Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Владимирский государственный университет
имени Александра Григорьевича и Николая Григорьевича Столетовых»

Кафедра бизнес-информатики

БАЗОВЫЕ СРЕДСТВА ЯЗЫКА С++

Методические указания к лабораторным работам по дисциплине «Программирование»

Составитель
К. О. БОЧЕНИНА

Владимир 2013

УДК 004.432.2
ББК 32.973-018
Б17

Рецензент
Кандидат технических наук,
доцент кафедры управления и информатики
в технических и экономических системах
Владимирского государственного университета
имени Александра Григорьевича и Николая Григорьевича Столетовых
Д. А. Градусов

Печатается по решению редакционно-издательского совета ВлГУ

Базовые средства языка C++ : метод. указания к лаб. работам
Б17 по дисциплине «Программирование» / Владим. гос. ун-т им. А. Г. и Н. Г. Столетовых ; сост. К. О. Боченина. – Владимир : Изд-во ВлГУ, 2013. – 32 с.

Содержат лабораторные работы по первой части двухсеместрового курса дисциплины «Программирование».

Предназначены для студентов первого-второго курсов очной формы обучения направления 080500 – Бизнес-информатика.

Рекомендованы для формирования профессиональных компетенций в соответствии с ФГОС 3-го поколения.

Ил. 2. Табл. 2. Библиогр.: 5 назв.

УДК 004.432.2
ББК 32.973-018

Лабораторная работа № 1. Составление блок-схемы алгоритма

Теоретическая часть

Алгоритм – точный набор инструкций, описывающих порядок действий исполнителя для достижения необходимого результата (например, решения задачи) за конечное время.

Каждая такая инструкция называется **командой**. Команды алгоритма выполняются последовательно в заданном порядке, причем исполнитель не может перейти к выполнению следующей команды, не завершив предыдущую.

Форма записи, состав и количество операций алгоритма зависят от того, кто будет исполнителем этого алгоритма. Если задача решается с помощью ЭВМ, алгоритм решения этой задачи должен быть записан в понятной для машины форме, то есть в виде программы. Всякий алгоритм может быть:

- записан на естественном языке;

Пример 1. Вычислить факториал числа n ($f = n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot n$).

Описание алгоритма:

1. Полагаем $f = 1, i = 1$.

2. Проверяем, не превышает ли i числа n.

3. Если $i <= n$, полагаем $f = f \cdot i$, увеличиваем i на 1 и переходим к п. 2. Если $i > n$, выводим результат и прекращаем работу алгоритма.

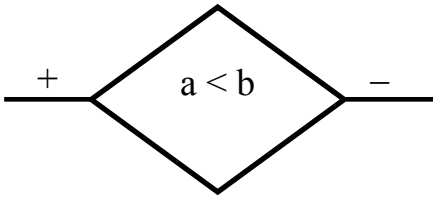
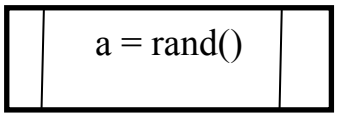
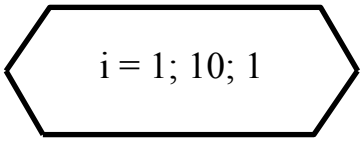
- изображен в виде блок-схемы;

- записан на алгоритмическом языке.

Блок-схема – графическое представление алгоритма. Каждый пункт алгоритма отображается на схеме некоторой геометрической фигурой (блоком) и дополняется элементами словесной записи (табл. 1).

Блоки на схемах соединяются линиями потоков информации. Количество входящих линий для блока не ограничено. Выходящая линия должна быть одна (исключение составляет логический блок).

Табл. 1. Основные элементы блок-схем

Символ	Наименование	Содержание
	Блок вычислений	Вычислительные действия или последовательность действий
	Логический блок	Выбор направления выполнения алгоритма в зависимости от некоторого условия
	Блок ввода-вывода данных	Ввод (вывод) данных
	Начало (конец)	Начало или конец алгоритма
	Подпрограмма	Вычисление по стандартной программе или подпрограмме
	Блок модификации	Используется для организации циклических конструкций с параметром

Базовые структуры алгоритмов – это определенный набор блоков и стандартных способов их соединения для выполнения типичных последовательностей действий. К основным структурам относятся **линейные** (рис. 1, *а*), **разветвляющиеся** (рис. 1, *б*), **циклические** (рис. 1, *в*; 1, *г*).

Рис. 1, *а*. Линейная структура алгоритма

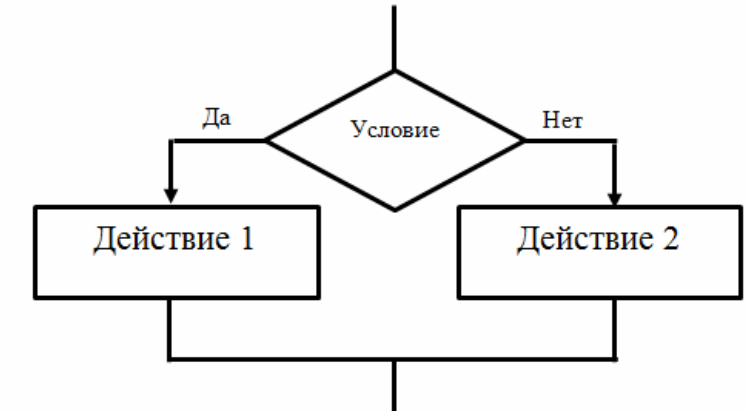


Рис. 1, *б*. Разветвляющаяся структура алгоритма

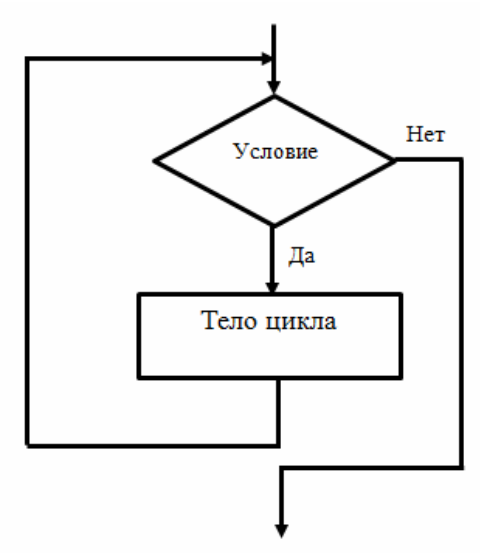


Рис. 1, *в*. Цикл с предусловием

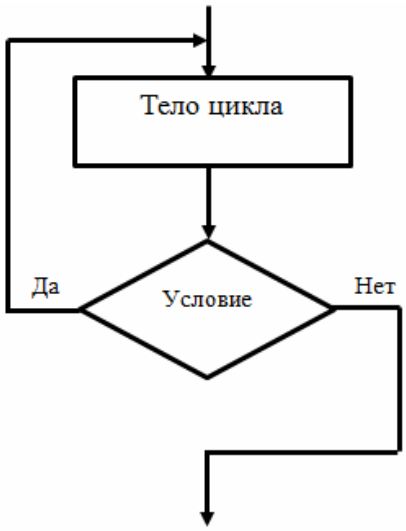


Рис. 1, *г*. Цикл с постусловием

Пример 2. Построить блок-схему вычисления факториала числа *n* (рис. 2, *а*; 2, *б*).

Рис. 2, *а* демонстрирует блок-схему, составленную в соответствии со словесным описанием алгоритма из примера 1, рис. 2, *б* – использование блока модификации. Запись i := 1; n; 1 обозначает буквально «i меняется от 1 до n с шагом 1», то есть цикл будет выполнен ровно n раз для i, принимающего значения от 1 до n. Каждое выполнение цикла называется *итерацией*, переменная i называется *параметром (счетчиком) цикла*.

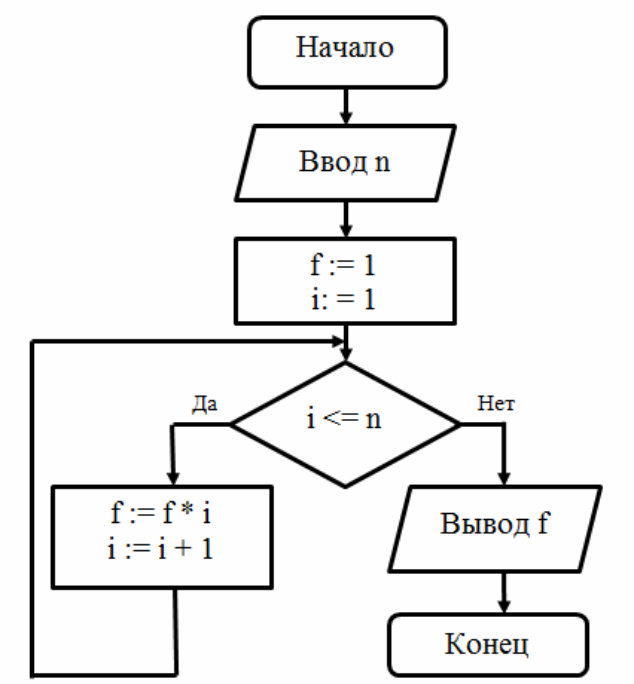


Рис. 2, *а*. Блок-схема вычисления факториала с использованием цикла с предусловием

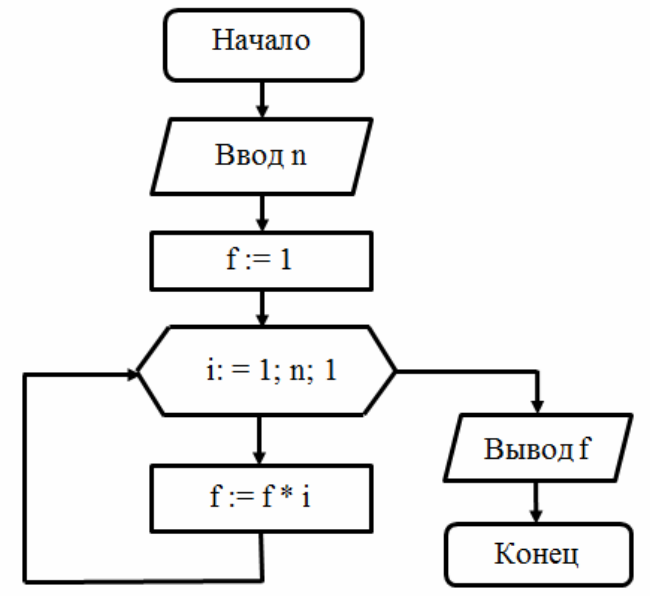


Рис. 2, *б*. Блок-схема вычисления факториала с использованием блока модификации

Задание к работе

Построить блок-схему алгоритма в соответствии со своим вариантом.

1. Алгоритм определения максимального числа в последовательности из n элементов.

2. Алгоритм получения произведения n произвольных чисел.

3. Алгоритм вычисления суммы квадратов первых n чисел натурального ряда.

4. Алгоритм вычисления суммы квадратов первых n четных чисел.

5. Алгоритм вычисления суммы четных чисел, находящихся в диапазоне от 1 до n.

6. Алгоритм решения квадратного уравнения.

7. Алгоритм вычисления площади треугольника по формуле Герона ($S = \sqrt{p(p-a)(p-b)(p-c)}$, где p – полупериметр, a, b, c – стороны).

8. Алгоритм проверки треугольника на вырожденность (треугольник считается вырожденным, если сумма длин любых двух сторон больше длины третьей стороны).

9. Дано натуральное число n. Вычислить $\dfrac{n^{k+1} - 2k}{n + 4k}$.

10. Вычислить значение функции $y = \begin{cases} \sqrt{x}, x \le 0 \\ 0, x = 0 \\ x^2, x \ge 0 \end{cases}$.

11. Алгоритм расчета суммы $s1$ и $s2$ положительных и отрицательных n случайных чисел в диапазоне от -100 до 100. **Примечание:** случайные числа генерирует подпрограмма rand().

12. Распечатать произведения чисел A и B, изменяющихся от 17 и -25 с шагом -3 и 5 соответственно до тех пор, пока это произведение – отрицательное число.

13. Сколько слагаемых должно быть в сумме 1 + ½ + 1/3 + …+ 1/N, чтобы сумма оказалась больше 5?

14. Урожай яблок в 1990 году составил 20 тонн. Далее каждые два года урожай уменьшался на 20 %:

а) начиная с какого года, будет собрано менее 5 т?

б) в каком году суммарный урожай яблок превысит 90 т?

15. Составьте таблицу значений функции $y = 4x^2 - 5x + 10$ на отрезке [-9; 9] с шагом 3.

Содержание отчета

Цель работы, краткая теория, постановка задачи, блок-схема алгоритма, вывод.

Вопросы для самопроверки

1. Что такое алгоритм?
2. Какие существуют формы представления алгоритмов? Кратко охарактеризуйте их.
3. С помощью какой фигуры изображается этап вычисления? Проверки условия? Вывода данных?
4. В чем отличие цикла с предусловием от цикла с постусловием?
5. Составьте блок-схему алгоритма из примера 2 с использованием цикла с постусловием.
6. Являются ли цикл с постусловием и блок модификации взаимозаменяемыми конструкциями? Обоснуйте ответ.

Лабораторная работа № 2. Типы данных, операции и управляющие конструкции языка C++. Средства ввода-вывода

Теоретическая часть

Переменные и константы

Переменная – это величина, значение которой может быть изменено в ходе исполнения алгоритма (например, счетчик цикла). Так как данные, с которыми работает программа, хранятся во время ее исполнения в оперативной памяти, **переменная** представляет собой поименованный участок памяти, предназначенный для хранения текущего значения некоторой величины.

С понятием переменной связаны следующие характеристики: 1) *имя (идентификатор)* – обозначение переменной в программе; 2) *тип данных* – множество допустимых значений переменной (так же определяет возможные операции, применимые к ней, и объем памяти, отводимый под переменную); 3) *значение*.

В языке C++ *имена переменных* составляются из букв и цифр, первым символом должна быть буква или символ подчеркивания _. C++ – регистрозависимый язык, поэтому имена example и Example будут восприняты компилятором как два разных имени. Ключевые слова (такие, как if, while, float) зарезервированы и не могут быть использованы в качестве имен.

Различают переменные следующих простых типов: целые (int, long, short), вещественные (float, double) и символьные (char); а также их расширения (unsinged, singed – знаковые и беззнаковые). Имеется также тип с отсутствующим значением (void) и логический тип (bool).

Перед использованием переменной ее необходимо *объявить* в вашей программе, то есть указать компилятору тип переменной и имя, по которому вы будете к ней обращаться. Объявление переменной заканчивается точкой с запятой. Например:

```
int a; // объявляем переменную с именем a и типом int
```

Можно объявлять несколько переменных в одной строке, например:

```
float x, y, z; // объявляем 3 переменных типа float
```

Объявление переменной можно совместить с ее *инициализацией* – присвоением начального значения:

```
char c = 'C'; // объявляем переменную с типа char и присваиваем
      // ей значение 'C'
```

К любой переменной в объявлении может быть применен квалификатор const для указания того, что ее значение далее не будет изменяться.

```
const double pi = 3.1415926;
```

Операции языка C++.

Язык C++, как и другие языки программирования, включает в себя набор *операций*. Каждая операция обозначается определенным знаком или комбинацией знаков. Введенное обозначение операции называется *оператором* языка программирования. Оператор сообщает компилятору, какие действия необходимо совершить над *операндами*.

Основные операции языка C++ в порядке убывания приоритета приведены в табл. 2.

Табл. 2. Основные операции языка C++

Приоритет	Оператор	Описание, примеры
1	++ – –	Постфиксный инкремент (декремент). Переменная сначала используется в выражении, а затем увеличивается (уменьшается) на 1. Пример: int a = 1, b = a++; // b = 1, a = 2;
	( )	Вызов функции. Пример: int x = 25, a = sqrt(x); // a = 5
	[ ]	Индексация элементов массива (однотипной последовательности значений). Пример: int a[5]; // массив из 5 элементов целого типа
2	++ – –	Префиксный инкремент (декремент). Переменная увеличивается (уменьшается) на 1, а затем используется в выражении. Пример: int a = 1, b = ++a; // b = 2, a = 2;
	+ –	Унарный плюс (минус). Операция эквивалентна умножению операнда на +1 (–1). Пример: float a = 5.13, b = –a; // b = -5.13
	!	Логическое отрицание. Отрицанием нулевого значения является 1, отрицанием любого ненулевого значения является 0. Пример: int a = 5, b = !a; // b = 0
	~	Побитовое отрицание. Производится поразрядное инвертирование битов. Пример: unsigned char a = 5, b = ~a; // b = 250, т. к. ~(0000 0101) = 1111 1010 (2 СС) = 250 (10 СС)
	& *	Операция & позволяет получать адрес объекта по его имени. Операция * является обратной к & и позволяет выполнять *разыменование*, то есть получение значения, хранящегося по адресу
	sizeof	Определение размера в байтах. Например, для определения объема памяти, занимаемой переменной типа int, можно написать sizeof(int)
	new delete	Операции динамического выделения (освобождения) памяти

Продолжение табл. 2

Приоритет	Оператор	Описание, примеры
3	* /	Умножение и деление. Пример: int a = 10, b = a/2, c = a*3; // b = 5, c = 30
	%	Операция получения целочисленного остатка от деления. Пример: int a = 15, b = a%6; // b = 3
4	+ –	Сложение и вычитание. Пример: int a = 5, b = a++ – 3; // a = 4, b = 2
5	<< >>	Побитовый сдвиг влево (вправо). Пример: 10001111<<4 = 11110000; 10001111>>2 = 00100011
6	< <= > >=	Операторы «меньше», «меньше или равно», «больше», «больше или равно»
7	= = !=	Операторы «равно», «не равно»
8	&	Побитовое «И» (AND, конъюнкция). Результат равен 1 в том и только в том случае, когда оба бита равны 1. Пример: 10011001 & 10101010 = 10001000
9	^	Побитовое «Исключающее ИЛИ» (XOR, сложение по модулю 2). Результат равен 1, если сравниваемые биты разные. Пример: 10011001 ^ 10101010 = 00110011
10	|	Побитовое «ИЛИ» (OR, дизъюнкция). Результат равен 1, если хотя бы один из битов равен 1. Пример: 10011001 | 10101010 = 10111011
11	&&	Логическое «И» (AND, конъюнкция). Результат – истина в том случае, когда истинны оба операнда. Пример: int a = 5, b = 0, c = a&&b; // c = 0
12	| |	Логическое «ИЛИ» (OR, дизъюнкция). Результат – истина в том случае, когда истинен хотя бы один из операндов. Пример: int a = 5, b = 0, c = a| |b; // c = 1
13	?:	Тернарный оператор условия. Синтаксис: *выр1 ? выр2 : выр3*, где *выр2* выполняется в случае, если *выр1* истинно, *выр3* – в противном случае. Пример: int a = 3, b = 1, c = a > b ? a – b : a+b; // c = 2, т. к. a > b

Окончание табл. 2

Приоритет	Оператор	Описание, примеры

14	= += –= *= /= %= <<= >>= &= ^= |=	Операторы присваивания. Выражение вида a+=b можно переписать с помощью простого оператора присваивания как a = a+b. Пример:
		int a = 9; a/=3; // a = 3
		int b = 15; b%=6; // b = 3

15	,	Выполняет роль разделителя, приводит к вычислению выражения слева направо. Пример: int i = 10, j = (i = 12, i+8); // i = 12, j = 20

Средства ввода-вывода языка C++

Ввод-вывод в языке C++ осуществляется с помощью функций, объявленных в заголовочных файлах. Основные функции ввода-вывода задаются в заголовочном файле <stdio.h>. Для подключения данного заголовочного файла необходимо использовать *директиву препроцессора #include <stdio.h>*.

Функция *printf("форматная строка", арг1, арг2, ...)* предназначена для вывода информации на экран. Примеры:

1. Простой вывод строки.

```
printf("Hello world!"); // выводит на экран строку «Hello world!»
```

2. Вывод с переводом строки. В данном примере используется *управляющий символ* или *Esc-последовательность \n.* Он формируется из символа обратной косой черты (backslash) и латинской буквы.

```
printf("Hello world!\n");
```

3. Вывод значений переменных на экран.

В форматной строке для каждой выводимой переменной указывается спецификатор формата (%d – для вывода целых чисел в десятичной форме, %u – для вывода целых чисел в десятичной форме без знака, %o – для вывода целых чисел без знака в восьмеричной форме, %x – для вывода целых чисел без знака в шестнадцатеричной форме, %c – для вывода символов, %f – для вывода вещественных чисел в обычной форме, %e – для вывода вещественных чисел в шестнадцатеричной форме).

При выводе на экран значения переменных из списка аргументов будут подставлены в форматную строку вместо указанных спецификаторов (Esc-последовательность \t используется для табуляции):

```
int a = 5; char c = 'Z';
printf("a=%d\tc=%c",a,c); // будет выведено: a=5      c=Z
```

Спецификаторы формата также можно использовать для контроля ширины поля вывода и количества выводимых знаков после запятой, например (ширина поля вывода – 5 символов, число знаков после запятой – 2):

```
float pi = 3.1415926;
printf("pi=%5.2f",pi); // вывод: pi= 3.14
```

Функция *scanf("форматная строка", &арг1, &арг2, ...)* считывает данные с консоли и сохраняет их в переменных списка аргументов. Обратите внимание, что в списке аргументов указываются на сами переменные, а их адреса. Пример:

```
int a, b;
printf("Введите значения a и b: ");
scanf("%d%d",&a,&b);
```

Рассмотрим также ввод-вывод в C++ с помощью *потоков ввода-вывода*. Для использования потоков необходимо подключить заголовочный файл iostream:

```
#include <iostream>
```

Вывод строки на экран:

```
std::cout << "Hello world!");
```

Чтобы явно не указывать название пространства имен std при каждом использовании cout и cin, после подключения библиотеки iostream следует добавить строку:

```
using namespace std;
```

Вывод значений переменных на экран с переводом строки:

```
int a = 5, b = 3;
cout << "a=" << a <<", b=" << b << endl; // a=5, b=3
```

Считывание значений переменных:

```
int a, b;
cin >> a >> b;
```

Управляющие конструкции языка C++

Управляющие конструкции позволяют организовывать циклы и ветвления в программах.

1. *Фигурные скобки {}* – позволяют объединить несколько простых операторов в составной оператор, или *блок*. Переменные, описанные внут-

ри блока, создаются при входе в блок и уничтожаются при выходе из него. Пример:

```
int x = 5, y = 10;    // объявляем и инициализируем переменные
{
      int temp = x;    // переменная temp является вспомогательной
      x = y;     // и служит для обмена значениями между x и y
      y = temp;
} // после выхода из блока переменная temp будет уничтожена
```

2. *Оператор if (конструкция ветвления)*. Позволяет организовывать разветвляющуюся структуру алгоритма (см. рис. 1, *б*). Имеет две формы:

```
if (условие) действие;
```

Если условие истинно, будет выполнено *действие*.

```
if (условие) действие1;
else действие 2;
```

Если условие истинно, будет выполнено *действие1*, в противном случае будет выполнено *действие2*.

Если по условию должно быть выполнено несколько операторов, можно использовать фигурные скобки для объединения группы операторов в блок.

Оператор if может быть вложенным (если необходима проверка нескольких условий).

Пример 3. Вычислить значение функции $y = \begin{cases} x^2 + 2x, x > 0 \\ 0, x = 0 \\ |x|, x < 0 \end{cases}$.

```
cout << "Введите x: ";
int x, y;
cin >> x;
if (x >= 0) y = x*x + 2*x;
else if (x==0) y = 0;
else y = -x;
```

3. Цикл for. Используется в тех случаях, когда заранее известно число итераций цикла.

Синтаксис: *for (инициализация; условие; инкремент) тело цикла*.

В круглых скобках можно видеть три блока, разделенных точкой с запятой. В блоке *инициализации* задаются начальные значения переменных (как правило, счетчиков цикла). *Условие* проверяется перед началом каждой итерации цикла, если оно истинно, цикл продолжает работу. В блоке,

условно названном *инкремент*, располагаются операторы, которые выполняются после каждой итерации цикла перед проверкой условия (как правило, в этом блоке изменяется значение счетчика цикла).

Пример 4. Подсчет суммы чисел от 1 до 10.

```
int s = 0;
for (int i = 1; i <=10; i++) s = s+i;
```

Если тело цикла состоит из нескольких операторов, их объединяют в блок.

Пример 5. Подсчет суммы чисел от 1 до 10 с выводом накопленной суммы.

```
int s = 0;
for (int i = 1; i <=10; i++)
{
	s = s+i;
	cout << "Накопленная сумма: " << s << endl;
}
```

4. Циклы с предусловием и постусловием.

Синтаксис цикла с предусловием: *while (условие) тело цикла.*

Синтаксис цикла с постусловием: *do тело цикла while (условие).*

Цикл продолжает выполняться до тех пор, пока проверяемое условие истинно. При использовании while условие проверяется в первый раз **перед** первой итерацией цикла, а при использовании do…while – **после** первой итерации.

Задание 1. Переписать примеры 4, 5 с использованием циклов с предусловием и постусловием. Объяснить, всегда ли возможна взаимозамена цикла for и циклом с предусловием и постусловием.

Пример 6. Вывод квадратов четных чисел от 10 до 2 (цикл с предусловием).

```
x = 10;
while (x > 0)
{
	cout << x*x << endl;
	x-=2;
}
```

Пример 7. Вывод квадратов четных чисел от 10 до 2 (цикл с постусловием).

```
x = 10;
do
{
     cout << x*x << endl;
     x -= 2;
} while (x>0);
```

5. Оператор выбор (переключатель) switch. Применяется как альтернатива конструкции ветвления в случае, если переменная может принимать одно значение из некоторого набора, и должны быть выполнены разные ветви кода в зависимости от значения этой переменной.

Синтаксис switch:

```
switch (переменная){
case значение1: операторы1; break;
…
case значение_n: операторы_n; break;
default: операторы; break;
}
```

Пример 8. Вывод текстовых описаний оценок.

```
int c; cin >> c;
switch (c){
case 2: cout << "Неудовлетворительно"; break;
case 3: cout << "Удовлетворительно"; break;
case 4: cout << "Хорошо"; break;
case 5: cout << "Отлично"; break;
default: cout << "Такой оценки нет"; break;
}
```

Операторы блока default выполняются в случае, если в метках case не найдено соответствий для значения проверяемой переменной.

Задание 2. Переписать пример 8 с использованием конструкции ветвления.

Задание к работе

Часть 1. Написать программу, реализующую алгоритм (в соответствии с вариантом) из лабораторной работы № 1.

Часть 2. Выполнить задания:

1. Объявите переменную radius типа float, переменную f типа float со значением 3,14, переменную s – площадь круга, переменную c – длина окружности. Значение radius вводит пользователь.

Рассчитать площадь круга и длину окружности, записать в переменные c и s, вывести результат на экран.

2. Пользователь вводит коэффициенты квадратного уравнения: a, b и c. Рассчитать дискриминант d. Если дискриминант меньше нуля, вывести сообщение о том, что корней нет. Если дискриминант равен нулю, рассчитать и вывести один корень x. Иначе рассчитать корни x1, x2.

3. Подсчитать в цикле квадраты и кубы чисел от -5 до 5, вывести на экран.

4. Пользователь вводит числа с клавиатуры до тех пор, пока не будет введено первое отрицательное число. Вывести на экран сумму чисел, введенных пользователем: а) без учета отрицательного числа; б) с учетом отрицательного числа.

5. Пользователь вводит два числа: x и y и номер действия: 1 – подсчитать сумму, 2 – подсчитать разность. Если введен некорректный номер действия, выводить предупреждающее сообщение. Иначе рассчитать результат и вывести на экран.

Часть 3

Реализовать на C++ калькулятор с возможностью выполнения четырех арифметических действий. У пользователя запрашивать аргументы и знак операции (символ). Программу «зациклить» – после вывода результата предлагать пользователю произвести вычисления еще раз. Выход из программы производить по нажатию клавиши q. Осуществить проверки: 1) деления на ноль (в этом случае выводить предупреждающее сообщение и запрашивать делитель еще раз); 2) корректности ввода знака операции (в этом случае просить ввести корректный знак операции).

Пример работы с программой:

```
Введите первый операнд: 5
Введите второй операнд: 0
Введите знак операции: /
На ноль делить нельзя!
Введите второй операнд: 2
Результат: 2,5
Нажмите любую клавишу для продолжения (выход - q)…
Введите первый операнд: 3
Введите второй операнд: 2
Введите знак операции: %
Некорректно введен знак операции!
Введите знак операции: ?
Некорректно введен знак операции!
```

```
Введите знак операции: +
Результат: 5
Нажмите любую клавишу для продолжения (выход - q)…
```

Содержание отчета

Титульный лист, цель работы, задание к работе, краткая теория, выполнение работы (листинг программы), результаты работы, вывод.

Вопросы для самопроверки

1. Что такое переменная? Объявление переменной? Инициализация переменной? Имя переменной? Тип данных переменной?

2. Перечислите базовые типы данных языка C++.

3. Перечислите основные операции языка C++ в порядке убывания приоритетов.

4. Опишите синтаксис и приведите примеры использования функций printf(), scanf().

5. Для чего используются спецификаторы формата? Перечислите основные спецификаторы формата.

6. Каким образом можно контролировать ширину поля для вывода числа? Число знаков после запятой при выводе вещественной переменной?

7. Опишите синтаксис и приведите примеры использования функций cin, cout.

8. Что такое составной оператор (блок)? Каким образом операторы можно объединить в блок?

9. Опишите и приведите примеры использования конструкции ветвления.

10. Что такое тело цикла? Итерация цикла? Счетчик цикла?

11. Опишите и приведите примеры использования цикла for, циклов с предусловием и постусловием, оператора выбора switch.

Лабораторная работа № 3. Массивы в языке C++. Сортировка массивов

Теоретическая часть

Массив – это именованная последовательность однотипных элементов. Массив представляет собой структуру данных, которая позволяет хранить несколько значений в одной переменной.

При объявлении массива в программе вам необходимо указать тип хранимых элементов, имя массива и его размерность, то есть количество хранимых элементов. Размерность массива представляет собой целочисленную константу.

Синтаксис объявления одномерного массива:

тип имя_массива[размерность];

Пример 9. Объявление массива из 5 элементов типа int.

```
int arr[5];
```

Для обращения к определенным значениям, хранящимся в массиве, необходимо использовать значение *индекса*, которое указывает на требуемый элемент. *Индексация* массивов в C++ начинается с нуля, то есть массив arr из примера 9 имеет 5 элементов: arr[0], arr[1], arr[2], arr[3], arr[4].

Пример 10. Присваивание значений элементам массива.

```
float f[3];
f[0] = 1.11; f[1] = 1.13; f[2]=5.15;
```

Пример 11. Ввод элементов массива пользователем.

```
const int N = 5; // размерность массива
int arr[N];
for (int i = 0; i < N; i++)
{
      cout << "Введите a[" << i << "]: ";
      cin >> a[i];
}
```

Массивы, как и переменные, в C++ возможно инициализировать при объявлении, например:

```
int a[5] = {1, 2, 3, 4, 5}; // a[0]=1, a[1]=2, a[2]=3, a[3]=4, a[4]=5
```

Если мы укажем в списке инициализации массива а не пять значений, а три, остальные элементы будут проинициализированы нулями:

```
int a[5] = {1, 2, 3}; // a[0]=1, a[1]=2, a[2]=3, a[3]=0, a[4]=0
```

При объявлении массива допустимо не указывать его размерность, но только в том случае, когда это объявление производится совместно с инициализацией (в этом случае размерность будет определена исходя из числа элементов в списке инициализации), например:

```
int a[] = {1, 2, 3, 4, 5};
```

Возможно создание многомерных массивов (например, если необходимо хранение матрицы, мы можем создать двумерный массив).

Синтаксис объявления n-мерного массива в C++:

тип имя_массива[размерность1] [размерность2]... [размерность_n];

Пример 12. Объявить и проинициализировать массив для хранения единичной матрицы размера 3×3.

```
int a[3][3]={1, 0, 0, 0, 1, 0, 0, 0, 1}; // a[0][0] = 1 …, a[2][2]=1
```

Пример 13. Ввод элементов двумерного массива пользователем.

```
const int N = 3; // размерность массива
int arr[N];
for (int i = 0; i < N; i++)
     for (int j = 0; j < N; j++)
{
     cout << "Введите a[" << i << "][" << j << "]: ";
     cin >> a[i][j];
}
```

Сортировка массивов

Сортировкой, или *упорядочением* массива, называется расположение его элементов по возрастанию (или убыванию). Если не все элементы различны, то надо говорить о *неубывающем* (или *невозрастающем*) порядке.

Метод «пузырька»

Алгоритм состоит в повторяющихся проходах по сортируемому массиву. За каждый проход элементы последовательно сравниваются попарно и, если порядок в паре неверный, выполняется обмен элементов. Проходы по массиву повторяются до тех пор, пока на очередном проходе не окажется, что обмены больше не нужны, что означает – массив отсортирован. При проходе алгоритма элемент, стоящий не на своём месте, «всплывает» до нужной позиции, как пузырёк в воде, отсюда и название алгоритма.

Возьмём массив с числами «5 1 4 2 8» и отсортируем значения по возрастанию, используя сортировку пузырьком. Выделены те элементы, которые сравниваются на данном этапе.

Первый проход:

(5 1 4 2 8) (1 5 4 2 8), Здесь алгоритм сравнивает два первых элемента и меняет их местами.

(1 5 4 2 8) (1 4 5 2 8), меняет местами, так как 5 > 4.

(1 4 5 2 8) (1 4 2 5 8), меняет местами, так как 5 > 2.

(1 4 2 5 8) (1 4 2 5 8), теперь, ввиду того, что элементы стоят на своих местах (8 > 5), алгоритм не меняет их местами.

Второй проход:

(1 4 2 5 8) (1 4 2 5 8)

(1 4 2 5 8) (1 2 4 5 8), меняет местами, так как 4 > 2.

(1 2 4 5 8) (1 2 4 5 8)

(1 2 4 5 8) (1 2 4 5 8)

Теперь массив полностью отсортирован, но алгоритм не знает, так ли это. Поэтому ему необходимо сделать полный проход и определить, что перестановок элементов не было.

Третий проход:

(1 2 4 5 8) (1 2 4 5 8)

(1 2 4 5 8) (1 2 4 5 8)

(1 2 4 5 8) (1 2 4 5 8)

(1 2 4 5 8) (1 2 4 5 8)

Теперь массив отсортирован, и алгоритм может быть завершён.

Сортировка вставкой

На каждом шаге алгоритма мы выбираем один из элементов входных данных и вставляем его на нужную позицию в уже отсортированном списке до тех пор, пока набор входных данных не будет исчерпан. Метод выбора очередного элемента из исходного массива произволен; может использоваться практически любой алгоритм выбора. Обычно (и с целью получения устойчивого алгоритма сортировки) элементы вставляются по порядку их появления во входном массиве. Приведенный ниже алгоритм использует именно эту стратегию выбора. На j-м этапе мы вставляем j-й элемент M[j] в нужную позицию среди элементов M[1], M[2], ... , M[j-1], которые уже упорядочены. После этой вставки первые j элементов массива M будут упорядоченны.

Возьмём массив с числами «5 1 4 2 8» и отсортируем значения по возрастанию, используя сортировку вставкой. Сортировка начинается со второго элемента массива.

j = 2: (5 1 4 2 8) (1 5 4 2 8)

j = 3: (1 5 4 2 8) (1 4 5 2 8)

j = 4: (1 4 5 2 8) (1 2 4 5 8)

j = 5: (1 4 5 2 8) (1 2 4 5 8)

Сортировка выбором

Идея сортировки с помощью выбора не сложнее двух предыдущих. На j-м этапе выбирается элемент наименьший среди M[j], M[j+1], ... , M[N] и меняется местами с элементом M[j]. В результате после j-го этапа все элементы M[j], M[j+1], ... , M[N] будут упорядоченны.

Возьмём массив с числами «5 1 4 2 8» и отсортируем значения по возрастанию, используя сортировку выбором.

(5 1 4 2 8) (1 5 4 2 8) – ищем минимальный элемент массива, ставим его на первое место.

(1 5 4 2 8) (1 2 4 5 8) – ищем минимальный среди всех элементов массива, кроме первого, ставим его на второе место.

(1 2 4 5 8) (1 2 4 5 8) – остальные этапы аналогично, в этом примере изменений не происходит.

Задание к работе

Часть 1

1. В массиве arr1 хранятся числа от 1 до 5, элементы массива arr2[5] вводит пользователь. Подсчитать произведения соответствующих элементов arr1 и arr2, записать в массив arr3 и вывести на экран.

2. Дана матрица размером 3×3. Элементы матрицы вводит пользователь. Рассчитать и вывести на экран определитель матрицы. Осуществить проверку матрицы на вырожденность. Вывести на экран транспонированную матрицу (строки заменены на столбцы).

3. Дана матрица размером 3×3. Подсчитать и вывести на экран сумму и произведение элементов матрицы по строкам и по столбцам.

Часть 2

Реализовать алгоритмы сортировки «пузырьком», методом вставки и методом выбора на языке C++. Подсчитать число операций сравнения элементов по каждому из этих алгоритмов для массива a[10] = {2,7,9,5,4,0,3,8,6,1}.

Вопросы для самопроверки

1. Дайте определение массива.

2. Опишите синтаксис объявления одномерного/многомерного массива в языке C++.

3. Может ли массив содержать в себе разнотипные элементы?

4. Что такое тип массива? Размерность массива? Индекс элемента массива?

5. Какой индекс в C++ присваивается первому элементу массива?

6. Что такое инициализация массива?

7. Перечислите способы инициализации массивов.

8. Что такое сортировка массива?

9. Перечислите известные вам алгоритмы сортировки массивов.

Содержание отчета

Титульный лист, цель работы, задание к работе, краткая теория, выполнение работы (листинг программы), результаты работы, вывод.

Лабораторная работа № 4. Символы и символьные строки в языке С++

Теоретическая часть

Символы в С++

Для представления символов в С++ предназначен тип char. Его размер составляет 1 байт, что позволяет хранить в переменной типа unsigned char числа от 0 до 255, а в signed char – от -128 до 127. Каждому символу сопоставлено определенное число из приведенного диапазона – *ASCII*-код символа.

Для ввода/вывода символов с помощью функций printf(), scanf() используется спецификатор %c. Если необходимо вывести код символа, используется спецификатор %d.

Пример 14. Вывести на экран таблицу ASCII-кодов английских заглавных букв (код буквы A – 65, Z – 90).

```
unsigned char a;
for (a = 65; a <= 90; a++)
{
      printf("Символ: %c",a);
      printf("\tКод символа: %d\n",a);
}
```

Ввод-вывод символов также можно производить с помощью функций getch(), putch() (для их использования необходимо подключить заголовочный файл conio.h).

```
char ch = getch(); // считываем символ в переменную ch
putch(ch); // выводим считанный символ на экран
```

Символьные строки в С++

Символьная строка – это массив элементов типа char. Размер массива определяется программистом в зависимости от того, что будет храниться в строке. Символьная строка в С++ обязательно должна заканчиваться NULL-символом (записывается как '\0' или NULL). NULL-символ также называют *признаком конца строки*.

Инициализировать символьную строку можно во время ее объявления, например:

```
char name[5]={'М','а','р','к','\0'};
char name[5]="Марк"; // упрощенный синтаксис для строк
```

Пример 15. Записать в строку английский алфавит и вывести на экран.

```
char alphabet [26]; // 25 букв плюс NULL-символ;
int index = 0;
```

```
for (char letter = 'A'; letter <= 'Z'; letter++, index++)
     alphabet[index] = letter;
alphabet[index] = '\0';
cout << "Буквы " << alphabet;
```

Обратите внимание, что при посимвольном заполнении строки контроль за включением NULL-символа в конце строки остается за программистом.

Для ввода/вывода строк с помощью printf(), scanf() используется спецификатор формата %s.

Функции работы со строками в C++.

Для использования функций работы со строками необходимо подключить заголовочный файл "string.h".

Кратко опишем и приведем примеры использования наиболее часто употребляемых функций библиотеки <string>.

strupr(строка) – преобразует строку в верхний регистр;

strlwr(строка) – преобразует строку в нижний регистр;

strlen(строка) – возвращает длину строки;

strset(строка,символ) – заменяет все символы строки на указанный символ;

strrev(строка) – меняет порядок символов в строке на противоположный.

Пример 16. Продемонстрировать работу вышеприведенных функций работы со строками.

```
char title[]="C++ strings";
cout << strupr(title) << endl; // вывод: C++ STRINGS
cout << strlwr(title) << endl; // вывод: c++ strings
cout << strlen(title) << endl; // вывод: 11
cout << strrev(title) << endl; // вывод: sgnirts ++c
cout << strset(title,'x') << endl; // вывод: xxxxxxxxxx
```

strcpy(строка1, строка2) – копирует строку s2 в строку s1. Возвращает значение s1. Массив символов s1 должен быть достаточно большим, чтобы хранить строку и ее завершающий нулевой символ, который также копируется.

char str1[5],str2[3]="No";

strcpy(str1,str2); // str1="No", str2="No";

strcmp(строка1, строка2) – сравнивает строки s1 и s2 (по ASCII-кодам). Функция возвращает значение 0, если строки s1 и s2 равны, значе-

ние меньше ноля, если строка s1 меньше s2, и значение больше ноля, если s1 больше s2.

Пример 17. Сравнить строки «Yes» и «No».

```
char str1[]="Yes",str2[]="No";
int i = strcmp(str1,str2);
if (i < 0) cout << "str1 < str2";
else if (i == 0) cout << "str1 = str2";
else cout << "str1 > str2";
```

В примере 17 str1>str2, так как код символа 'Y' больше кода символа 'N'.

strcat(строка1, строка2) – добавляет строку s2 к строке s1. Первый символ строки s2 записывается поверх NULL-символа строки s1. Возвращает s1.

```
char st1[25] = "День";
cout << strcat(st1, " добрый!"); // вывод: День добрый!
```

strncpy(s1, s2, n) – копирует не более n символов строки s2 в массив символов s1. Возвращает s1.

strncmp(s1, s2, n) – сравнивает до n символов строки s1 со строкой s2.

strncat(s1, s2, n) – присоединяет первые n символов строки s2 в строку s1. Возвращает s1.

strchr(s, c) – проверяет строку s на содержание символа, хранящегося в *c*. Результатом функции является адрес первого вхождения символа *c* в строку s.

```
char str[20] = "ABCDEXYZ";
cout << strchr(str, 'X'); // на экране будет XYZ
```

или

```
char str[20] = "ABCDEXYZ";
if (strchr(str, 'q') == NULL) cout << "Нет такого символа!";
```

Пример 18. Подсчет числа вхождений буквы в строку.

```
char ch;
cout << "Введите символ: ";
cin >> ch;
printf("Введите строку: ");
char str[100];
scanf("%s", str);
int number = 0;
for (int i = 0; i < strlen(str); i++)
     if (str[i]==ch) ++number;
cout << number;
```

Пример 19. Записать в строку str1 строку str без вхождения символа ch.

```
// считывание str и ch
int i = 0, j = 0;
while ( str[i] != '\0' )
if ( str[i++] != ch ) str1[j++] = str[i-1];
str1[j] = '\0';
cout << "Результирующая строка: " << str1;
```

Пример 20. Вывести те буквы str1, которых нет в str2.

```
char str1[100] = "кормчий";
char str2[100] = "крыло";
cout << "Строка: " << str1 << endl;
int i = 0, j = 0;
bool isInString = false;
while ( str2[j] != '\0' )
{
      isInString = false;
      while ( str1[i] != '\0' )
      {
            if ( str1[i] == str2[j] )
            {
                  isInString = true;
                  break;
            }
            i++;
      }
      if ( isInString == false )
            cout << "Нет буквы " << str2[j] << endl;
      i = 0;
      j++;
}
```

Задание к работе

Часть 1. Выполняется без использования функций библиотеки <string>.

1. Пользователь вводит строку str1. Записать в строку str2 и вывести на экран str1 в обратном порядке.

2. Дана строка str1, в которой хранится произвольное предложение. Подсчитать и вывести на экран: а) число слов в предложении; б) число букв в самом длинном слове предложения; в) число знаков препинания в предложении.

3. Дана строка str1. Поменять местами первый и последний символы строки.

Часть 2. Разрешается использовать функции библиотеки <string>.

1. В строке str записан некоторый текст из нескольких предложений. Вывести на экран: а) все слова с удвоенной буквой «н»; б) все слова, начинающиеся с заглавной буквы.

2. Пусть X, Y – слова одинаковой длины. Подсчитать, сколько букв нужно исправить в слове X, чтобы получилось слово Y. X и Y вводит пользователь. Если он ввел слова разной длины, выдавать предупреждающее сообщение и предложить ввести слова еще раз.

3. Написать программу, позволяющую выяснить, можно ли из букв слова X составить слово Y. X и Y вводит пользователь.

Вопросы для самопроверки

1. Какой тип данных используется в C++ для хранения символов?

2. Каким образом можно получить ASCII-код символа? Вывести символ по его коду?

3. Перечислите способы ввода/вывода символов в C++.

4. Что такое символьная строка?

5. Зачем применяется NULL-символ?

6. Перечислите способы инициализации строк в C++.

7. Перечислите, опишите и приведите примеры использования функций работы со строками в C++.

Содержание отчета

Титульный лист, цель работы, задание к работе, краткая теория, выполнение работы (листинг программы), результаты работы, вывод.

Лабораторная работа № 5. Структуры, объединения, перечисления

Теоретическая часть

Структуры

В лабораторной работе № 3 мы познакомились с понятием массива как совокупности однотипных элементов. При программировании часто возникает потребность в наборе данных, который содержит в себе разнотипные элементы (например, если необходимо хранить имя, возраст, пол, заработную плату человека).

Структура – это набор данных, содержащих разнотипные элементы. Переменные, содержащиеся в структуре, называются *членами*, *полями,* или

элементами структуры. Элементы структуры могут представлять собой переменные, массивы или другие структуры.

Синтаксис описания структуры в языке C++:

```
struct имя {
тип1 имя_поля1;
тип2  имя_поля2;
…
тип_n имя_поля_n;
};
```

Обратите внимание, что описание структуры заканчивается точкой с запятой.

Пример 21. Описать структуру для хранения информации об автомобиле.

```
struct car {
char brand[30]; // марка автомобиля
char color[15]; // цвет автомобиля
float engine_capacity; // объем двигателя
int horsepower; // число лошадиных сил
int year; // год выпуска
};
```

Описывая структуру, мы описываем не переменную, а новый (пользовательский) тип данных (в примере 21 – тип данных car).

Для того чтобы пользоваться преимуществами пользовательского типа данных, в программе необходимо *объявить переменную* такого типа. Доступ к полям структуры осуществляется с помощью точки.

Пример 22. Объявить переменную типа car, проинициализировать ее.

```
car mycar;
strcpy(mycar.brand, "Mersedes");
strcpy(mycar.color, "Red");
mycar.engine_capacity = 2.7;
mycar.horsepower = 660;
mycar.year = 2005;
```

Объявить переменную типа структуры можно и при описании структуры:

```
struct car {
char brand[30]; // марка автомобиля
...
int year; // год выпуска
} car1, car2;
```

Структуры могут быть объединены в массивы структур. Делается это аналогично созданию массивов других типов данных. Например, если нам необходимо хранить информацию о 10 автомобилях, мы можем создать массив структур типа car.

Пример 23. Создание массива структур типа car.

```
car all_cars[10];
strcpy(all_cars[0].brand, "BMW");
…
all_cars[0].year = 2010;
strcpy(all_cars[1].brand, "Toyota");
…
allcars[9].year = 1998;
```

Объединения.

Объединения C++ очень похожи на структуры. Они описываются с помощью ключевого слова union. Например,

```
union distance {
float kilometres;
int metres;
} path1, path2;
```

Мы объявили две переменные path1 и path2 типа distance для хранения длин первого и второго пути.

В отличие от структур при использовании объединений мы можем задать значение только для одного элемента (поля) в каждый момент времени. Если бы distance был структурой, мы могли бы задать длины пути одновременно в километрах и в метрах. Но так как distance – объединение, длина задается или в километрах, или в метрах. Переменная типа distance при этом будет занимать столько же места, сколько отводится самому большому полю в объединении, то есть 4 байта.

Если сначала присвоить значение какому-либо полю объединения, а затем другому полю, первое значение будет потеряно, что демонстрирует пример 24.

Пример 24. Особенности работы с объединениями.

```
path1.kilometres = 1.7;
path1.metres = 1700;
cout << path1.kilometres << endl; // вывод: 2.38221e-042
cout << path1.metres << endl;// вывод: 1700
```

Перечисления

Перечисления – это еще один пользовательский тип данных, который предназначен для хранения некоторого небольшого множества значений.

Предположим, что нашей программе необходимо хранить строковые константы, содержащие цвета радуги. Это можно записать так:

```
const char color1[]="красный";
…
const char color7[]="фиолетовый";
```

Более эффективным способом решения такой задачи является использование перечисления:

```
enum colors = {красный, оранжевый, желтый, зеленый, голубой,
синий, фиолетовый};
```

Каждому элементу перечисления присваивается целочисленная константа – номер. По умолчанию элементы перечисления нумеруются с нуля, но это можно изменить при описании перечисления.

Пример 25. Использование перечислений.

```
enum cardinal = {north = 5, west = 15, south = 25, east = 7};
colors a =  (colors)0; // требуется явное преобраз. к типу colors
colors b = синий;
cardinals n =  (cardinals)5;
if (a == красный) cout << "Это красный цвет" << endl;
if (b == 5) cout << "Это синий цвет" << endl;
if (n == north) cout << "Север" << endl;
```

Задание к работе

Описать типы структур: а) book – для хранения информации о библиотечной книге (поля – название, автор, число страниц, год издания); б) reader – для хранения информации о читателе (поля – ФИО, год рождения, пол, серия паспорта, номер паспорта, домашний адрес, телефон); в) list – для хранения информации о выданных книгах (поля – читатель, книга, дата выдачи, срок в неделях или месяцах, на который выдана книга).

Создать приложение на C++, реализующее следующий функционал:

1) добавление новой книги;

2) поиск книги по автору или названию;

3) добавление нового читателя;

4) поиск читателя по фамилии;

5) редактирование списка выданных книг;

6) вывод книг, выданных читателю с заданной фамилией.

При создании приложения использовать массивы структур, объединения, перечисления.

Содержание отчета

Титульный лист, цель работы, задание к работе, краткая теория, выполнение работы (листинг программы), результаты работы, вывод.

Библиографический список

1. Алгоритмы. Основные алгоритмические конструкции : сб. задач / Челяб. гос. пед. ун-т ; сост. С. А. Рогозин. – Челябинск, 2008. – 42 с.

2. Голицына, О. Л. Основы алгоритмизации и программирования : учеб. пособие / О. Л. Голицына, И. И. Попов. – М. : ФОРУМ, 2008. – 432 с.

3. Павловская, Т. А. C/C++. Программирование на языке высокого уровня / Т. А. Павловская. – СПб. : Питер, 2003. – 461 с.

4. Подбельский, В. В. Программирование на языке Си : учеб. пособие / В. В. Подбельский, С. С. Фомин. – М. : Финансы и статистика, 2004. – 600 с.

5. Керниган, Б. Язык программирования Си / Б. Керниган, Д. Ритчи. – СПб. : Невский диалект, 2001. – 352 с.

ОГЛАВЛЕНИЕ

БАЗОВЫЕ СРЕДСТВА ЯЗЫКА С++

Методические указания к лабораторным работам
по дисциплине «Программирование»

Составитель
БОЧЕНИНА Клавдия Олеговна

Ответственный за выпуск – зав. кафедрой доцент С. Ю. Абрамова

Подписано в печать 08.07.13.
Формат 60x84/16. Усл. печ. л. 1,86. Тираж 50 экз.
Заказ
Издательство
Владимирского государственного университета
имени Александра Григорьевича и Николая Григорьевича Столетовых.
600000, Владимир, ул. Горького, 87.