 0, *a* = 0; **Г.** Вечный цикл.

5. Что будет на экране после выполнения следующего фрагмента кода?

```
int i = 2, a;
while (i != a) {
        a = i % 3; i++;
}
cout<<a;
```

Варианты ответов: **А.** *a* = 2; **Б.** *a* = 0; **В.** *a* = 1; **Г.** Вечный цикл.

ЛАБОРАТОРНАЯ РАБОТА 6

РАЗРАБОТКА ПРОГРАММ НА ЯЗЫКЕ С++ С ИСПОЛЬЗОВАНИЕМ ЦИКЛА С ПОСТУСЛОВИЕМ *do … while*

Цель работы – приобретение и закрепление практических навыков при написании циклических программ на языке С++ с использованием цикла с постусловием ***do … while***.

Цикл с постусловием *do … while*

На языке программирования С++ цикл с постусловием реализован оператором ***do …while*** (приложение Б, рис. Б.1). Цикл с постусловием похож на цикл с предусловием ***while.*** Разница состоит в том, что в цикле с постусловием проверка условия производится сразу же при входе в цикл, и, лишь затем, если условие *истинно* – выполняется тело цикла. В цикле с предусловием в любом случае сначала выполняется тело цикла и только потом идет проверка условия. Если условие *истинно*, выполнение действия продолжается, а если нет, то выполнение передается следующему за ***while*** оператору. Другими словами, в отличие от ***while*** внутри цикла с постусловием (***do … while***), тело цикла хотя бы один раз выполняется. Если синтаксис языка требует, чтобы в теле цикла присутствовал только один оператор, то операторные скобки не указываются. Если в теле цикла необходимо выполнить несколько операторов, то необходимо использовать составной оператор, т.е. заключить тело цикла в фигурные скобки *{ }*.Оператор ***do … while*** обычно используют, когда цикл требуется обязательно выполнить, хотя бы один раз (например, если в цикле производится ввод данных).

Пример.

Написать программу (используя цикл с постусловием ***do … while***), которая находит сумму всех целых чисел от 1 до 10 включительно.

```
int main ( )
{
    int i, s;
    s = 0; // переменная для накопления суммы
    i = 1; // управляющая переменная цикла
```

```
do
{
   s = s + i; // накапливание суммы
   i++; // изменение управляющей переменной
}
while (i <= 10); // проверка условия, сравнение управляющей переменной с окончанием диапазона
cout<< " s = " << s <<'\n'; // вывод результата на экран
return 0;
}
```

В начале программы объявляется переменная ***i* = 1,** а также переменная для накапливания суммы ***s* = 0**. Далее в теле цикла выполняется накапливание суммы ***s* = *s* + *i***, к предыдущему значению переменной ***s*** прибавляется текущее значение переменной ***i*** и новое значение перезаписывается в переменную ***s*.** Также в теле цикла выполняется увеличение значения переменной ***i*** на **1**. Далее происходит проверка условия выполнения тела цикла (***while*** (***i*** <= **10**)) – значение переменной ***i*** меньше или равно **10**. Если условие принимает значение *true,* то опять выполняется тело цикла, если условие – *false,* то происходит выход из цикла и выполнение оператора следующего за циклом, а именно вывод на экран значение переменной ***s*.**

ПРИМЕРЫ РЕШЕНИЯ ЗАДАНИЙ

Задание 6.1

Написать программу, которая выводит на экран количество делителей целого числа *n*, вводимого с клавиатуры.

I. *Выбор метода*

Делителем целого числа *n* является число, которое делится на *n* без остатка, т.е. остаток равен нулю. Для поиска делителей числа *n*, вводимого с клавиатуры, необходимо организовать цикл, в теле которого будет проверяться остаток от деления числа *n* на целые числа из диапазона от 1 до *n* / 2. При этом если число является делителем, т.е. остаток от деления равен нулю, то необходимо увеличить счетчик на 1.

Например: число 12 имеет 5 делителей – 1, 2, 3, 4, 6.

II. *Описание решения задачи на псевдокоде*

1. Начало.

2. Ввести значение *n*.

3. Проверить является ли число *i* делителем *n*.

4. Если *i* – делитель, то увеличить на единицу счетчик *k*, иначе – повторить пункт 3.

5. Вывести результат *y* на экран.

6. Конец.

III. *Схема алгоритма работы программы*

Блок–схема алгоритма программы показана на рисунке 1.3.

IV. *Разработка текста программы*

1. Подключаем в файле ***stdafx.h*** необходимые для работы программы библиотеки:

 # include <iostream> – для работы операторов ввода/вывода.

using namespace std;

2. Разработка раздела описания переменных

int n, i, k;

n – целое число, вводимое с клавиатуры;

i – делители числа *n*;

k – количество делителей числа *n*.

3. Разработка тела программы

Вводим с клавиатуры исходные данные – целое число *n*. Для указания пользователю что вводить используем оператор вывода ***cout***<< с соответствующим текстом. Затем используем оператор ввода ***cin***>> с указанием необходимых переменных.

cout<< " Введите целое число n \n";

cin>> n;

Для определения количества делителей необходимо обнулить переменную *k* и определить начальные установки для цикла

***k* = 0;**

***i* = 1;**

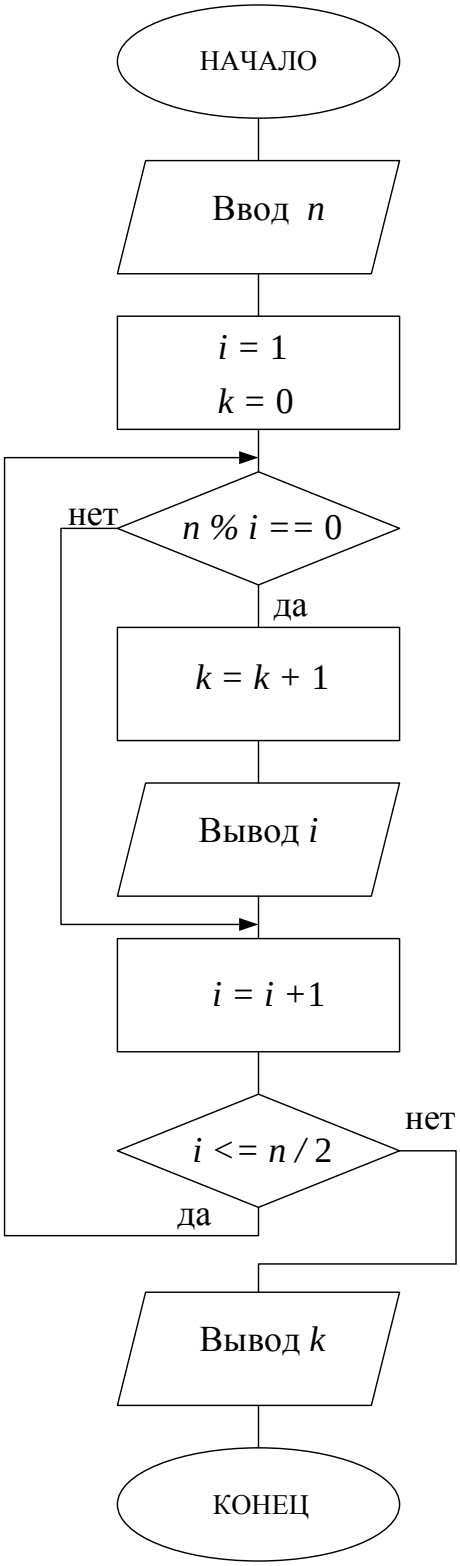

Рисунок 1.3 – Блок-схема алгоритма программы задания 6.1

Цикл с постусловием для определения делителей целого числа *n*

```
do
{
        if (n % i = = 0)
                {
                k++;
                cout<< i << '\n';
                }
        i++;
}
while (i <= n / 2);
```

Синтаксис программы

```
# include <iostream>
using  namespace  std;
int main ( )
{
     int n, i, k; // описание переменных целого типа
     cout<< "Введите целое число n \n";
     cin>>n; // ввод значения переменной n
     i = 1; // установка начального значения цикла
     k = 0;  // обнуление счетчика для определения количества делителей
     cout<< " Делители числа " << n << " : ";
     do// начало цикла с постусловием
     {
        if  (n % i = = 0)  // проверка условия равен ли  остаток от деления нулю
                {
                k++; // увеличение на единицу счетчика, если i – делитель
                cout<< i << " ,  "; //вывод делителя на экран
                }
        i++; // следующий делитель
     }
     while (i <= n / 2); // условие выхода из цикла с постусловием
```

```
    cout<< " Количество делителей числа " << n << "–" << k <<"\n"; // вывод на экран результата
    return 0;
}
```

4. Отладка и запуск программы

Для отладки программы используем клавишу *F7*, убеждаемся в отсутствии ошибок и запускаем программу на выполнение с помощью комбинации клавиш *Ctrl* + *F5*. Ниже приведены результаты работы программы.

Введите целое число n – 12

Делители числа 12: 1, 2, 3, 4, 6

Количество делителей числа 12 – 5

Порядок выполнения лабораторной работы

1. В соответствии с номером по журналу выберите индивидуальное задание.
2. Разработайте алгоритм решения задачи.
3. Составьте текст программы.
4. Создайте проект в интегрированной среде разработки *Microsoft Visual Studio* (*lab5*_фамилия.*cpp)*.
5. Введите текст программы.
6. Скомпилируйте программу. Если в программе есть ошибки, их необходимо исправить. Если ошибок нет, то появится сообщение об успешной компиляции.
7. Запустите программу на выполнение, проанализируйте результаты и убедитесь в правильности решения задачи.
8. Выполните отчет по лабораторной работе, который должен содержать:

– титульный лист;

– цель работы;

– индивидуальное задание;

– алгоритм работы программы;

– текст программы;

– результаты работы программы;

– выводы.

Индивидуальные задания

1. Составить программу нахождения суммы всех делителей целого числа *n*, вводимого с клавиатуры.

2. Составить программу нахождения произведения всех делителей целого числа *n* вводимого с клавиатуры.

3. Ввести последовательность целых чисел, пока они четные. Составить программу поиска их среднего арифметического.

4. Ввести последовательность вещественных чисел, пока не введете 0. Составить программу нахождения суммы положительных и отрицательных чисел.

5. Составить программу нахождения наибольшего общего делителя (НОД) двух натуральных чисел, введенных с клавиатуры.

6. Составить программу, определяющую, является ли заданное целое число, введенное с клавиатуры степенью числа 3.

7. Ввести последовательность целых чисел, пока не введете 0. Составить программу нахождения среднего арифметического чисел, кратных 3.

8. Составить программу поиска трехзначных целых чисел, делящихся нацело на 9. Результат вывести на экран.

9. Составить программу, определяющую, все целые числа из промежутка от 300 до 600, у которых сумма делителей кратна 10.

10. Составить программу определяющую, сколько и какие двузначные числа отвечают условию: сумма квадратов цифр кратна 13.

11. Составить программу, определяющую, количество точек с целочисленными координатами, принадлежащих кругу радиуса *R* с центром в начале координат.

12. Составить программу поиска всех трёхзначных чисел, сумма цифр которых делится на *n* без остатка.

13. Составить программу поиска количества трехзначных натуральных чисел, сумма квадратов цифр которых равна заданному целому числу *n*, введенного с клавиатуры.

14. С клавиатуры вводится целое число *n*. Составить программу поиска среднего арифметического его цифр.

15. С клавиатуры вводится целое число *n*. Определить сколько раз в нем встречается цифра *a*.

16. Известен факториал числа *n*. Составить программу поиска этого числа.

17. Найти все натуральные числа меньше 100, которые при возведении в квадрат дают палиндром.

18. Введите последовательность вещественных чисел, пока не введете 0. Найти сумму трехзначных положительных чисел.

19. Вывести на экран все делители целого числа *n*, вводимого с клавиатуры.

20. Напечатать в возрастающем порядке все числа от 100 до 999, в десятичной записи, которых нет одинаковых цифр.

Контрольные вопросы

1. Что будет на экране после выполнения следующего фрагмента кода?

```
int i = 0, a = 3;
do
{
    i++;
    a++;
}
while (i > a);
cout<<i<<a;
```

Вариатны ответов: **А.** 0, 3; **Б.** 1, 3; **В.** 2, 4; **Г.** Ошибка на этапе компиляции; **Д.** 1, 4.

2. Чему будет равно значение переменной *i* после выполнения следующего фрагмента кода?

```
int i = –2;
int  a = –3;
do
{
    a = i++;
}
while (a != 2);
cout<<"i="<<i;
```

Варианты ответов: **А.** *i* = 3; **Б.** *i* = –3; **В.** *i* = –2; **Г.** *i* = 1; **Д.** Ошибка на этапе выполнения

3. Что будет на экране после выполнения следующего фрагмента кода?

```
int i = 3;
do
{
        i ––;
        while (i == 2)
        {
                break;
                i = –1;
        }
}
while (i > 0)
cout<<” i= ”<<i;
```

Варианты ответов: **А.** $i = 1$; **Б.** $i = 2$; **В.** $i = 0$; **Г.** Ошибка на этапе компиляции; **Д.** Вечный цикл.

4. Что будет на экране после выполнения следующего фрагмента кода?

```
int i = 1, a = 2;
do
{
        i++; a++;
}
while (i > a);
cout<<”i=”<<i<<”a=”<<a;
```

Варианты ответов: **А.** $i = 3$, $a = 2$; **Б.** $i = 2$, $a = 1$; **В.** $i = 2$, $a = 3$; **Г.** Вечный цикл; **Д.** Ошибка на этапе выполнения.

5. Что будет на экране после выполнения следующего фрагмента кода?

```
int i = 2, a;
do
{
        a = i % 3; i++;
        cout<<a;
}
while (i != a);
```

Варианты ответов: **А.** $a = 2$; **Б.** $a = 0$; **В.** $a = 1$; **Г.** Вечный цикл.

ЛАБОРАТОРНАЯ РАБОТА 7

РАЗРАБОТКА ПРОГРАММ НА ЯЗЫКЕ С++ С ИСПОЛЬЗОВАНИЕМ ЦИКЛА С ПАРАМЕТРАМИ *for*

Цель работы – приобретение и закрепление практических навыков при написании циклических программ на языке С++ с использованием цикла с параметрами ***for.***

Параметрический цикл *for*

Цикл с параметрами (параметрический цикл ***for)*** позволяет удобно и наглядно описывать необходимые параметры цикла в случае, когда заранее известно начальное и конечное значение параметра цикла, а также известно как изменяется параметр цикла. Данный оператор теоретически является полной аналогией цикла ***while***, а практически позволяет организовать цикл с более удобным управлением. Оператор ***for*** используется для организации циклов со счетчиками, т.е. с целочисленными переменными, которые изменяют свое значение при каждом проходе цикла регулярным образом (например, увеличиваются на единицу). Блок-схема алгоритма и синтаксис параметрического цикла ***for*** представлен на рисунке В.1 (приложение В). Синтаксис параметрического цикла ***for*** на языке программирования С++:

```
for (инициализация_переменной; проверка_условия; изменение_переменной)
    {
        тело цикла;
    }
```

Основные параметры и элементы цикла

- Начальные установки (*инициализация_переменной*) – задание начального значения, которое принимает переменная – параметр цикла перед первым выполнением циклического действия. При этом задание и инициализация цикла всегда выполняется только один раз.
- Тело цикла (*тело цикла*) – операторы, которые выполняются в цикле.

• Модификация параметра цикла (*изменение_переменной*) – выражение, которое определяет, как изменяется переменная параметра цикла.

• Проверка условия (*проверка_условия*) – конечное значение, которое определяет условие выхода из цикла. Цикл заканчивается, когда переменная параметра цикла достигает конечного значения.

Принцип работы цикла с параметрами for

1. Инициализация начального значения переменной.
2. Проверка условия.
3. Выполнение тела цикла, если условие истинно.
4. Если условие ложно, выполнение следующего за циклом оператора.
5. Если условие было истинно – изменение переменной параметра цикла.
6. Проверка условия. Далее снова выполняем пп. 3 и 4.

Пример.

Написать программу (используя цикл с параметрами ***for***), которая находит сумму всех целых чисел от 1 до 10 включительно.

```
int main ( )
{
    int  i, s;
    s = 0; // переменная для накопления суммы
    for (i = 1; i <= 10; i++) // параметры цикла
       s = s + i; // накапливание суммы
    cout<< ”s = ” << s << ’\n’; // вывод результата на экран
    return 0;
}
```

Рассмотрим пример использования цикла ***for*** в программе, которая выводит на экран числа от 1 до 5.

```
int main ( )
{
    for (int i = 1; i <= 5; i++)
       cout<< i;
    return 0;
}
```

В параметрах цикла ***for*** инициализируется переменная ***i,*** равная **1**. Затем осуществляется проверка значения этой переменной с помощью условия ***i*** **<= 5**. Если условие истинно (так будет пока ***i*** не достигнет значения 6) выполняется тело цикла, а именно вывод на экран значения ***i*** ***(cout<< i)*** и изменение значения параметра цикла на **1** (***i*++**). Затем снова проверяется условие. Если условие ложно (т.е. значение переменной параметра цикла стало равно 6), то программа переходит на следующую строчку за телом цикла.

Инициализация управляющей переменной

1. Инициализация и создание переменной осуществляется в цикле

```
for (int x = 1; x <= 100; x++)
    cout<< x;
```

2. Создание переменной производится до цикла, а инициализация в теле цикла

```
int x
for (x = 1; x <= 100; x++)
    cout<< x;
```

3. Инициализация и создание переменной производятся до цикла

```
int x = 1;
for (; x <= 100; x++)
    cout<< x;
```

Изменение управляющей переменной

Изменение управляющей переменной можно перенести внутрь тела цикла, как это происходит в ***while*** и ***do…while.***

```
for (int x = 1; x <= 100;)
    {
        cout<< x;
        x++;
    }
```

Условие

Условие конструкции также можно пропустить, однако в этом случае оно будет считаться по умолчанию истинным. Таким образом, мы получаем постоянно истинное условие и как следствие ВЕЧНЫЙ ЦИКЛ.

```
for (int x = 1;  ; x++)
        cout<< x;
```

Вложенные циклы

Принцип работы программы, реализующей вложенный цикл, основан на том, что внутренний цикл полностью выполняется на каждом шаге внешнего цикла от начала и до конца. Другими словами, пока программа не выйдет из вложенного цикла – выполнение внешнего не продолжится.

Пример.

Написать программу, которая выводит на экран таблицу умножения.

```
# include <iostream>
using  namespace  std;
int main ( )
{
   int i, j, p;
   for (i = 1; i < 10; i++)
   {
       for (j = 1; j < 10; j++)
       {
            p = i * j;
            cout<< p << ”\t”;
       }
   cout<<’\n’;
   }
   return 0;
}
```

Блок-схема к данной программе показана на рисунке 1.4.

Комментарий к программе.

1. Управляющие переменные внешнего и внутреннего циклов (***i, j***) осуществляют функции множителей.

2. Управляющая переменная ***i*** создается и инициализируется значением 1.

3. Программа проверяет условие ***i* < 10**, так как 1 меньше 10, условие является истинным и программа входит во внешний цикл.

4. Управляющая переменная ***j*** создается и инициализируется значением 1.

5. Программа проверяет условие ***j* < 10**, так как 1 меньше 10, условие является истинным и программа входит во внутренний цикл.

6. Осуществляется показ на экран произведения ***i*** на ***j* – 1.**

7. Осуществляется изменение управляющей переменной ***j*.**

8. Снова проверяется условие ***j* < 10**, так как 2 меньше 10, условие является истинным и программа снова входит во внутренний цикл.

9. Осуществляется показ на экран произведения ***i*** на ***j* – 2**.

10. Осуществляется изменение управляющей переменной ***j***.

Действия с 5 по 7 пункты повторяются до тех пор, пока ***j*** не становится равно 10, при этом текущее значение ***i* = 1** умножается на каждое значение ***j*** (от 1 до 9 включительно), результат показывается на экран. Получается строка таблицы умножения на 1. Затем программа выходит из внутреннего цикла и переводит экранный курсор на две строки вниз. После этого осуществляется увеличение переменной ***i*** на единицу и снова вход во внутренний цикл. Теперь уже для вывода цепочки умножения на 2. Таким образом, на экране появляется таблица умножения.

1	2	3	4	5	6	7	8	9
2	4	6	8	10	12	14	16	18
3	6	9	12	15	18	21	24	27
4	8	12	16	20	24	28	32	36
5	10	15	20	25	30	35	40	45
6	12	18	24	30	36	42	48	54
7	14	21	28	35	42	49	56	63
8	16	24	32	40	48	56	64	72
9	18	27	36	45	54	63	72	81

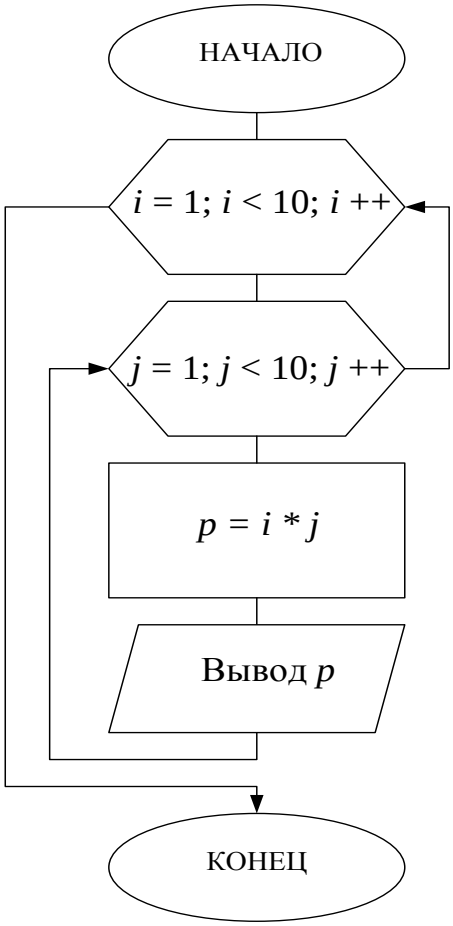

Рисунок 1.4 – Блок-схема алгоритма программы

ПРИМЕРЫ РЕШЕНИЯ ЗАДАНИЙ

Задание 7.1

Вычислить сумму ряда $y = x^{10} + 2 \cdot x^9 + 3 \cdot x^8 + \ldots + 10 \cdot x + 11$. Округлить полученное значение y к ближайшему целому числу. Определить, является ли полученное число:

1) простым числом;

2) совершенным числом, т.е. равным сумме всех своих положительных делителей, кроме самого этого числа. (Например, число 6 является совершенным, 6 = 1 + 2 + 3, где 1, 2 и 3 – делители числа 6).

I. *Выбор метода*

Сумму ряда $y = x^{10} + 2 \cdot x^9 + 3 \cdot x^8 + \ldots + 10 \cdot x + 11$ можно вычислить с помощью последовательного накопления или по схеме Горнера, используя равенство

$$y = x^{10} + 2 \cdot x^{9} + 3 \cdot x^{8} + \ldots + 10 \cdot x + 11 =$$
$$= 11 + x(10 + x(9 + x(8 + x(7 + x(6 + x(5 + x(4 + x(3 + x(2 + x)))))))))$$

Для округления значения переменной *y*, необходимо воспользоваться оператором приведения типа ***int k*** = (***int***) ***y***.Для определения является ли число *z* простым, необходимо организовать цикл с условием нахождения делителей числа *z* в интервале от 2 до *z* – 1. Простое число в указанном интервале делителей не имеет. Для определения является ли число совершенным, необходимо организовать цикл с поиском делителей этого числа, а также накопление сумму всех положительных делителей.

II. *Описание решения задачи на псевдокоде*

1. Начало.
2. Ввести значение *x*.
3. Вычислить *y*.
4. Определить является ли округленное число *y* простым.
5. Определить совершенное ли округленное число *y*.
6. Конец.

III. *Схема алгоритма работы программы*

Блок-схема алгоритма программы представлена на рисунке 1.5.

IV. *Разработка текста программы*

1. Подключаем в файле ***stdafx.h*** необходимые для работы программы библиотеки:

 # include <iostream> – для работы операторов ввода/вывода;

using namespace std;

2. Разработка раздела описания переменных

float y, x;

int s, i, k;

bool r;

x – вещественное число, вводимое с клавиатуры;

y – вещественное значение функции;

k – округленное значение переменной *y;*

i – делители числа *k*;

s – сумма делителей числа *k*;

r – логическое значение флага.

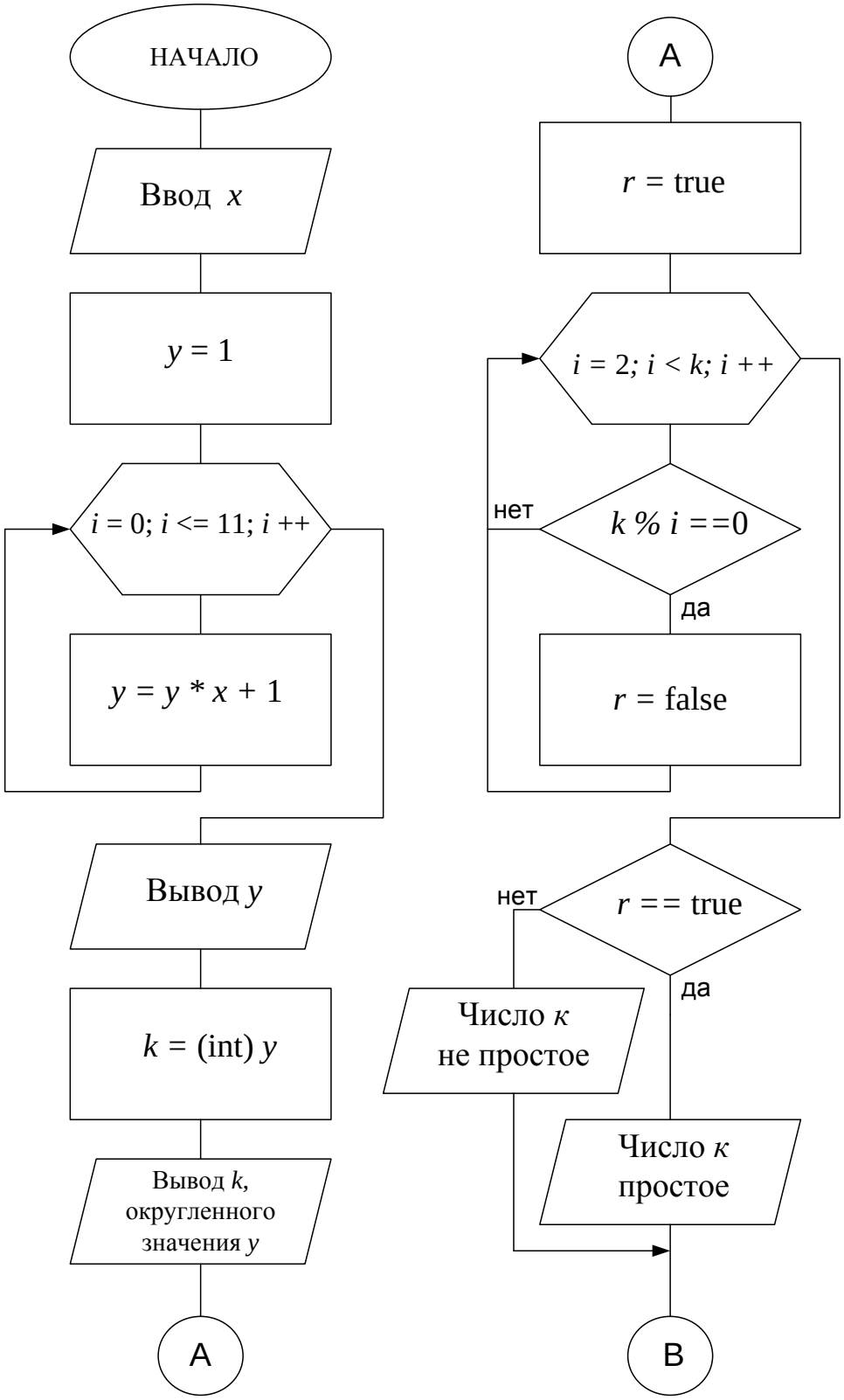

Рисунок 1.5 – Блок-схема алгоритма программы задания 7.1 (начало)

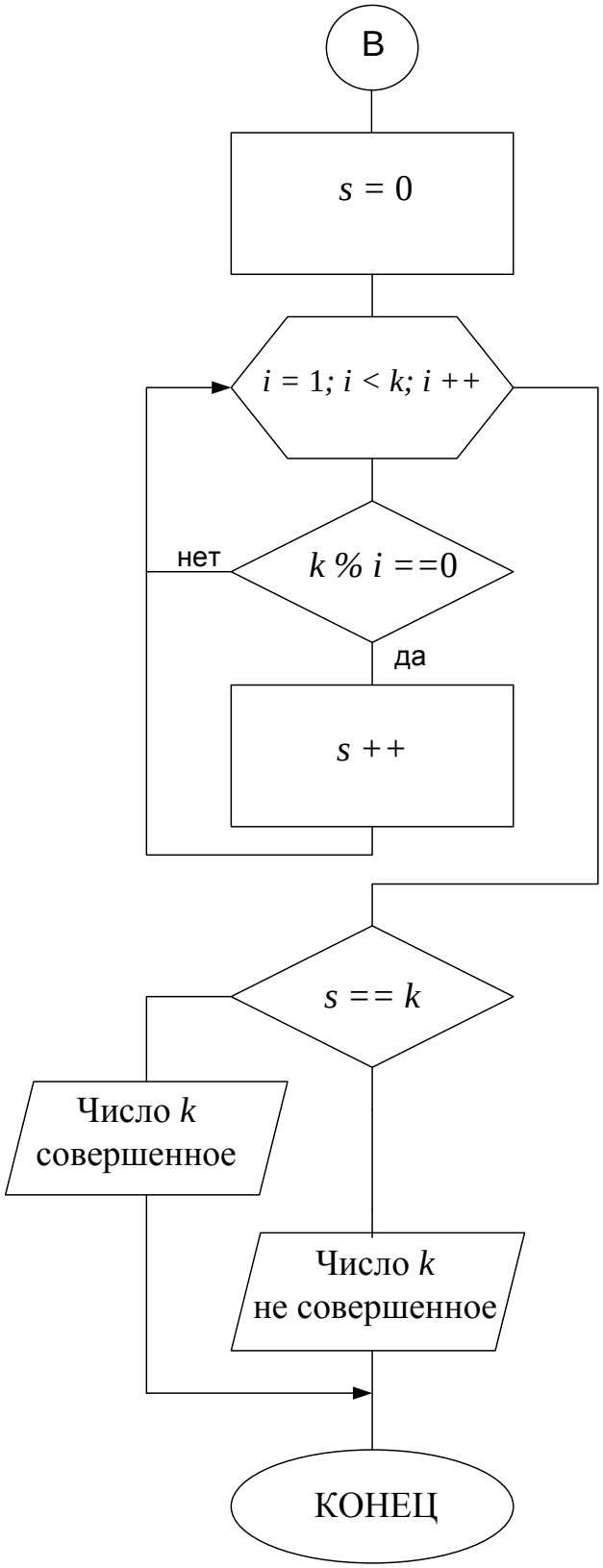

Рисунок 1.5 – Блок-схема алгоритма программы задания 7.1 (продолжение)

3. Разработка тела программы

Вводим с клавиатуры исходные данные – вещественное число *x*. Для указания пользователю что вводить, используем оператор вывода ***cout***<< с соответствующим текстом. Затем используем оператор ввода ***cin***>> с указанием необходимых переменных.

```
cout<< " Введите вещественное число x \n";
cin>> x;
```

Для вычисления суммы ряда $y = x^{10} + 2 \cdot x^{9} + 3 \cdot x^{8} + \ldots + 10 \cdot x + 11$ необходимо организовать параметрический цикл, в теле цикла которого будет последовательное накопление суммы

```
y = 1;
f = 0;
for (i = 0; i <= 11; i++)
        y = y*x + i;
```

Для округления значения *y* до целого числа воспользуемся оператором приведения типа:

```
int k = (int) y;
```

Для того чтобы определить является ли округленное число *y* (целое число *k*) простым числом, необходимо установить флаг – значение логической переменной *r*, равное *true*, и организовать цикл с параметрами для поиска делителей числа *k* в диапазоне от 2 до $k - 1$.

```
r = true;
for (i = 2; i < k; i++)
        if (k % i = = 0)
                r = false;
if (r = = true)
        cout<< " Число простое ";
else
        cout<< " Число не простое ";
```

Для определения является ли округленное число *y* (в дальнейшем это целое число *k*) совершенным числом, с помощью цикла с параметрами организуем поиск делителей числа *k* с накоплением суммы этих делителей, затем сравниваем полученную сумму со значением переменной *k*.

```
s = 0;
for (i = 1; i < k; i++)
```

```
        if (k % i = = 0)
                s++;
if (s = = k)
        cout<< “Число совершенное”;
else
        cout<< “Число не совершенное”;
```

Синтаксис программы

```
# include <iostream>
using  namespace  std;

int main ( )
{
    float y, x;
    int s, i, k;
    bool r;
    // ввод с клавиатуры значения переменной x
    cout<< ” Введите значение переменной x \n”;
    cin>> x;
    //цикл вычисления значения выражения y
    y= 1; // установка начального значения
    for (i = 0; i <= 11; i++)
       y = y*x + i; // накопление суммы
    cout<< “ y = “<<y<<’\n’;
    k = (int) y; // округление переменной y к ближайшему целому числу
    cout<< ” Округленное значениеy = ”<< k << ’\n’;
    //определение, является ли число k простым
    r = true; // установка начального значения
    for (i = 2; i < k; i++) // цикл для нахождения делителей числа k
       if (k % i = =0) //проверка условия ,является ли i делителем числа k
            r = false;
    if (r = = true) // проверка, переменная r изменило своё значение
       cout<< “ Число “<< k << ” простое ” <<’\n’;
    else
```

```
    cout<< “Число“ << k << ”не простое” <<’\n’;
  // определение, является ли число k совершенным
  s = 0; // установка начального значения
  for (i = 1; i < k; i++) // цикл для нахождения делителей числа k
    if (k % i = = 0) // проверка, является ли i делителем числа k
          s++; // накопление суммы делителей
  if (s = = k) // проверка, равна ли сумма делителей числу k
    cout<< “ Число ” << k << ” совершенное ” << ’\n’;
  else
    cout<< “ Число “ << k << ” не совершенное ” <<’\n’;
  return 0;
}
```

4. Отладка и запуск программы

Для отладки программы используем клавишу *F7*, убеждаемся в отсутствии ошибок и запускаем программу на исполнение с помощью комбинации клавиш *Ctrl+F5*. Ниже приведены результаты работы программы.

Введите значение переменной x

0.3

y = 15.1020419

Округленное значение y = 15

Число 15 *не простое*

Число 15 *не совершенное*

Порядок выполнения лабораторной работы

1. В соответствии с номером по журналу выберите индивидуальное задание.
2. Разработайте алгоритм решения задания.
3. Составьте текст программы.
4. Создайте проект в интегрированной среде разработки *Microsoft Visual Studio* (*lab5*_фамилия.*cpp)*.
5. Введите текст программы.
6. Скомпилируйте программу. Если в программе есть ошибки, их необходимо исправить. Если ошибок нет, то появится сообщение об успешной компиляции.

7. Запустите программу на выполнение, проанализируйте результаты и убедитесь в правильности решения задачи.

8. Выполните отчет по лабораторной работе, который должен содержать:

− титульный лист;

− цель работы;

− индивидуальное задание;

− алгоритм работы программы;

− текст программы;

− результаты работы программы;

− выводы.

Индивидуальные задания

1. Написать программу, которая вычисляет функцию $y(x) = 10\,x!$. Результат вывести на экран.

2. Вычислить наибольший общий делитель двух целых чисел, введенных с клавиатуры. Результат вывести на экран.

3. Подсчитать количество цифр в десятичной записи целого неотрицательного числа n, введенного с клавиатуры.

4. Составить программу поиска первых n четных чисел (значение переменой n вводится с клавиатуры) в последовательности чисел Фибоначчи.

5. Составить программу поиска всех простых чисел в промежутке $[n1; n2]$, где $n1$ и $n2$ – два целых числа, введенных с клавиатуры.

6. Составить программу, которая определяет, является ли целое число n, введенное с клавиатуры, – простым.

7. Составить программу, которая выводит первые n чисел Фибоначчи.

8. Написать программу, которая вычисляет сумму ряда $y = \cos x^2 + \cos x^3 + \cos x^4 + \ldots + \cos x^{20}$. Результат вывести на экран.

9. Написать программу, которая вычисляет сумму ряда $y = 1! + 2! + 3! + \ldots + n!$, $(n > 1)$. Результат вывести на экран.

10. Вычислить сумму квадратов всех целых чисел, попадающих в интервал $(\ln x, e^x)$, где $x > 1$ – вещественное число, введенное с клавиатуры.

11. С клавиатуры вводится число Фибоначчи. Составить программу, определяющую его порядковый номер в последовательности Фибоначчи

12. Вычислить количество точек с целочисленными координатами, попадающих в круг радиусом R ($R > 0$) с центром в начале координат.

13. Напечатать в возрастающем порядке все числа от 100 до 999 в десятичной записи, которых нет одинаковых цифр.

14. Определить количество трехзначных целых чисел, сумма цифр которых равна n. Результат вывести на экран.

15. Определить является ли заданное натуральное число палиндромом, т.е. таким, которое читается одинаково слева направо и справа налево.

16. Определить число, полученное выписыванием в обратном порядке цифр заданного натурального числа.

17. Написать программу поиска всех натуральных чисел меньше 100, которые при возведении в квадрат дают палиндром.

18. Написать программу поиска максимального натурального числа из интервала от a до b с максимальной суммой делителей.

19. Составить программу всех дружественных чисел в интервале от 1 до 350 (числа являются дружественными, если каждое из них равно сумме делителей другого).

20. Если к сумме цифр двузначного числа прибавить квадрат разности цифр, то получится то же самое число. Составить программу поиска всех таких чисел.

Контрольные вопросы

1. Что будет на экране в результате выполнения следующего фрагмента кода?

```
int   k = 20;
for (k = 3; k < 20; k++)
{
    k++;
    cout<<" 1 ";
}
```

Варианты ответов: **А**. 20 единиц; **Б**. 9 единиц; **В**. 17 единиц; **Г**. Ошибка на этапе компиляции; **Д**. Ошибка на этапе выполнения.

2. Что будет на экране в результате выполнения следующего фрагмента кода?

```
int k;
for (k = 0; k < 10; k++)
    k += k;
cout<<k;
```

Варианты ответов: **А**. 10; **Б**. 15; **В**. 20; **Г**. Вечный цикл; **Д**. Ошибка на этапе компиляция.

3. Что будет на экране в результате выполнения следующего фрагмента кода?

```
for (int i = 0; i < 10; i++)
    for (int j = 0; j < 10; j++)
        if (i == 0)
    {
            cout<<"$$";
        else
            cout<<"&&";
    }
```

Варианты ответов: **А**. $$&&; **Б**. &&$$; **В**. $&$&; **Г**. &$&$; **Д.** Ошибка на этапе компиляция.

4. Что будет на экране в результате выполнения следующего фрагмента кода?

```
for (int i = 0; i < 2; i++)
    for (int j; j < 3; j++)
        cout<<1;
```

Варианты ответов: **А.** 11111, **Б.** 111111; **В.** Вечный цикл; **Г.** 1; **Д.** Ошибка на этапе компиляция.

5. Что будет на экране в результате выполнения следующего фрагмента кода?

```
for (int i = 0; i < 3; i++)
    for (int j = 2; j <= 10; j––)
        cout<<j;
```

Варианты ответов: **А.** На экран ничего не выводиться; **Б.** Вечный цикл; **В.** Ошибка на этапе компиляция; **Г.** 2; **Д.** 21.

ЛАБОРАТОРНАЯ РАБОТА 8

ОДНОМЕРНЫЕ МАССИВЫ НА ЯЗЫКЕ ПРОГРАММИРОВАНИЯ С++

Цель работы – приобретение и закрепление практических навыков при написании программ с использованием одномерных массивов на языке программирования С++.

Массивы переменных

Массив – это пронумерованная последовательность величин одинакового типа, обозначаемая одним именем. Массив данных в программе рассматривается как переменная структурированного типа. Массиву присваивается *имя_массива*, посредством которого можно ссылаться как на массив данных в целом, так и на любую из его компонент (рисунок 1.6). При этом каждая переменная в массиве является самостоятельной единицей под названием – *элемент_массива*. Каждый элемент имеет свой порядковый номер – *индекс*. По *индексу* можно обращаться к конкретному элементу массива. *Индекс* в обозначении *элемента массива* может быть константой, переменной или выражением порядкового типа (целочисленный, перечисляемый диапазон). Количество *элементов* в массиве определяет *размерность* массива. По этому признаку массивы делятся на одномерные (линейные), двумерные, трёхмерные и т.д.

Для того чтобы использовать массив в программе, необходимо его объявить с указанием типа данных для элементов массива. Массив может иметь как стандартный тип данных: *int, float, char, bool,* так и тип данных созданный пользователем. *Индекс* первого элемента массива равен нулю, таким образом, *индекс* последнего элемента всегда на единицу меньше размерности массива. Автоматический контроль выхода индекса за границы массива не производится, поэтому необходимо следить за этим. Для того чтобы обратиться к элементу массива необходимо указать *имя_массива* и в квадратных скобках *индекс* элемента массива.

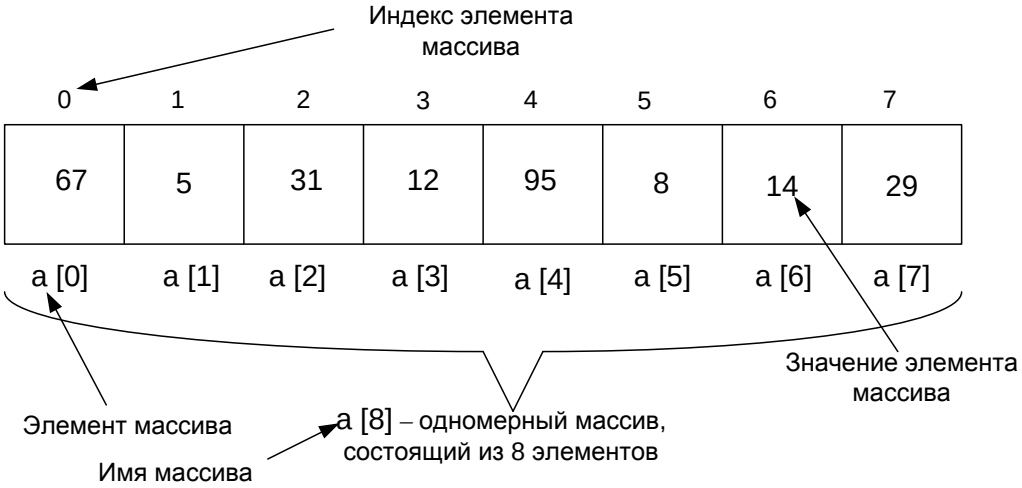

Рисунок 1.6 – Схема одномерного массива из восьми элементов

Описание одномерного массива

Тип_массива имя_массива [*размерность_массива*];

- *Тип_массива* – тип каждого элемента массива.
- *Имя_массива* – любое имя массива, которое подчиняется правилу идентификатора.
- *Размер_массива* – количество элементов в массиве. В квадратных скобках указывается количество элементов массива – *целочисленное константное значение* (целочисленный литерал, либо константная целочисленная переменная).

Пример

***int a* [10]** – массив с названием <*a*> из 10 элементов целочисленного типа. Для данного массива элементы имеют индексы от 0 до 9.

Инициализация массива

Инициализация массива – задание (присвоение) начальных значений элементам массива.

1. Инициализация элементов массива при описании массива.

тип_массива имя_массива [*размер*] = {иниц. 1, иниц. 2, …};

Пример.

int a* [5] = {4, 9, 0, 5, 2}*;

a [0] ← 4, *a* [1] ← 9, *a* [2] ← 0, *a* [3] ← 5, *a* [4] ← 2.

Количество элементов массива соответствует параметру «размер». Далее идет список инициаторов (значение, которое сразу при описании помещается в ячейку памяти, соответствующую данной переменной), «иниц.1» будет записан в ячейке памяти, где хранится 0-й элемент массива и т.д. Количество инициаторов – это размер массива. Если инициализирующих значений меньше, чем элементов в массиве, остаток массива обнуляется, если больше – лишние значения не используются.

2. Инициализация элементов массива с помощью оператора присваивания

имя_массива [*индекс*] = *значение*;

Пример.

int a* [5]*;

a* [0] = 12*; // *присвоение значения* 12-му *элементу массива с индексом* 0

a* [1] = 0*; *//присвоение значения* 0-му *элементу массива с индексом* 1

a* [2] = 36*; *// присвоение значения* 36-му *элементу массива с индексом* 2

a* [3] = 59*; *// присвоение значения* 59-му *элементу массива с индексом* 3

a* [4] = 7*; *// присвоение значения* 7-му *элементу массива с индексом* 4

***cout*<< *a* [2];** *// вывод на экран значения элемента с индексом* 2

3. Инициализация элементов массива с помощью ввода значений с клавиатуры.

cin>>имя_массива [индекс];

Пример.

***int a* [5];**

***cin*>> *a* [0];** *// ввод с клавиатуры значения элемента массива с индексом* 0

***cin*>> *a* [1];** *// ввод с клавиатуры значения элемента массива с индексом* 1

***cin*>> *a* [2];** *// ввод с клавиатуры значения элемента массива с индексом* 2

***cin*>> *a* [3];** *// ввод с клавиатуры значения элемента массива с индексом* 3

***cin*>> *a* [4];** *// ввод с клавиатуры значения элемента массива с индексом* 4

Ввод одномерного массива с клавиатуры с помощью цикла ***for***

***int a* [10];**

for (i* = 0*; i* < 10*; i*++*)

***cin*>> *a* [*i*];**

В данном цикле переменная ***i*** является индексом элемента массива ***a***. Значение, принимаемое переменной ***i*** при первом прохождении цикла, равно 0, так как индекс первого элемента 0, выход из цикла осуществляется при достижении переменной ***i*** значения 9, так как последний индекс элемента массива равен 9.

4. Инициализация элементов массива с помощью генератора случайных чисел.

Для использования в программе функции генератора случайных чисел необходимо подключить специальную библиотеку

 # include <stdlib.h>

Функция, которая генерирует последовательность псевдослучайных целых чисел в диапазоне от 0 до RAND_MAX. Значение RAND_MAX равно 32 767.

int rand ( );

Пример.

int x;

***x = rand* ( );**

Переменной ***x*** целого типа будет присвоено любое целое число из диапазона от 0 до 32 767.

Задание значений элементам одномерного массива с помощью генератора случайных чисел:

***int a* [10];**

for (i* = 0*; i* < 10*; i*++*)

a* [*i*] = *rand* ( ) % 250*;

Элементам одномерного массива *a* будут заданы значения целых чисел из диапазона от 0 до 250 случайным образом.

Для того чтобы повысить универсальность программы, размер массива лучше указывать через константу.

Пример.

```
# include <iostream>
using namespace std;
const int n = 5; // объявление константы
int main ( )
{
        int a [n], i; // размер массива задан через константу
        for (i = 0; i < n; i++) // цикл для перебора индекса массива
                a [i] = –30 + rand (     ) % 131;     //     задание     значений
элементам массива из диапазона [–30; 100]
}
```

Если точное количество элементов в исходном массиве не задано, но известно, что оно не может превышать некое конкретное значение, в этом случае память под массив выделяется «по максимуму». В программе же можно задать через клавиатуру количество элементов массива, и тогда заполняется только необходимая часть памяти. Несмотря на то, что значение константы ***n*** определяется «с запасом», надо обязательно проверять, не вводит ли пользователь большее количество элементов, чем возможно.

Пример.

```
using namespace std;
const int n = 1000; // объявление константы
int main ( )
{
        int a [n], i, kol; // размер массива задан по максимуму
        cout<< " Введите количество элементов массива \n ";
        cin>> kol;
        if (kol < n) // проверка на превышение размера массива
           for (i = 0; i < kol; i ++)
                cin>>a [i];// ввод значений элементов массива
        else
                cout<< " Превышение размеров массива \n ";
}
```

Приведем пример полной программы, которая создает, заполняет и выводит на экран значения элементов массива.

```
using namespace std;
const int n = 5; // объявление константы
int main ( )
{
        int a [n], i;// размер массива задан через константу
        for (i = 0; i < n; i ++) // цикл, перебирающий индекс массива
        {
                cout<< " Введите " << i << " элемент массива ";
                cin>> a [i]; // ввод с клавиатуры значений элементов
массива
        }
        cout<< '\n'; // перевод строки
        for (i = 0; i < n; i ++) // цикл, перебирающий индекс массива
                cout<< a [i] << '\n';// вывод на экран значения i-го
элемента массива
        return 0;
}
```

ПРИМЕРЫ РЕШЕНИЯ ЗАДАНИЙ

Задание 8.1

Написать программу, которая находит сумму всех отрицательных элементов одномерного массива из 10 целых чисел, вводимых с клавиатуры. Результат вывести на экран.

I. *Выбор метода*

Ввод значений элементов массива с клавиатуры осуществляется с помощью параметрического цикла с перебором индекса массива. Для нахождения суммы отрицательных элементов массива необходимо организовать цикл, в котором осуществляется проверка условия: является ли элемент отрицательным числом и, если да, то накапливать сумму.

II. *Описание решения задачи на псевдокоде*

1. Начало.
2. Ввод с клавиатуры значений элементов массива.

3. Задать начальные значения для индекса элемента массива.

4. Увеличить индекс массива на единицу.

5. Проверить, является ли элемент массива отрицательным числом, если да, то выполнить пункт 6, если нет – возврат к пункту 4.

6. К текущему значению суммы прибавить значение элемента массива и перезаписать сумму.

7. Если индекс массива больше количества элементов массива – перейти к пункту 8, иначе – возврат к пункту 4.

8. Вывод на экран результата суммы.

III. *Схема алгоритма работы программы*

Блок-схема алгоритма программы представлена на рисунке 1.7.

IV. *Разработка текста программы*

1. Подключаем в файле ***stdafx.h*** необходимые для работы программы библиотеки:

 # include <iostream> – для работы операторов ввода/вывода;

using namespace std;

const int n = 10; *// объявление константы*

2. Разработка раздела описания переменных

int a [n], s, i;

a – имя одномерного массива из целых чисел;

n – целочисленная константа, равная 10;

s – целочисленная переменная для накопления суммы;

i – целочисленная переменная для индекса элементов массива.

3. Разработка тела программы

Для ввода с клавиатуры одномерного массива необходимо организовать цикл ***for*** с переменной параметром – индекс элемента массива. В цикле используем оператор вывода ***cout<<*** с соответствующим текстом, а затем оператор ввода ***cin>>***.

```
for (i = 0; i < n; i ++) // цикл, перебирающий индекс массива
{
        cout<< " Введите " << i << " элемент массива : ";
        cin>> a [i]; // ввод с клавиатуры значений элементов массива
}
```

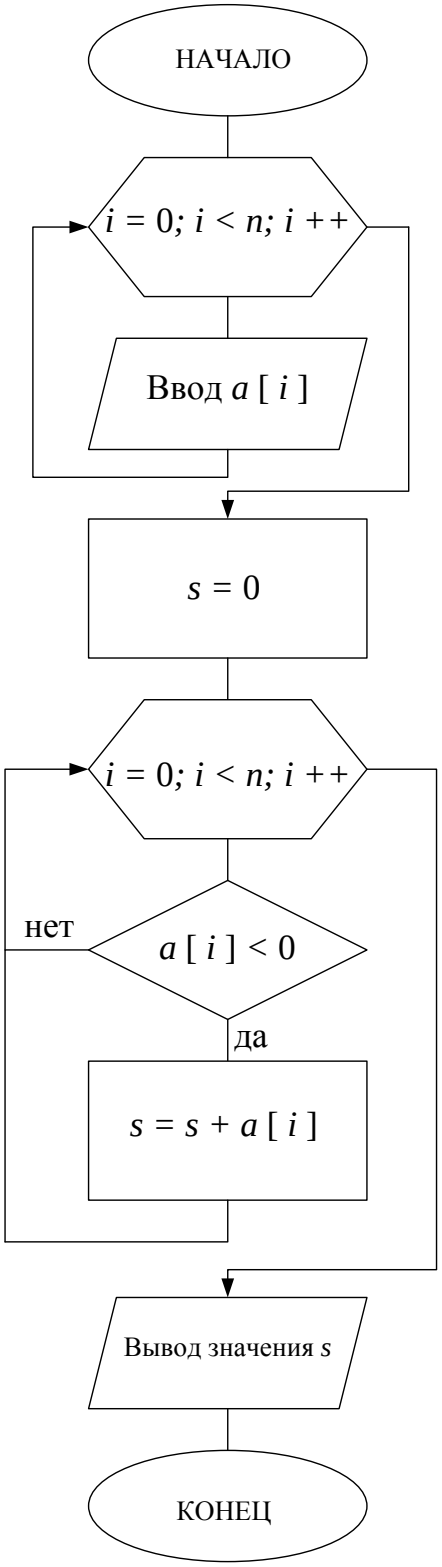

Рисунок 1.7 – Блок-схема алгоритма программы задания 8.1

Для нахождения суммы отрицательных элементов воспользуемся параметрическим циклом ***for.*** Параметром цикла является индекс массива, который перебирается от 0 до ***n.*** При этом ***n*** не входит в проверяемый диапазон, так как индекс последнего элемента ***n* – 1.** В теле цикла происходит проверка на отрицательное значение элемента массива и если значение меньше 0, то оно прибавляется к сумме.

```
s = 0; // переменная для накопления суммы
for (i = 0; i < n; i ++)// цикл, перебирающий индекс массива
{
        if (a [i] < 0)
                s = s + a [i];
}
```

Вывод на экран результата с помощью оператора вывода ***cout***<<.

```
cout<< " s= " << s << '\n';
```

Синтаксис программы

```
# include <iostream>
using   namespace  std;
const int n = 10;
int main ( )
{
        int a [n], s, i;
        s = 0; // переменная для накопления суммы
        for (i = 0; i < n; i ++) // цикл, перебирающий индекс массива
        {
                cout<< " Введите" << i << "элемент массива : ";
                cin>>a [i]; // ввод с клавиатуры значений элементов
массива
        }
        cout<< '\n'; // перевод строки
        for (i = 0; i < n; i ++) // цикл, перебирающий индекс массива
        {
                if (a [i] < 0) // если значение элемента отрицательное
                        s = s + a [i]; // добавить его значение к общей сумме
        }
```

```
        cout<< " s = " << s << '\n'; // вывод значения суммы на экран
        return 0;
}
```

4. Отладка и запуск программы

Для отладки программы используем клавишу F7, убеждаемся в отсутствии ошибок и запускаем программу на выполнение с помощью комбинации клавиш Ctrl + F5. Ниже приведены результаты работы программы.

Введите 0 *элемент массив:* 5
Введите 1 *элемент массива:* –4
Введите 2 *элемент массива:* 56
Введите 3 *элемент массива:* –3
Введите 4 *элемент массива:* –24
Введите 5 *элемент массива:* 12
Введите 6 *элемент массива:* 83
Введите 7 *элемент массива:* –8
Введите 8 *элемент массива:* 55
Введите 9 *элемент массива:* –9
s = –48

Задание 8.2

Написать программу, которая для целочисленного массива из 10 элементов, вводимого с клавиатуры, определяет количество четных элементов, расположенных между максимальным и минимальным элементами.

I. *Выбор метода*

В начале программы необходимо определить, где в массиве расположены максимальный и минимальный элементы, а точнее найти их местоположение (индексы). Далее необходимо перебрать все элементы, расположенные между ними и, если значение четное, увеличить счетчик элементов на единицу.

Например, рассмотрим одномерный массив из 10 целых чисел.

0	1	2	3	4	5	6	7	8	9
5	23	3	11	8	14	9	2	17	10
	макс						мин		

Максимальный элемент массива имеет значение 23 и местоположение его, т.е. индекс, равен 1, минимальный элемент – 2 и его местоположение, т.е. индекс, равен 7. Между максимальным и минимальным элементами расположены следующие числа – 3, 11, 8, 14, 9. Среди них четными числами являются: 8, 14. Таким образом, количество четных элементов, расположенных между максимальным и минимальным числами, равно 2. Очевидно, что порядок расположения элементов в массиве заранее неизвестен, и сначала может следовать как максимальный, так и минимальный элемент, кроме того, они могут совпадать. Поэтому, прежде чем просматривать массив в поисках количества четных элементов, требуется определить, какой из этих индексов больше.

II. *Описание решения задачи на псевдокоде*

1. Начало.
2. Ввод с клавиатуры значений элементов массива.
3. Определить местоположение максимального элемента.
4. Определить местоположение минимального элемента.
5. Определить границы просмотра массива.
6. Обнулить счетчик.
7. Просмотреть массив в указанном диапазоне и, если очередной элемент является четным числом, увеличить счетчик на единицу.
8. Вывод результата на экран.

III. *Схема алгоритма работы программы*

Блок-схема алгоритма программы представлена на рисунке 1.8.

IV. *Разработка текста программы*

1. Подключаем в файле ***stdafx.h*** необходимые для работы программы библиотеки:

 # include <iostream> – для работы операторов ввода/вывода;

using namespace std;

const int n = 10; *// объявление константы*

2. Разработка раздела описания переменных

int a [n], k, i, min, max, min_i, max_i;

a – одномерный массив из 10 целых чисел;

n – целочисленная константа, равная 10;

i – целочисленная переменная – индекс элемента массива;

k – целочисленная переменная – количество четных элементов;

min – целочисленная переменная – значение минимального элемента;

max – целочисленная переменная – значение максимального элемента;

min_i – целочисленная переменная – индекс минимального элемента;

max_i – целочисленная переменная – индекс максимального элемента.

3. Разработка тела программы

Для ввода с клавиатуры одномерного массива необходимо организовать цикл ***for*** с переменной параметром – индекс элемента массива. В цикле используем оператор вывода ***cout***<< с соответствующим текстом, затем оператор ввода ***cin***>>.

```
for (i = 0; i < n; i ++) // цикл, перебирающий индекс массива
{
        cout<< " Введите " << i << " элемент массива \n ";
        cin>> a [i]; //ввод с клавиатуры значений элементов массива
}
```

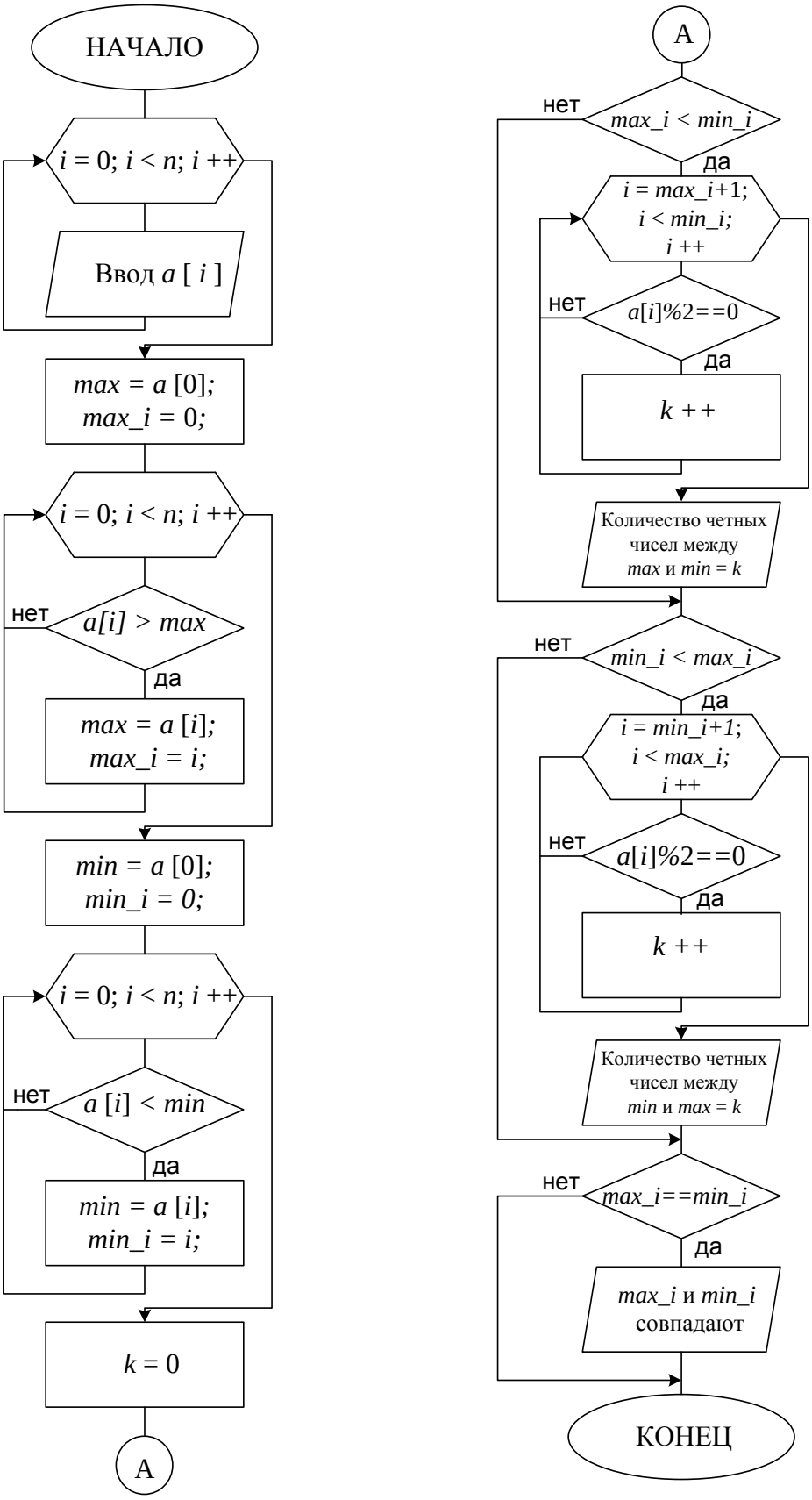

Рисунок 1.8 – Блок-схема алгоритма программы задания 8.2

Для того чтобы определить местоположение максимального элемента, необходимо задать начальное значение для индекса максимального элемента (***max_i***), например, равное нулю. Начальное значение для переменной максимального элемента (***max***) задается равным первому элементу массива ***a* [0]**. Далее необходимо просмотреть все элементы массива, поочередно сравнивая каждый элемент ***a* [*i*]** со значением максимума. Если очередной элемент (***a* [*i*]**) больше ранее найденного максимума (***max***), то принять этот элемент за новый максимум, а также в переменной ***max_i*** запомнить индекс данного элемента.

```
max = a [0]; // установка начальных значений
max_i = 0; // установка начальных значений для индекса
for (i = 0; i < n; i ++) // цикл для нахождения максимального элемента
        if (a [i] > max) // сравнение текущего значения с максимумом
        {
        max = a [i]; // запомнить значение максимального элемента
        max_i = i; // запомнить индекс максимального элемента
        }
```

Для того чтобы определить местоположение минимального элемента, необходимо задать начальное значение для индекса минимального элемента (***min_i***), например, равное нулю. Начальное значение для переменной минимального элемента (***min***) задается равным первому элементу массива ***a* [0]**. Далее необходимо просмотреть все элементы массива, поочередно сравнивая каждый элемент ***a* [*i*]** со значением минимума. Если очередной элемент (***a* [*i*]**) меньше ранее найденного минимума (***min***), то принять этот элемент за новый минимум, а также в переменной ***min_i*** запомнить индекс данного элемента.

```
min = a [0]; // установка начальных значений
min_i = 0; // установка начальных значений для индекса
for (i = 0; i < n; i ++) // цикл для нахождения минимального элемента
        if (a [i] < min) // сравнение текущего значения с минимумом
        {
        min = a [i]; // запомнить значение минимального элемента
```

min_i = i; *// запомнить индекс минимального элемента*
}

Для нахождения количества четных элементов между максимумом и минимумом, сначала необходимо определить границы просмотра массива. Если максимальный элемент расположен раньше, чем минимальный элемент (***max_i < min_i***), то необходимо организовать цикл ***for*** с параметрами – (***i = max_i + 1; i < min_i; i ++***)**.** Если минимальный элемент расположен раньше, чем максимальный элемент (***min_i<max_i***), то необходимо организовать цикл ***for*** с параметрами: ***(i = min_i+1; i<max_i; i++).*** Если индексы максимального и минимального элемента совпадают, то количество четных элементов равно 0. Переменную, которая принимает количество четных элементов, до цикла требуется обнулить (***k = 0***). Для того чтобы проверить является ли число четным, необходимо проверить остаток от деления этого числа на 2, если остаток равен 0 – число четное, если не равен нулю – число нечетное (***a [i] % 2 = = 0***).

k = 0; *// обнуление счетчика*
if (max_i < min_i) *// сравнение индексов*
{
for (i = max_i + 1; i < min_i; i++) *// цикл для нахождения количества четных чисел между максимумом и минимумом*
if (a [i] % 2 = = 0) *// проверка, является ли число четным*
k++; *// увеличение счетчика на единицу*
cout<< " Количество четных чисел между " << max << "и" << min << " равно " << k << '\n'; *// вывод на экран количества четных чисел*
}
if (min_i < max_i) *// сравнение индексов*
{
for (i = min_i + 1; i < max_i; i++) *// цикл для нахождения количества четных чисел между минимумом и максимумом*
if (a [i] % 2 = = 0) *// проверка, является ли число четным*
k++; *// увеличение счетчика на единицу*
cout<<" Количество четных чисел между "<<min<<"и "<<max<<" равно "<<k<<'\n'; *// вывод на экран количества четных чисел*

}

if (min_i = = max_i) *// проверка совпадают ли индексы максимального и минимального элементов*

cout<< ”Максимальный элемент – ” << a [max_i] << ”и минимальный элемент – ” << a [min_i] << ”совпадают \n”; *// вывод на экран сообщения*

Синтаксис программы

```
# include <iostream>
using  namespace  std;
const  int  n = 10; // количество элементов в массиве

int main ( )
{
    int a [n], k, i, min, max, min_i, max_i; // описание переменных
    for (i = 0; i < n; i ++) // цикл, перебирающий индекс массива
    {
       cout<< ” Введите ” << i << ” элемент массива: “;
       cin>> a [i]; // ввод с клавиатуры значений элементов массива
    }
    max = a [0];  // установка начального значения максимального элемента
    max_i = 0;  // установка начального значения индекса максимального элемента
    for (i = 0; i < n; i ++) // цикл для нахождения местоположения максимального элемента
       if (a [i] > max) // сравнение текущего элемента массива с максимальным
       {
           max = a [i]; // запомнить значение максимального элемента
           max_i = i; // запомнить индекс максимального элемента
       }
    min = a [0];  // установка начального значения минимального элемента
    min_i = 0;  // установка начального значения индекса минимального элемента
```

```
for (i = 0; i < n; i ++) // цикл для нахождения минимального элемента
    if (a [i] < min) // сравнение текущего элемента массива с минимальным
    {
        min = a [i]; // запомнить значение минимального элемента
        min_i = i; // запомнить индекс минимального элемента
    }
k = 0; // обнуление счетчика
if (max_i < min_i) // сравнение индексов
{
for (i = max_i + 1; i < min_i; i ++) // цикл для нахождения количества четных чисел между максимумом и минимумом
    if (a [i] % 2 = = 0) // проверка, является ли число четным
    k++; // увеличение счетчика на единицу
cout<< " Количество четных чисел между " << max << "и" << min << " равно " << k << '\n'; // вывод на экран количества четных чисел
}
if (min_i<max_i)// сравнение индексов
{
for (i = min_i + 1; i < max_i; i ++) // цикл для нахождения количества четных чисел между минимумом и максимумом
        if (a [i] % 2 == 0) // проверка на четность элемента массива
            k++;// увеличение счетчика на единицу
cout<< " Количество четных чисел между " <<min << "и" << max << " равно " << k << '\n'; // вывод на экран количества четных чисел
}
if (min_i = = max_i) // сравнение индексов
    cout<< "Максимальный элемент – " << a [max_i] << "и минимальный элемент – " << a [min_i] << "совпадают \n";// вывод на экран
return 0;
}
```

4. Отладка и запуск программы

Для отладки программы используем клавишу *F7*, убеждаемся в отсутствии ошибок и запускаем программу на исполнение с помощью комбинации клавиш *Ctrl+F5*. Ниже приведены результаты работы программы.

Введите 0 *элемент массива:* 27
Введите 1 *элемент массива:* 6
Введите 2 *элемент массива:* 103
Введите 3 *элемент массива:* 5
Введите 4 *элемент массива:* 18
Введите 5 *элемент массива:* 2
Введите 6 *элемент массива:* 21
Введите 7 *элемент массива:* 4
Введите 8 *элемент массива:* 13
Введите 9 *элемент массива:* 7
Количество четных чисел между 27 *и* 2 *равно* 3

Порядок выполнения лабораторной работы

1. В соответствии с номером по журналу выберите индивидуальное задание.
2. Разработайте алгоритм решения задачи.
3. Составьте текст программы.
4. Создайте проект в интегрированной среде разработки *Microsoft Visual Studio* (*lab5*_фамилия.*cpp).*
5. Введите текст программы.
6. Скомпилируйте программу. Если в программе есть ошибки, их необходимо исправить. Если ошибок нет, то появится сообщение об успешной компиляции.
7. Запустите программу на выполнение, проанализируйте результаты и убедитесь в правильности решения задачи.
8. Выполните отчет по лабораторной работе, который должен содержать:

− титульный лист;

− цель работы;

− индивидуальное задание;
− алгоритм работы программы;
− текст программы;
− результаты работы программы;
− выводы.

Индивидуальные задания 1

1. Создать массив из 20 вещественных чисел. Найти разность между минимальным и максимальным элементами массива.
2. Создать массив из 15 вещественных чисел. Определить наибольшее из отрицательных чисел, округлить его до ближайшего целого.
3. Создать массив из 10 целых чисел случайным образом. Определить четность наименьшего из положительных чисел.
4. Создать массив из 20 целых чисел. Найти порядковый номер второго максимального числа.
5. Создать массив из 15 вещественных чисел. Определить, образуют ли элементы массива возрастающую последовательность.
6. Создать одномерный массив из 25 целых чисел. Определить количество положительных чисел, которые делятся на 3 без остатка.
7. Создать массив из 30 целых чисел случайным образом. Вывести на экран элементы, порядковые номера которых являются нечетными.
8. Создать массив из 20 вещественных чисел. Определить какое количество элементов больше своих «соседей», т.е. предыдущего и последующего чисел.
9. Создать массив из 20 целых чисел. Определить сколько раз во вводимой последовательности происходит чередование положительных и отрицательных чисел, т.е. изменяется знак.
10. Создать массив из 30 целых чисел. Вычислить сумму тех из них, порядковые номера которых – простые числа.
11. Создать одномерный массив из 15 целых чисел, являющихся числами Фибоначчи.
12. Создать массив из 20 вещественных чисел. Поменять местами максимальный и минимальный элементы массива. Результат вывести на экран.

13. Дан массив из 20 вещественных чисел. Найти среднее арифметическое элементов массива между максимальным и минимальным элементами, включая их.

14. Создать массив из 30 целых чисел. Вывести на экран те элементы, которые являются числами Фибоначчи.

15. Ввести с клавиатуры массив из 20 целых чисел. Вывести на экран те элементы, которые являются совершенными числами.

16. Создать одномерный массив, состоящий из делителей целого числа *n*, введенного с клавиатуры. Вывести на экран элементы массива, являющиеся простыми числами.

17. Дан массив из 10 целых чисел. Найти минимальный элемент среди $a_1 * a_2$, $a_3 * a_4$, …, $a_9 * a_{10}$.

18. Создать массив из 10 целых чисел. Проверить, является ли массив палиндромом.

19. Массив из 10 вещественных чисел задан случайным образом. Заменить все отрицательные элементы массива их модулями.

20. Создать массив из цифр целого числа *n*, введенного с клавиатуры. Найти количество четных чисел созданного массива.

Индивидуальные задания 2

Вариант 1

1. В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

а) сумму положительных и количество отрицательных элементов массива;

б) произведение элементов массива, расположенных между максимальными и минимальными элементами.

2. Заполнить массив из 15 элементов последовательными простыми числами. Результат вывести на экран.

Вариант 2

1. В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

а) произведение положительных элементов массива;

б) сумму элементов массива, расположенных до минимального элемента.

2. Даны два массива из 20 вещественных чисел каждый. Найти количество элементов первого массива, которые не входят во второй.

Вариант 3

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

а) произведение отрицательных элементов массива и количество положительных;

б) сумму положительных элементов массива, расположенных до максимального элемента.

2. Даны два массива из 20 целых чисел каждый. Найти минимальный из элементов первого массива, который не входит во второй.

Вариант 4

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

а) сумму положительных элементов массива;

б) произведение элементов массива, расположенных между максимальным по модулю и минимальным по модулю элементами.

2. Дан массив из 30 вещественных чисел. Найти среднее арифметическое элементов, индексы которых являются числами Фибоначчи.

Вариант 5

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

а) произведение элементов массива с четными номерами;

б) сумму элементов массива, расположенног между первым и последним нулевыми элементами.

2. Даны два массива a и b из 20 вещественных чисел. Заполнить массив b так, чтобы b_i было равно среднему арифметическому всех элементов массива a, кроме a_i.

Вариант 6

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

а) сумму элементов массива с нечетными номерами;

б) сумму элементов массива, расположенным между первым и последним отрицательными элементами.

2. Даны два массива a и b из 15 вещественных чисел. Найти $(a_1 + b_{15})$ $(a_2 + b_{14})$ … $(a_{15} + b_1)$.

Вариант 7

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

а) произведение элементов, не равных нулю;

б) произведение элементов массива, расположенных между первым и вторым отрицательными элементами.

2. Дан массив из 10 вещественных чисел. Определить, есть ли в массиве одинаковые числа.

Вариант 8

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

а) минимальный из положительных элементов, его порядковый номер;

б) сумму элементов массива, расположенных до последнего положительного элемента.

2. Дан массив из 20 вещественных чисел. Сформировать два других массива, включая в первый четные элементы данного массива, а во второй нечетные в порядке их следования.

Вариант 9

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

а) среднее арифметическое двух наименьших элементов массива;

б) сумму элементов массива, расположенных между первым и вторым отрицательными элементами.

2. Дан массив из 10 вещественных чисел. Заменить все элементы, начиная со второго суммой всех предшествующих.

Вариант 10

1. В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

а) количество и произведение отрицательных элементов массива;

б) сумму элементов массива, расположенных между первым и последним положительными элементами.

2. Дан массив из 20 целых чисел. Заменить каждый элемент наибольшим из всех предшествующих.

Вариант 11

1. В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

а) сумму положительных и количество отрицательных элементов массива;

б) сумму элементов массива, расположенных между первым и вторым положительными элементами.

2. Ввести с клавиатуры целое число *n* ($n < 999\,999$). Заполнить массив цифрами данного числа, расположенными в обратном порядке (первый элемент равен последней цифре, второй элемент равен предпоследней цифре и т.д.). Незаполненные элементы должны быть равны 0.

Вариант 12

1. В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

а) минимальный по модулю элемент массива;

б) сумму модулей элементов массива, расположенных после первого элемента, равного нулю.

2. Дан массив из 10 вещественных чисел. Найти среднее арифметическое элементов массива между максимальным и минимальным элементами, включая их (не включая их).

Вариант 13

1. В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

а) номер минимального по модулю элемента массива;

б) сумму модулей элементов массива, расположенных после первого отрицательного элемента.

2. Дан массив из 18 вещественных чисел. Преобразовать его следующим образом: a_1, a_{18}, a_2, a_{17}, a_3, a_{16} ….

Вариант 14

1. В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

а) максимальный элемент из отрицательных и его порядковый номер;

б) сумму целых частей элементов массива, расположенных после последнего отрицательного элемента.

2. Дан массив из 25 целых чисел. Определить, образует ли массив возрастающую последовательность.

Вариант 15

1. В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

а) номер максимального по модулю элемента массива;

б) сумму элементов массива, расположенных после первого положительного элемента.

2. Дан массив из 30 вещественных чисел. Убрать из него все элементы равные максимальному и конец массива заполнить 0.

Вариант 16

1. В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

а) количество элементов массива, лежащих в диапазоне от *a* до *b;*

б) сумму элементов массива, расположенных после максимального элемента.

2. Дан массив из 20 целых чисел. Создать массив, обратный данному ($a_0=a_{19}$, $a_1=a_{18}$, $a_2=a_{17}$ …).

Вариант 17

1. В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

а) количество элементов массива, равных 0;

б) сумму элементов массива, расположенных после минимального элемента.

2. Дан массив из 20 целых чисел. Вывести на экран те элементы, которые являются числами Фибоначчи.

Вариант 18

1. В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

а) максимальный элемент из отрицательных и его порядковый номер;

б) произведение элементов массива, расположенных после максимального по модулю элемента.

2. Дан массив из 25 целых чисел. Вывести на экран те элементы, которые являются простыми числами.

Вариант 19

1. В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

а) количество отрицательных элементов массива;

б) сумму модулей элементов массива, расположенных после минимального по модулю элемента.

2. Дан массив из 20 целых чисел. Вывести на экран те элементы, которые являются степенями 2.

Вариант 20

1. В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

а) количество положительных элементов массива;

б) сумму элементов массива, расположенных после последнего элемента, равного нулю.

2. Дан массив из 30 вещественных чисел. Найти среднее арифметическое элементов, индексы которых являются числами Фибоначчи.

Контрольные вопросы

1. Что будет на экране после выполнения следующего фрагмента программы?

int a [5] = {1, 2, 3, 4, 5};
cout<<*a* [2.2+1.8]*;*

Варианты ответов: **А.** 5; **Б.** 0; **В.** 4; **Г.** Ошибка на этапе компиляции; **Д.** Ошибка на этапе выполнения.

2. Что будет на экране после выполнения следующего фрагмента программы?

int a [10] = {6, 7.8, 8.0, –6, –5.7, 5, 5.7};
cout<<*a* [6 % 8 / 4+21 % 9]*;*

Варианты ответов: **А. –**5.7; **Б.** –5; **В.** 6; **Г.** –6; **Д.** 5.

3. Что будет на экране после выполнения следующего фрагмента программы?

int s = 10;
int a [*s*] = {0};
for (int i = 0; *i* < *s; i++)*
    cout<<*a* [*i*]<<” * “*;*

Варианты ответов:

А. 0 * 1 * 2 * 3 * 4 * 5 * 6 * 7 * 8 * 9;

Б. 0 * 0 * 0 * 0 * 0 * 0 * 0 * 0 * 0 * 0;

В. 0 1 2 3 4 5 6 7 8 9;

Г. Ошибка на этапе компиляции;

Д. Ошибка на этапе выполнения.

4. Что будет на экране после выполнения следующего фрагмента программы?

int a [5] = {1, 2, 3, 4, 5};
for (int i = 0; *i* < 5*; i++) {*
    a [*i*] = 5 –*i;*
    cout<<*a* [*i*]<<” “*; }*

Варианты ответов:

А. 0 1 2 3 4;

Б. 5 4 3 2 1;

В. 4 3 2 1 0;

Г. 1 2 3 4 5;

Д. Ошибка на этапе компиляции.

5. Что будет на экране после выполнения следующего фрагмента программы?

```
int s = 2;
int a [10] = {s*=s, s*=s, s*=s, s*=s};
for (int i = 0; i < 10; i++)
        cout<<a [i]<<"  ";
```

Варианты ответов:

А. 2 2 2 2 2 2 2 2 2 2;

Б. 2 2 2 2 0 0 0 0 0 0;

В. 4 16 256 65536 0 0 0 0 0 0;

Г. Ошибка на этапе компиляции;

Д. 0 0 0 0 0 0 0 0 0 0.

6. Укажите правильные объявления массивов

А. *int a* [10];

Б. *const float s; float a* [*s*];

В. *float a* [5.5];

Г. *const int n* = 10; *double* [*n*];

Д. *const int n* = 10; *char a* [*n*].

7. Что будет на экране после выполнения следующего фрагмента программы?

```
int s, i;
int a [10] = {2, 3, 4, 5, 6, 7, 8};
s = 0;
for (i = 0; i <= 5; i++)
        s+=a [a [i] ];
cout<<s;
```

Варианты ответов: **А.** 20; **Б.** 30; **В.** 35; **Г.** 26; **Д.** 0.

8. Что будет на экране после выполнения следующего фрагмента программы?

```
int s, i;
int a [10] = {2, 3, 4, 5, 6, 7, 8};
s = 0;
for (i = 1; i < 5; i++)
        s+=a [ i ];
cout<<s;
```

Варианты ответов: **А.** 35; **Б.** 18; **В.** 20; **Г.** 26; **Д.** 0.

ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ А

Цикл с предусловием *while*

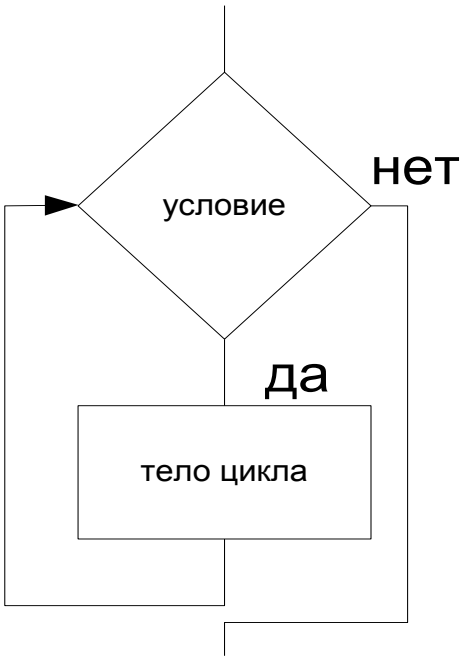

Рисунок А.1 – Блок-схема цикла с предусловием ***while***

Синтаксис цикла с предусловием ***while***

while (условие)

оператор;

Составной оператор

```
while (условие)
{
  оператор 1;
  оператор 2;
  …;
}
```

ПРИЛОЖЕНИЕ Б

Цикл с предусловием *do … while*

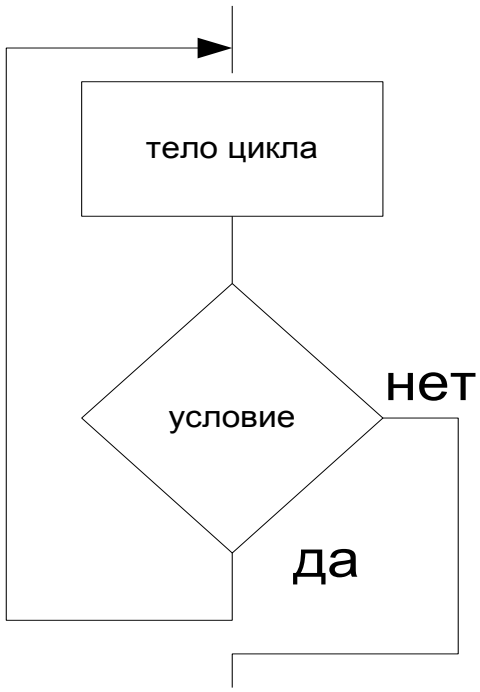

Рисунок Б.1 – Блок-схема цикла с постусловием ***do … while***

Синтаксис цикла с предусловием ***do … while***

```
{
    оператор;
}
while (условие);
```

Составной оператор

```
do
{
    оператор 1;
    оператор 2;
    …,
}
while (условие);
```

ПРИЛОЖЕНИЕ В

Цикл с параметрами *for*

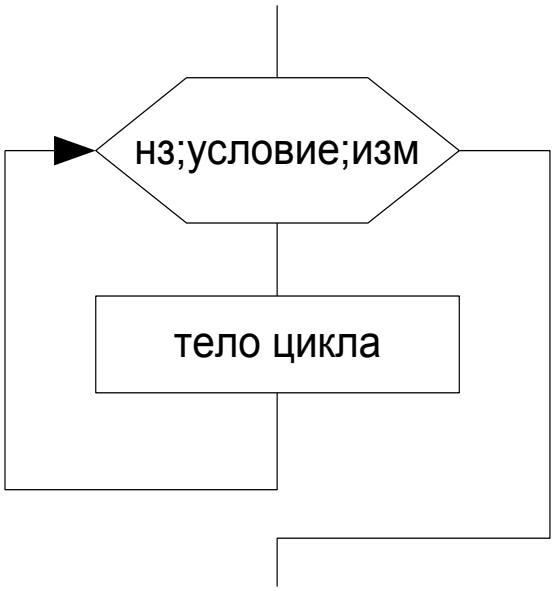

Рисунок В.1 – Блок-схема цикла с параметрами ***for***

Синтаксис цикла с параметрами ***for***

for (НЗ; условие; ИЗМ)

оператор;

Составной оператор

for (НЗ; условие; ИЗМ)
{
оператор 1;
оператор 2;
…
оператор n;
}

СПИСОК ЛИТЕРАТУРЫ

1. Светозарова Г. И. Практикум по программированию на алгоритмических языках / Г. И. Светозарова, Е. В. Сигитов, А. В. Козловский. – Минск: Наука, 1980. – 317 с.

2. Дейтел Х. М. , П. Дж. Дейтел Как программировать на С++ / Х. М. Дейтел, П. Дж. Дейтел. – М. : ЗАО «Издательство БИНОМ», 2000 – 1024 с.

3. . Язык программирования С++ / Бъерн Страуструп. – М.: Бином, 2002. – 1098с.

4. Шилдт Герберт. Справочник программиста по С/С++ / Герберт Шилдт; пер. с англ. – 3-е изд.:. –М.: Издательский дом «Вильямс», 2003. – 432 с.

5. Павловская Т.А., Щупак Ю.А. Структурное программирование С/С++. Практикум – Спб.: Питер, 2007. – 239 с.

6. Подбельский В.В. Язык Си+: учеб. пособ. / В.В. Подбельский – М.: БИНОМ, 1995 – 400 с.

СОДЕРЖАНИЕ

НАВЧАЛЬНЕ ВИДАННЯ

Методичні вказівки
для виконання практичних та лабораторних робіт

з курсу «Інформатика»

ОСНОВИ ПРОГРАМУВАННЯ НА МОВІ С++.
ОДНОМІРНІ МАСИВИ І ЦИКЛИ

для студентів факультету «Автоматика та приладобудування»
денної та заочної форми навчання

Укладачі: ТВЕРИТНИКОВА Олена Євгенівна
КРИЛОВА Вікторія Анатоліївна
ОПРИШКІНА Марина Ігорівна

Відповідальний за випуск *О.І. Рогачов*

Редактор *Л.А. Пустовойтова*

План 2014 р., поз. 6

Підп. до друку . .14. Формат 60×84 1/16. Папір друк. № 2.
Друк – ризографія. Гарнітура Times New Roman. Ум. друк. арк. 3,2.
Обл. – вид. арк. 2,7. Наклад 100 прим. Зам. № .Ціна договірна.

Видавничий центр НТУ «ХПІ». 61002, Харків, вул.. Фрунзе, 21.
Свідоцтво про реєстрацію ДК № 3657 від 24.12.2009 р.

Друкарня НТУ «ХПІ», 61002, Харків, вул. Фрунзе, 21.