Министерство науки и высшего образования Российской Федерации
ФГБОУ ВО «Удмуртский государственный университет»
Институт права, социального управления и безопасности
Кафедра информационной безопасности в управлении

М. М. Гайсин

МЕТОДИКА ВЫПОЛНЕНИЯ ЛАБОРАТОРНЫХ РАБОТ ПО ДИСЦИПЛИНЕ «ЯЗЫКИ ПРОГРАММИРОВАНИЯ»

Методические рекомендации

Ижевск
2022

УДК 004.43(075.8)
ББК 32.973.2я73-5
М545

Рекомендовано к изданию Учебно-методическим советом УдГУ

Рецензент: канд. техн. наук О.В. Меркушев

Гайсин М. М.

М545 Методика выполнения лабораторных работ по дисциплине «Языки программирования»: метод. рек. – Ижевск : Удмуртский университет, 2022. – 29 с.

Методические рекомендации подготовлены в соответствии с федеральным государственным образовательным стандартом высшего образования по направлению подготовки 10.03.01 «Информационная безопасность» и 10.05.05 «Безопасность информационных технологий в правоохранительной сфере». В них приводятся методические рекомендации по выполнению лабораторных работ обучающихся, требования к оформлению отчета работ и образцы их заполнения.

Рекомендации предназначены для обучающихся очной и очно-заочной форм обучения направления подготовки 10.03.01 «Информационная безопасность» и 10.05.05 «Безопасность информационных технологий в правоохранительной сфере» Института права, социального управления и безопасности, а также для преподавательского состава.

УДК 004.43(075.8)
ББК 32.973.2я73-5

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

При изучении дисциплины «Языки программирования» обучающиеся выполняют лабораторные работы, задания к которым содержатся в соответствующих главах Самоучителя [1].

Варианты заданий выдаются обучающимся заранее с тем, чтобы они имели возможность подготовиться к выполнению лабораторной работы: просмотреть теоретический материал по теме работы и продумать алгоритмы решения задач.

Программы пишутся на языке C++ и после отладки предоставляются преподавателю. После устранения всех замечаний составляется отчет (требования к отчету см. ниже) и сдается преподавателю без ошибок в электронном виде в формате MS Word.

Цель данных методических рекомендаций − привить обучающимся навыки создания и отладки программ на языке C++ в среде разработки MS Visual Studio 17 на основе примеров, содержащихся в главах Самоучителя [1] и оформлению блок-схем алгоритмов на основе ГОСТ 19.701-90 [2].

1. ТЕМЫ И СОДЕРЖАНИЕ ЛАБОРАТОРНЫХ РАБОТ

№	Тема и содержание лабораторной работы	Виды работ
0	**Среда разработки MS Visual Studio (MS VS) 17), способы отладки и тестирования программ**	Ввод и отладка готовых решений задач из электронного учебника [3]
1	**Основы языка C++:** 1. Процедурное и объектно-ориентированное программирование 2. Структура программы в C++ 3. Создание простой программы 4. Использование переменных 5. Объявление и инициализация переменной 6. Базовые типы данных 7. Константы и литералы 8. Арифметические операторы 9. Логические операторы 10. Операторы сравнения 11. Побитовые операторы и двоичное представление чисел 12. Оператор присваивания и приведение типов 13. Тернарный оператор	Разработка и отладка программы согласно списку задач для самостоятельного решения гл. 1 Самоучителя [1], составление отчета по лабораторной работе и блок-схемы алгоритма в соответствии с требованиями ЕСПД [2]
2	**Управляющие инструкции:** 1. Условный оператор IF() 2. Условный оператор SWITCH() 3. Оператор цикла FOR() 4. Оператор цикла WHILE()	Разработка и отладка программы согласно списку задач для самостоятельного решения гл. 2 Самоучителя [1], составление отчета

	5. Инструкция безусловного перехода 6. Примеры решения задач	по лабораторной работе и блок-схемы алгоритма в соответствии с требованиями ЕСПД [2]
3	**Указатели, ссылки и массивы:** 1. Объявление и использование указателей 2. Адресная арифметика и сравнение указателей 3. Многоуровневая адресация 4. Знакомство со ссылками 5. Статические одномерные массивы 6. Указатель на массив 7. Двумерные массивы 8. Инициализация массивов 9. Массивы символов 10. Массивы указателей	Разработка и отладка программы согласно списку задач для самостоятельного решения гл. 3 Самоучителя [1], составление отчета по лабораторной работе и блок-схемы алгоритма в соответствии с требованиями ЕСПД [2]
4	**Функции:** 1. Объявление и использование функций 2. Механизмы передачи аргументов 3. Передача указателя аргументом функции 4. Передача массива аргументом функции 5. Передача строки аргументом функции 6. Аргументы функции MAIN() 7. Аргументы по умолчанию 8. Возвращение функцией указателя 9. Возвращение функцией ссылки 10. Указатели на функции	Разработка и отладка программы согласно списку задач для самостоятельного решения гл. 4 Самоучителя [1], составление отчета по лабораторной работе и блок-схемы алгоритма в соответствии с требованиями ЕСПД [2]

	11. Рекурсия 12. Перегрузка функцией	
5	**Текстовые строки и динамические массивы:** 1. Создание и инициализация строк 2. Нуль-символ окончания строки 3. Функции для работы со строками и символами 4. Строчные литералы 5. Двумерные символьные массивы 6. Динамическое выделение памяти 7. Динамические массивы 8. Многомерные динамические массивы	Разработка и отладка программы согласно списку задач для самостоятельного решения гл. 5 Самоучителя [1], составление отчета по лабораторной работе и блок-схемы алгоритма в соответствии с требованиями ЕСПД [2]
6	**Структуры, объединения и перечисления:** 1. Структуры 2. Массивы структур 3. Передача структур аргументами функций 4. Указатели на структуры 5. Битовые размеры поля 6. Объединения 7. Перечисления и определение типов	Разработка и отладка программы согласно списку задач для самостоятельного решения гл. 6 Самоучителя [1], составление отчета по лабораторной работе и блок-схемы алгоритма в соответствии с требованиями ЕСПД [2]
7	**Объектно-ориентированное программирование.** **Классы и объекты**: 1. Объявление класса 2. Открытые и закрытые члены класса 3. Статические члены класса 4. Перегрузка методов	Разработка и отладка программы согласно списку задач для самостоятельного решения гл. 7 Самоучителя [1], составление отчета по лабораторной работе и блок-схемы алго-

		ритма в соответствии с требованиями ЕСПД [2]
8	**Работа с объектами:** 1. Передача объектов аргументами 2. Возвращение результатом объектов 3. Указатели на объекты 4. Указатели на члены класса 5. Использование ссылок на объекты 6. Массивы объектов 7. Динамическое выделение памяти под объекты 8. Дружественные функции и классы	Разработка и отладка программы согласно списку задач для самостоятельного решения гл. 8 Самоучителя [1], составление отчета по лабораторной работе и блок-схемы алгоритма в соответствии с требованиями ЕСПД [2]
9	**Конструкторы и деструкторы:** 1. Создание и перегрузка конструктора 2. Использование деструкторов 3. Вызов конструктора 4. Конструктор создания копии 5. Создание бинарного дерева 6. Поле-объект 7. Поле-массив объектов 8. Вызов в конструкторе метода	Разработка и отладка программы согласно списку задач для самостоятельного решения гл. 9 Самоучителя [1], составление отчета по лабораторной работе и блок-схемы алгоритма в соответствии с требованиями ЕСПД [2]
10	**Перегрузка операторов:** 1. Внешняя операторная функция для переопределения бинарного оператора 2. Перегрузка операторной функции 3. Переопределение унарных операторов внешними функциями	Разработка и отладка программы согласно списку задач для самостоятельного решения гл. 10 Самоучителя [1], составление отчета по лабораторной работе и блок-

	4. Перегрузка операторов методами класса 5. Перегрузка оператора присваивания 6. Индексирование объектов 7. Вектор-функция	схемы алгоритма в соответствии с требованиями ЕСПД [2]
11	**Наследование и виртуальные функции:** 1. Наследование классов и типы наследования 2. Переопределение методов и виртуальные функции 3. Многоуровневое наследование 4. Многократное наследование 5. Конструкторы и деструкторы при наследовании 6. Чисто виртуальные методы и абстрактные классы 7. Существующие несуществующие члены класса	Разработка и отладка программы согласно списку задач для самостоятельного решения главы 11 Самоучителя [1], составление отчета по лабораторной работе и блок-схемы алгоритма в соответствии с требованиями ЕСПД [2]
12	**Шаблоны:** 1. Обобщенные функции 2. Перегрузка обобщенных функций 3. Обобщенные классы 4. Типы по умолчанию и явная специализация класса	Разработка и отладка программы согласно списку задач для самостоятельного решения гл. 12 Самоучителя [1], составление отчета по лабораторной работе и блок-схемы алгоритма в соответствии с требованиями ЕСПД [2]

2. МЕТОДИКА ВЫПОЛНЕНИЯ РАБОТ

Разработка программы в соответствии с индивидуальным заданием обучающегося производится по аналогии с предоставленными в Самоучителе [1] примерами программ и их исходными кодами на виртуальном диске соответствующей лабораторной работы и фрагментами кодов работающих программ, разобранных на соответствующих лекциях.

При изучении среды разработки MS Visual Studio 17 (Лабораторная работа № 0) готовые коды программ в соответствии со случайно сгенерированными номерами задач выбираются из электронного учебника [3], самостоятельно вводятся и отлаживаются студентом. Сдача работы происходит путем демонстрации студентом работающего кода 4-х задач из разных разделов электронного учебника.

3. СОДЕРЖАНИЕ ОТЧЕТА ПО ЛАБОРАТОРНЫМ РАБОТАМ 1 И 2

Выполненный отчет по лабораторным работам 1 и 2 должен содержать следующие пункты:

1. Титульный лист (оформленный в соответствии с предъявляемыми требованиями).

2. Цель работы (своя для каждой лабораторной работы).

3. Задание.

4. Схема алгоритма по ГОСТ 19.701-90 [2].

5. Текст программы.

5.1. Каждая строка или группа однотипных строк должны содержать комментарий о смысле данных строк.

5.2. Программа должна в начале работы выводить информацию о разработчике программы и ее назначении.

6. Результат тестирования программы в виде скриншота.

7. Проверка правильности вычислений.

4. СОДЕРЖАНИЕ ОТЧЕТА ПО ЛАБОРАТОРНЫМ РАБОТАМ 3-12

Выполненный отчет по лабораторным работам 3-12 должен содержать следующие пункты:

1. Титульный лист (оформленный в соответствии с предъявляемыми требованиями).

2. Цель работы (своя для каждой лабораторной работы).

3. Задание.

4. Схема алгоритма по ГОСТ 19.701-90 [2].

5. Требования к предоставляемому коду программы.

5.1. Каждая строка или группа однотипных строк должны содержать комментарий о смысле данных строк.

5.2. Программа должна в начале работы выводить информацию о разработчике программы и ее назначении.

5.3. Программа должна содержать код, соответствующий теме главы Самоучителя.

5.4. Программа должна быть зациклена до момента, пока пользователь не пожелает прекратить работу программы.

5.5. Ввод-вывод данных по желанию пользователя должен производиться либо с клавиатуры и на дисплей, либо с текстового файла (в текстовый файл), предварительно созданного пользователем в указанном месте (создаваемого программой).

5.6. Обязательна проверка вводимых пользователем данных на корректность (защита от «дурака»).

5.7. Непосредственно перед вводом данных необходимо продемонстрировать пользователю программы контрольный пример с правильными результатами расчетов.

6. Результат тестирования программы в виде скриншота.

7. Вывод о соответствии текста программы нужной главе Самоучителя.

5. ОФОРМЛЕНИЕ БЛОК-СХЕМЫ АЛГОРИТМА

Формально блок-схема является по отношению к коду программы первичным документом и составляется на этапе выполнения эскизного проекта разработки программы. Однако в лабораторном практикуме приходится оформлять ее постфактум. Способ оформления возможен любой доступный обучающемуся: редактор MS Word (см. рис.1), Visio, чертеж на бумаге, вставленный в виде графического файла в файл отчета и пр. При этом сама блок-схема алгоритма должна соответствовать ГОСТу [2].

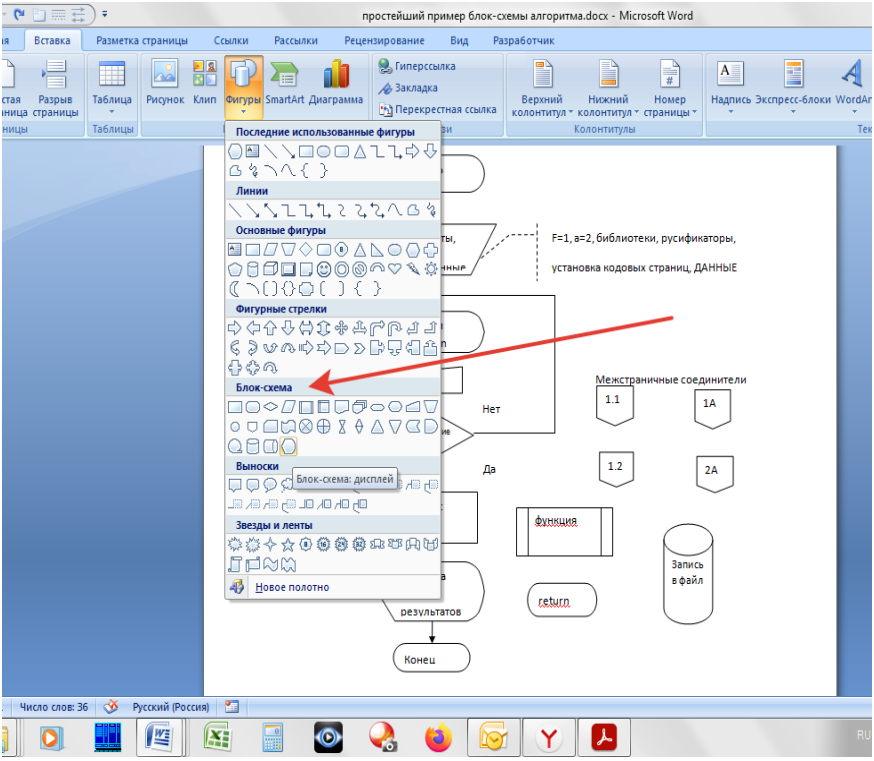

Рис. 1. Пример использования стандартных возможностей MS Word для оформления блок-схемы

Имеющиеся в редакторе стандартные блоки соответствуют требованиям ГОСТа [2].

5.1. Простейший пример использования блоков по ГОСТу 19.701-90

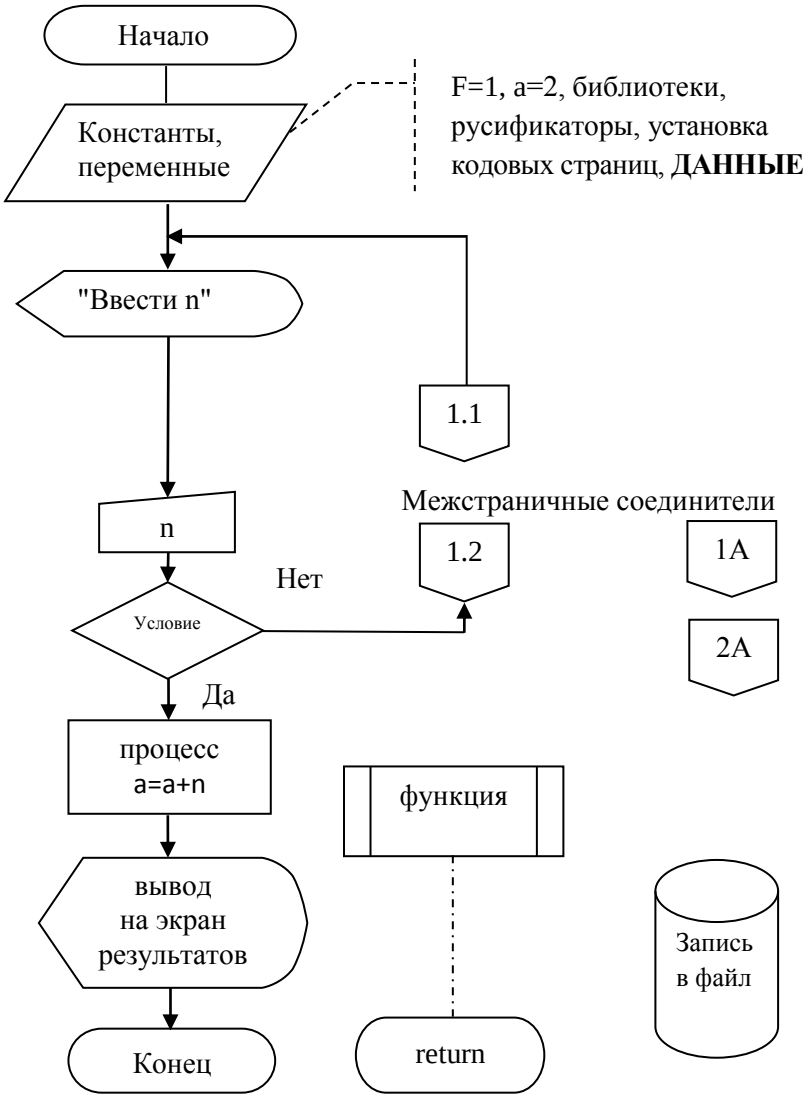

Рис. 2. Простейший пример блок-схемы алгоритма

Стрелочки на схеме алгоритма обязательны, если направление стрелки влево или вверх, пересечение стрелок не допускается.

Примечание: На рис. 2 приведены наиболее часто встречающиеся блоки схем алгоритмов. Используемые в сети Интернет для ввода/вывода **данных** блоки (параллелограмм) устарели, т. к. ввод/вывод осуществляется теперь не с перфокарт или бланков, а с клавиатуры/дисплея или с файлов на магнитных дисках (памяти). Аналогичная ситуация с универсальными обозначениями циклов − хотя и соответствуют ГОСТу, но применимы при разработке проектов больших систем, в которых заранее может быть неизвестно не только то, какой именно цикл будет использоваться, но и даже на каком именно языке программирования будет написан данный модуль ПО!

5.2. Блок-схемы типовых управляющих конструкций языка C++

5.2.1. Цикл for (int n = 1; n <= N; n++)

5.2.1.1. Первый вариант

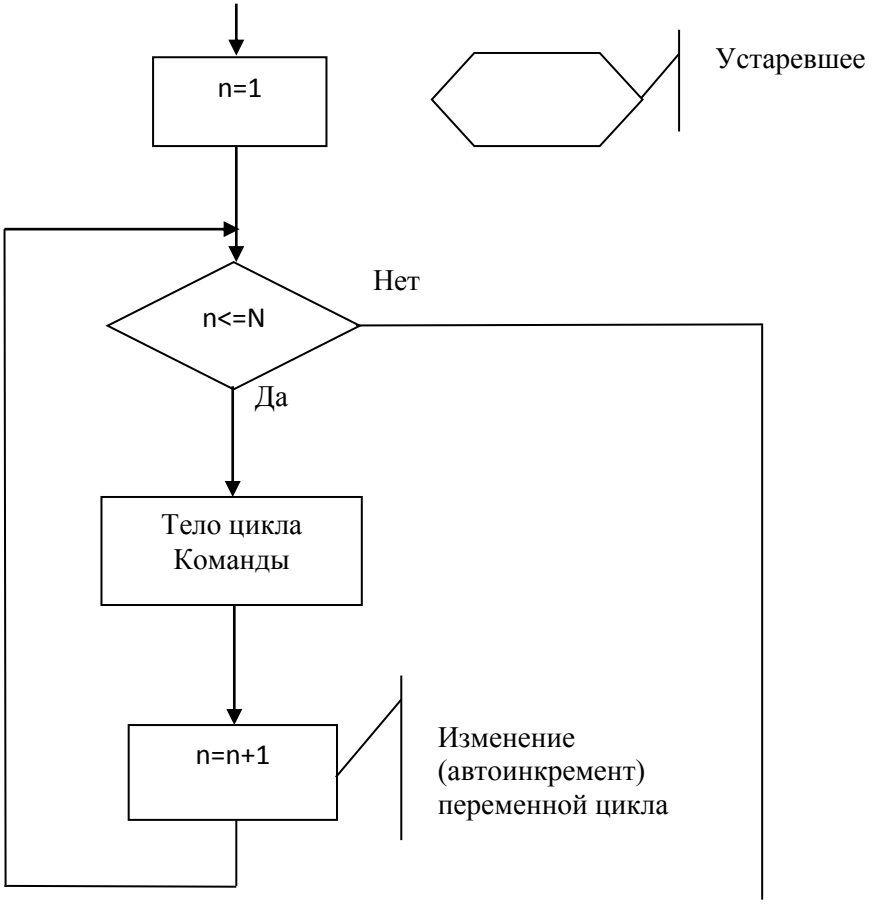

Рис. 3. Схема алгоритма цикла FOR

Пример кода:

```
int main(){
int i,j;
for(i=1;i<=3;i++){  // цикл 1
  for(j=1;j<=5;j++) cout<<3*(j-1)+i<<" "; // тело цикла 1 вся строка.
  cout<<"\n"; }
return 0;
}
cout<<3*(j-1)+i<<" ";  // тело цикла 2
```

5.2.1.2. Другой вариант

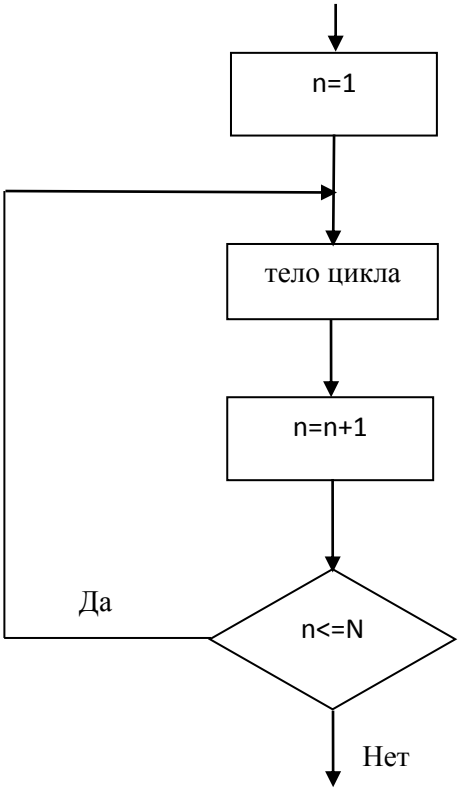

Рис.4. Возможный вариант схемы алгоритма цикла FOR

5.2.2. Цикл while − ПОКА выполняется условие

```
while (условие)
{ Тело цикла}
```

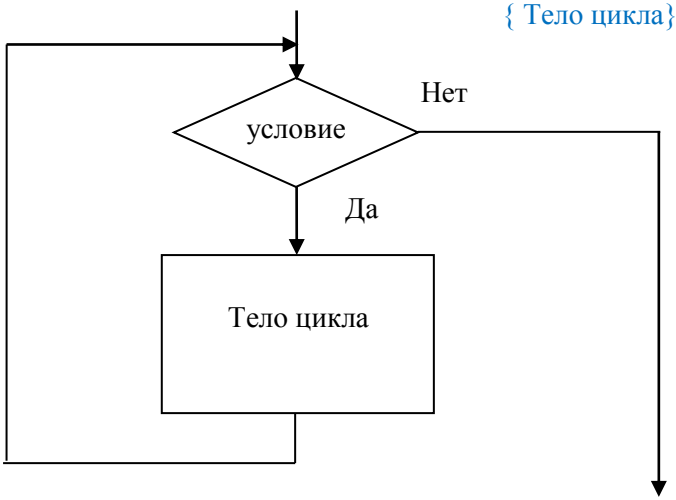

Рис. 5. Схема алгоритма цикла While

5.2.3. Цикл do-while – ДЕЛАТЬ ПОКА не выполнится условие

```
do{
}
while
(условие)
```

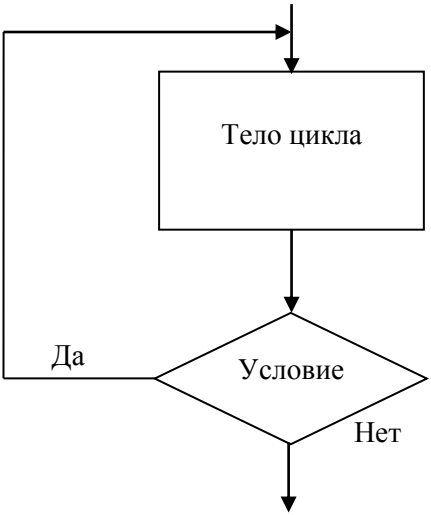

Рис.5. Схема алгоритма цикла Do-While.

5.2.4. Оператор Switch (ключ)

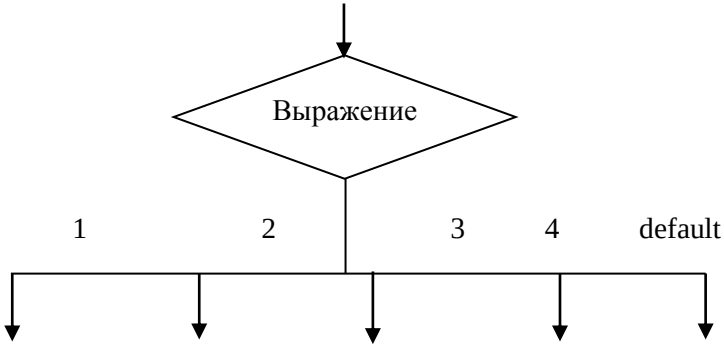

Рис. 6. Вариант представления схемы алгоритма оператора Switch согласно п. 4.3.1.1. ГОСТ 19.701-90

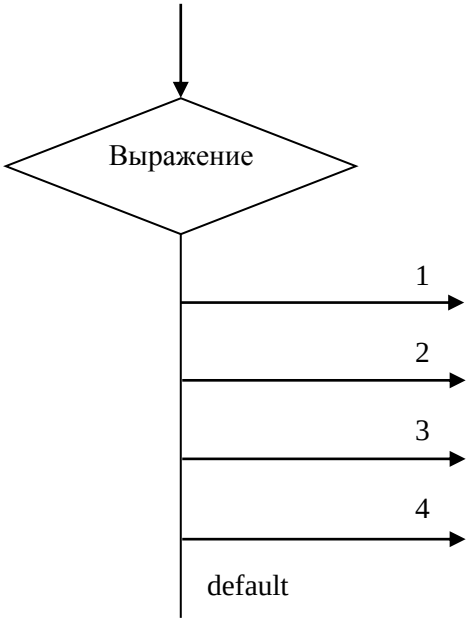

Рис. 7. Другой вариант представления схемы алгоритма оператора Switch согласно п. 4.3.1.1. ГОСТ 19.701-90

6. ПРИМЕРЫ ПРИЕМОВ ПРОГРАММИРОВАНИЯ

6.1. Защита от неправильного ввода данных

Защита от неправильного ввода данных может быть нескольких уровней.

Первый − рекомендация пользователю ввести данные в нужном формате или диапазоне: «Введите число с десятичной точкой» или «Введите положительное число в диапазоне от 0 до 20». При этом последствия ввода неправильных данных остаются на совести пользователя программы, а автор программы не гарантирует правильность работы при выборе неправильных данных.

Второй − проверка правильности ввода и реакция программы на неправильно введенные значения, например, «Введенное Вами число не соответствует требованиям, повторите ввод».

Третий − интеллектуальная реакция на попытку ввода данных: «Вы ввели число с запятой вместо точки, возможно Вы хотели ввести следующее...», и программа пытается исправить введенные данные сама, требуя подтверждения пользователя своим действиям.

Простейшая защита от «дурака» при выборе пунктов меню с помощью цикла Do-While:

```
do {
  cout << "Введите N>0: ";
  cin >> N;
  }
while( N <= 0 );
```

6.2. Зацикливание программы

Один из простейших способов зацикливания программы до момента, пока пользователь не захочет выйти из нее с помощью цикла while:

```
#include "pch.h"
#include <iostream>
using namespace std;

int main()
{
	setlocale(LC_ALL, "Russian");
	bool cyrcle = true;  // переменная цикла
	char  yes;  // переменная для реакции пользователя
	while (cyrcle) {
		// Сама программа..
      cout << "Hello World!\n";

		cout << "Желаете продолжать работу? y/n   ";
			cin >> yes;
			if (yes == 'n')cyrcle = false;
	}
}
```

Вариантов подобного зацикливания может быть огромное количество. Напрямую возможно заменить цикл while на цикл do-while. Возможно использование бесконечного цикла while(true) с прерыванием цикла по оператору if (yes == 'n') break;

Использование зацикливания с помощью операторов if, goto и метки возможно, но не рекомендуется (считается низким стилем программирования!).

6.3. Пример организации меню с помощью оператора Switch

```
#include "pch.h"
#include <iostream>
using namespace std;
void m1() {
	cout << " выполнилась функция m1";
}
void m2() {
	cout << " выполнилась функция m2";
}
void m3() {
	cout << " выполнилась функция m3";
}

int main()
{
	setlocale(LC_ALL, "Russian");
	int m;
// Следует стандартный заголовок программы и перечень пунктов меню
		cout << "Выберите пункт меню от 1 до 3\n m=? \n";
		cin >> m;
		switch (m)

		{
		case 1:
			m1();
			break;
		case 2:
			m2();
```

```
                break;
            case 3:
                m3();
                break;
            default:
                cout << "ненормальное завершение первой части программы\n";
            }
    cout << "\n работа программы завершена";
    return 0;
}
```

Схема алгоритма к данному примеру приведена ниже (см. рис. 8).

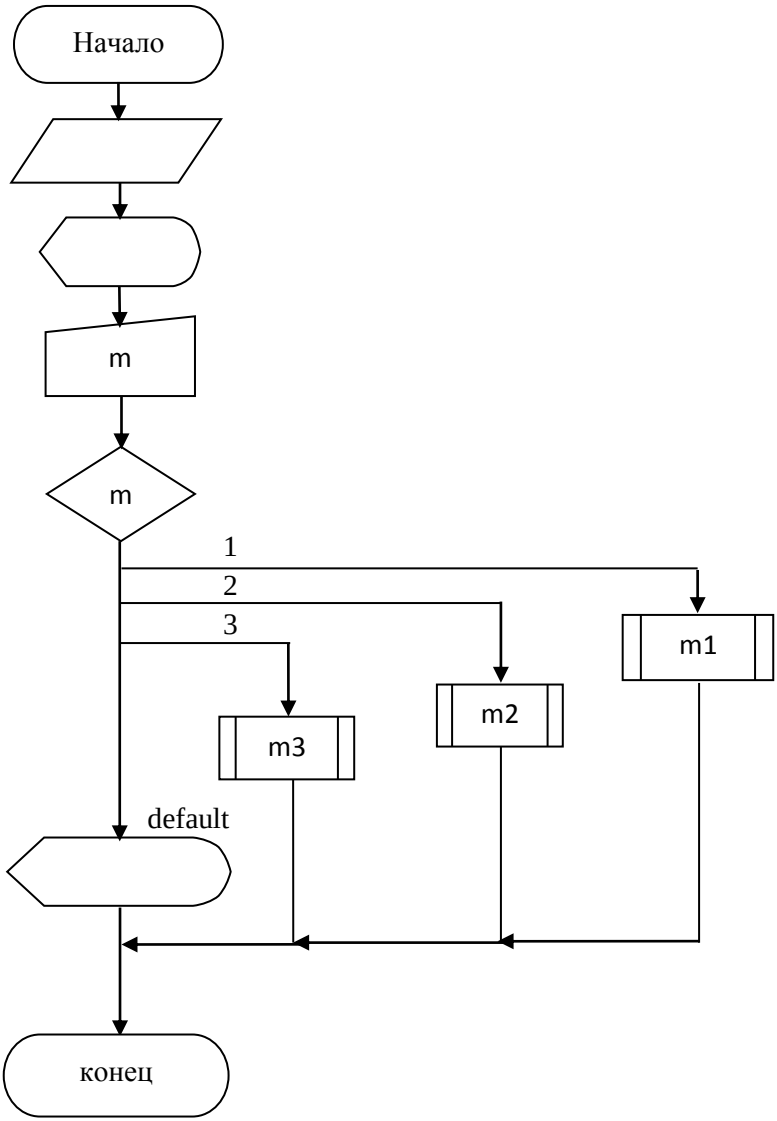

Рис.8. Схема алгоритма программы для организации меню с помощью оператора Switch

6.4. Работа с файлами

6.4.1. Чтение данных из файла

```
#include "pch.h"
#include <iostream>
#include <fstream>
#include <iomanip>
#include <stdlib.h>
using namespace std;

int main()
{
	setlocale(LC_ALL, "RUS");
	float a[4], s = 0;
	int  n = 0;
	fstream F; // назначаем файловому потоку имя f
	F.open("D:\\238.txt"); // открываем файл по указанному пути, если не указывать путь,
	// то программа будет искать файл в каталоге с исходным текстом
	if (F) // если нет ошибки при чтении файла
	{
		while (!F.eof()) // ПОКА не конец файла
		{
			F >> a[n]; // из файла берем число и записываем в элемент массива a
			cout << a[n] << endl; // выводим на экран для контроля
			s += a[n]; // накапливаем сумму
			n++; // автоинкремент индекса массива
		}
		F.close(); // закрываем файл
```

```
            cout << "Среднее арифметическое = " << s / n << endl;
        }
        else cout << "Файл не существует" << endl; // если ошибка при открытии файла
        return 0;
}
```

Пример исходного файла:

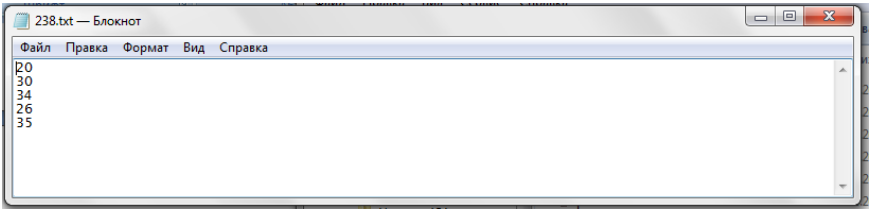

Рис. 9. Файл D:\\238.txt

6.4.2. Запись данных в файл

```
// ConsoleApplication74.cpp : Этот файл содержит функцию "main". Здесь начинается и заканчивается выполнение программы.

#include "pch.h"
#include <iostream>
#include <fstream>
// подключаем файловый поток
using namespace std;
int main()
{
        setlocale(LC_ALL, "RUS");
        int i, n, a;
        fstream f; // назначаем файловому потоку имя f
        f.open("number.txt", ios::app); // параметр app- append, т. е. дописывая в файл.
```

```
        n = 5;
        cout << "Введите 5 чисел:" << endl;
        for (i = 0; i < n; i++)
        {
            cin >> a;
            f << a; // число пишется в файл, последовательно друг за другом
            cout << a; // Обратить внимание, где появляется введенное число!
            // аналогия!
        }
        f.close(); // обязательно закрываем файл.

        return 0;
}
```

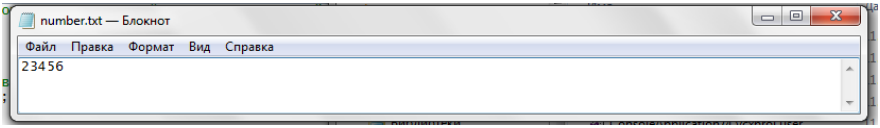

Рис. 10. Результат записи в файл 5 чисел подряд, начиная с 2

6.4.3 Пример записи/чтения строки в/из файла

```
// ConsoleApplication66.cpp : Этот файл содержит функцию "main". Здесь начинается и заканчивается выполнение программы.
//

#include "pch.h"
#include <iostream>


//Чтение из файла (запись и создание)

#include<iostream>
```

```
#include<string>
#include<fstream>

using namespace std;

int main()
{
	setlocale(LC_ALL, "Russian");
	cout << endl;
	cout << "Чтение из файла (запись и создание)" << endl << endl;

	string answer;

	cout << "Создать файл и записать в него текст? (y/n)" << endl;
	cin >> answer;
	if (answer == "y") {
		string txt; // строка до пробела
		cout << "Введите текст: ";
		cin >> txt; // два слова ввести не получится

		fstream fout("numbers.txt"); // назначаем поток для выво-
да в файл
		fout << txt;
		fout.close();
		cout << "Текст записан в файл  numbers.txt  ";
	}
	else
	{
		if (answer == "n") {
			char buff[100];  // буфер для строки
			fstream  fin("numbers.txt");  //  назначаем  поток
для ввода из файла
```

```
                fin.getline(buff, 100); // читаем строку
                cout << "Текст из файла: " << buff;
            }
            else { cout << "Некорректный ввод"; }
        }
}
```

Все указанные примеры совместимы между собой и могут быть использованы для решения задач в соответствии с требованиями к оформлению лабораторных работ. Рекомендуется оформлять отдельные логически не связанные блоки программы в виде процедур или функций для использования их в решении дальнейших задач.

7. ТРЕБОВАНИЯ К ЗАЩИТЕ ЛАБОРАТОРНОЙ РАБОТЫ

Защита лабораторной работы состоит из демонстрации работы программы в соответствии с указанными ранее требованиями и правильно составленного отчета в формате MS Word. После защиты работы отчет сдается преподавателю в электронном виде в формате Word. В случае искажений форм рисунков отчета дополнительно предоставляется копия отчета в формате PDF.

СПИСОК ЛИТЕРАТУРЫ

1. Васильев, А. Н. Самоучитель C++ с примерами и задачами / А. Н. Васильев; под ред. Финкова М. В. − 6-е изд. (перераб. и обновленное) [Книга + виртуальный CD]. − СПб.: Наука и Техника, 2019. − 480 с.

2. ГОСТ 19.701-90. Единая система программной документации. Схемы алгоритмов, программ, данных и систем: межгосударственный стандарт: издание официальное: утвержден и введен в действие Постановлением Государственного комитета СССР по управлению качеством продукции и стандартам от 26.12.90 №3294: дата введения 1992-01-01 / разработан Государственным комитетом СССР по вычислительной технике и информатике. – Москва: Стандартинформ, 2010. – с. 137.

3. Культин, Н. C/C++ в задачах и примерах / Н. Культин. – Санкт-Петербург: БХВ-Петербург, 2005. – 288 с.

4. Редактор MS Visual Studio. – URL: https://learn.microsoft.com/ru-ru/visualstudio/windows/?view=vs-2022.

Учебное издание

Гайсин Марс Марсельевич

МЕТОДИКА ВЫПОЛНЕНИЯ ЛАБОРАТОРНЫХ РАБОТ ПО ДИСЦИПЛИНЕ «ЯЗЫКИ ПРОГРАММИРОВАНИЯ»

Методические рекомендации

Редактор И.А. Бусоргина
Компьютерная верстка: Т.В. Опарина

Издательский центр «Удмуртский университет»
426034, Ижевск, ул. Ломоносова, 4Б, каб. 021
Тел.: + 7 (3412) 916-364, E-mail: editorial@udsu.ru