**Лекция 1**
**Алгоритмизация вычислительных задач.**
**Общие сведения о языке С++**

**Цель лекции**. Изучение основных понятий и конструкций схем алгоритмов.

Изучение алфавита языка С++. Типы данных, ключевые слова, идентификаторы, объявление переменных и констант в программе.

**Основные вопросы лекции:**

1. Введение.
2. Основные понятия и определения при алгоритмизации задач.
3. Простые операции и их базовые конструкции.
4. Составные операции и их базовые конструкции.
5. Виды алгоритмов.
6. Алфавит языка С++.
7. Ключевые слова, идентификаторы, комментарии в С++.
8. Типы данных в языке С++.
9. Объявление переменных и констант в С++.
10. Литералы и другие типы данных в С++.

## 1. Введение

Алгоритмический язык С++ - это один з основных языков программирования, который называется объектно-ориентированным языком. Причина такого названия будет рассмотрена в следующих лекциях. С++ будет изучаться нами в составе программного пакета Microsoft Visual Studio 2010 как Visual C++ 2010. Язык С++ позволяет решать множество задач – от простых школьных заданий до сложнейших задач ядерной физики и космических исследований.

В данном курсе будет изучаться синтаксис, семантика и техника программирования на языке С++. Приведено большое количество программ, иллюстрирующих возможности и особенности языка С++.

Конспект лекций предназначен для получения студентами в ходе изучения учебной дисциплины «Вычислительная техника и программирование» теоретических навыков применения языка С++ при решении прикладных задач на компьютере, а также в качестве вспомогательной литературы при курсовом и дипломном проектировании на старших курсах.

Кроме того, конспект лекций можно использовать при изучении дисциплины «Программирование».

## 2. Основные понятия и определения при алгоритмизации задач

Для решения любой задачи на компьютере необходимо выполнить следующие этапы:

1. Разработать математическую модель задачи, т. е. дать математическое описание объекта исследований.
2. Выбрать или разработать **метод решения** задачи.
3. Составить а**лгоритм** решения задачи.
4. На одном из языков программирования разработать **программу** для решения задачи.
5. Добиться выполнения программы, устраняя возможные ошибки.
6. Проанализировать полученный результат и сделать выводы.

В данной лекции остановимся подробнее на составлении алгоритма.

**Алгоритм решения задачи** - это строгая последовательность действий, которые необходимо выполнить над данными, чтобы получить искомый результат (решить поставленную задачу).

**Программа** – это записанный на языке программирования алгоритм решения задачи.

Каждый язык программирования или алгоритмический язык является набором символов и слов, с помощью которых программист записывает команды (инструкции) для компьютера. Затем программа на языке программирования переводится соответствующей программой (компилятором) в машинный код для дальнейшего выполнения компьютером.

Любой разрабатываемый алгоритм должен удовлетворять следующим требованиям:

1. **Определенность** – алгоритм не должен допускать неоднозначность толкования.

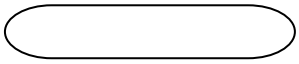
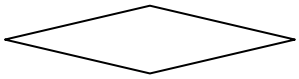
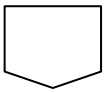
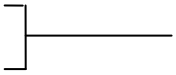
2. **Массовость** – возможность использования алгоритма при решении подобных задач.

3. **Результативность** – выполнение действий в соответствии с разработанным алгоритмом должно приводить к получению искомого результата.

В программировании принято алгоритм решения задачи изображать в графическом виде. При этом все операции или действия изображаются в виде отдельных блоков. Каждое действие, например, ввод данных, печать документа и др., имеет свое стандартное условное обозначение (см. таб. 1.1). Конфигурация и размер блоков определяются Государственным Стандартом (ГОСТОМ).

Последовательность действий, необходимых для решения поставленной задачи, изображенная в виде набора стандартных операционных блоков, называется **блок-схемой алгоритма**.

**Таблица 1.1 Основные операционные блоки схем алгоритмов**

| № п/п | Условное обозначение | Наименование | Описание операции |
|---|---|---|---|
| 1 |  | Начало, завершение | Начало и завершение алгоритма |
| 2 |  | Процесс | Вычислительная операция или их совокупность |
| 3 |  | Решение | Проверка условия и выбор дальнейшего направления процесса решения |
| 4 |  | Модификация | Заголовок цикла, проверка условий цикла |
| 5 |  | Данные | Ввод исходных данных, вывод данных и результатов |
| 6 |  | Типовой процесс | Использование ранее созданных алгоритмов, подпрограмм, функций |
| 7 |  | Печать документа | Вывод данных на печать |
| 8 |  | Соединитель внутристраничный | Разрыв линий потока в пределах одной страницы |
| 9 |  | Соединитель межстраничный | Перенос линий потока на другую страницу |
| 10 |  | Узел | Слияние линий потока |
| 11 |  | Комментарии | Описание операционного блока и его особенностей |

В блок-схемах алгоритмов операционные блоки соединяются друг с другом линиями потока. Линии потока, направленные вниз и направо могут быть без стрелок, указывающих направление.

Линии потока, направленные вверх или влево обязательно должны заканчиваться стрелками. Базовые конструкции разделяются на простые и составные.

### 3. Простые операции и их базовые конструкции

Простое действие – это одна операция. Основные простые действия следующие:

- присваивание;
- ввод;
- вывод.

**Присваивание** – действие, в результате которого переменная получает определенное значение. Операционный блок присваивания в схемах алгоритмов обозначается символом “процесс” (прямоугольник) (рис. 1.1).

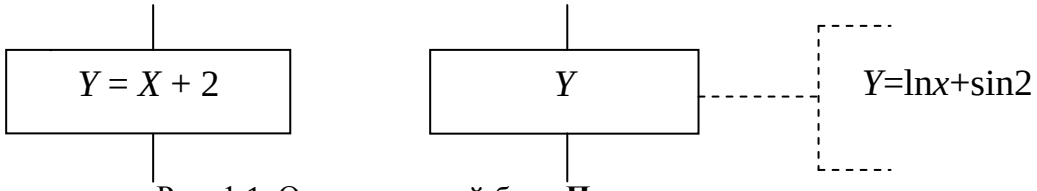



Рис. 1.1. Операционный блок **Присваивание**

В середине прямоугольника записывается команда или имя переменной. Если математическое выражение имеет сложный вид, тогда записывается комментарий к блоку.

**Ввод** – действие, в результате которого переменной присваивается начальное значение.

**Вывод** - действие, в результате которого данные выводятся для отображения.

Операционный блок ввода или вывода в схемах алгоритмов обозначается символом “данные” (параллелограмм) (рис. 1.2). В середине символа слово **Ввод** или **Вывод** и в круглых скобках имя переменной или просто перечисляются имена переменных, которые должны быть введены.

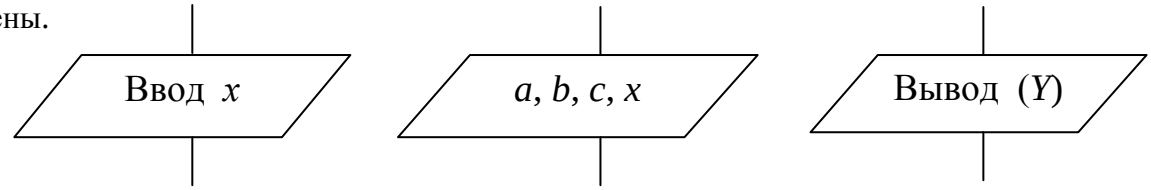



Рис. 1.2. Операционный блок **Ввод** (**Вывод**)

Исследуем процесс разработки алгоритма простейшей программы на следующем примере.

Необходимо вычислить силу тока I в новогодней гирлянде, состоящей из n=50 электрических лампочек сопротивлением **r**=20 Ом каждая. Используя закон Ома и формулу для расчета суммарного сопротивления последовательной цепи, составим блок-схему алгоритма (рис. 1.3). Перед началом расчетов вводятся значения переменных **r, n, U**. Затем идет операционный блок, в котором рассчитываются общее сопротивление гирлянды **R** и сила тока **I**. Значение **I** выводится.

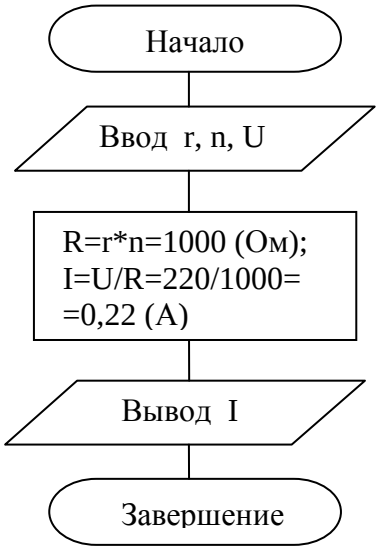


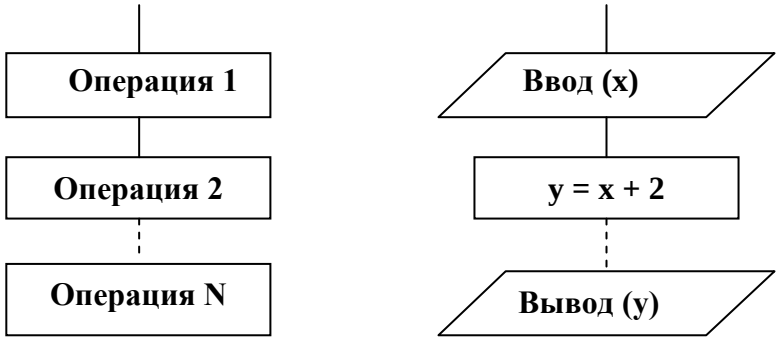

Рис. 1.3. Блок-схема алгоритма определения силы тока в новогодней электрогирлянде

## 4. Составные операции и их базовые конструкции

Составные операции состоят из простых операций и условий их выполнения и включают:

- прохождение;
- выбор или разветвление;
- повторение или цикл.

**4.1. Прохождение –** последовательность простых операций, выполняемых одна за другой (рис. 1.4).




а) б)

Рис. 1.4. Блок-схема составной операции **Прохождение:**
а) общий вид; б) пример

**4.2. Выбор** (разветвление) – это прохождение по одному из двух возможных направлений алгоритма в зависимости от некоторого условия (рис. 1.5).

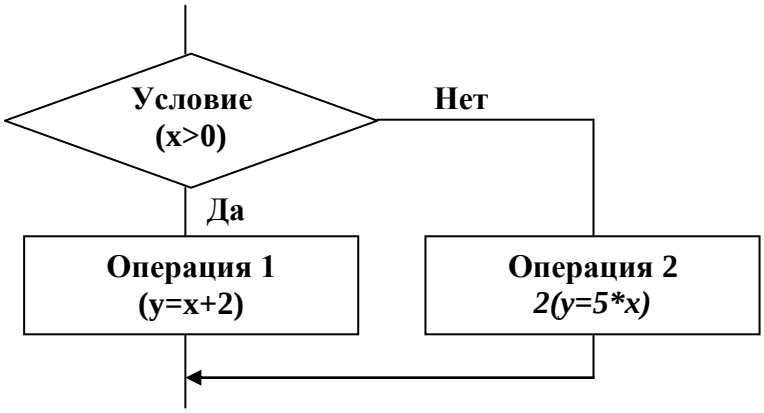



Рис. 1.5. Блок-схема составной операции **Выбор**

При выполнении операции **Выбор** проверяется, удовлетворяет ли значение переменной X некоторому условию. Если условие выполняется, то производится **Операция 1** (направление потока **Да**), в противном случае – **Операция 2** (направление **Нет**). В зависимости от условия задачи блок **Операция 2** может отсутствовать.

**3.3. Повторение** (цикл) – многократное выполнение одной и той же последовательности операций в зависимости от некоторого условия.

Подробно данная операция будет рассмотрена в 6-й и 7-й лекциях.

## 5. Виды алгоритмов

Как было отмечено выше, алгоритм – это последовательность определенных действий, выполняемых над исходными данными для решения какой-либо задачи. При этом используются различные виды простых и составных операций (см. раздел 3) или их комбинаций. В зависимости от вида этих операций различают следующие типы алгоритмов:

- линейные;
- разветвленные;
- циклические.

В **линейном алгоритме** действия выполняются последовательно одно за другим. Примером линейного алгоритма является достаточно простая блок-схема решения задачи по определению электрического сопротивления новогодней гирлянды (рис. 1.3).

**Разветвленный** и **циклический** алгоритмы, их блок-схемы и особенности использования будут нами рассмотрены ниже при изучении соответствующих операторов языка программирования С++.

## 6. Алфавит языка С++

В тексте на любом естественном языке различают четыре основных группы элементов: символы, слова, словосочетания и предложения. Подобные элементы содержит и любой язык программирования, только слова называют **лексемами** (элементарными конструкциями), словосочетания - **выражениями**, а предложения - **операторами**.

**Алфавит** языка - это базовый набор символов. Алфавит языка С++ состоит из латинских букв (прописных и строчных), арабских цифр, специальных и управляющих символов.

1. **Буквы** - A, B, C, ..., Z, a, b, c, ..., z.
2. **Цифры** - 0, 1, 2, ..., 9.
3. **Специальные символы** используются для организации процесса вычислений и для передачи компилятору определенных инструкций (см. таб. 1.2).

**Таблица 1.2 Специальные символы языка С++**

| Символ | Наименование | Символ | Наименование |
|---|---|---|---|
| , | Запятая | ) | Правая круглая скобка |
| . | Точка | ( | Левая круглая скобка |
| ; | Точка с запятой | } | Правая фигурная скобка |
| : | Двоеточие | { | Левая фигурная скобка |
| ? | Вопросительный знак | < | Меньше |
| ' | Апостроф | > | Больше |
| ! | Восклицательный знак | [ | Правая квадратная скобка |
| \| | Вертикальная черта | ] | Левая квадратная скобка |
| / | Дробная черта | # | Номер |
| \ | Обратная дробная черта | % | Процент |
| ~ | Тильда | & | Амперсанд |
| * | Звездочка | ^ | Логическое Нет |
| + | Плюс | = | Равно |
| - | Минус | " | Кавычки |

К специальным символам относятся также знаки пробела, табуляции, перевода строки, возврата каретки, новой страницы и новой строки. Эти символы отделяют друг от друга константы и идентификаторы. Последовательность разделительных символов рассматривается компилятором как один символ (последовательность пробелов).

4. **Управляющие символы**. В языке С++ используются управляющие последовательности – комбинации символов, используемые в функциях ввода и вывода информации. Управляющая последовательность строится на основе использования обратной дробной черты (\) (обязательный первый символ) и комбинации латинских букв и цифр (см. таб. 1.3).

**Таблица 1.3 Управляющие последовательности**

| Управляющая последовательность | Наименование |
|---|---|
| **\b** | Возвращение на шаг |
| **\n** | Переход на новый ряд |
| **\r** | Возвращение каретки |
| **\f** | Новая страница |
| \" | Кавычки |

| | |
|---|---|
| **\'** | Апостроф |
| **\0** | Ноль-символ |
| **\t** | Горизонтальная табуляция |
| **\v** | Вертикальная табуляция |
| **\a** | Звуковой сигнал |
| **\ ** | Обратная дробная черта |
| **\?** | Вопросительный знак |

### 7. Идентификаторы. Ключевые слова. Комментарии

**Идентификатор** - это имя программного объекта. В идентификаторе могут использоваться латинские буквы, цифры и знак подчеркивания. Прописные и строчные буквы различаются. Первым символом идентификатора может быть буква или знак подчеркивания. Например, **a**, **b**, **x**, **summa**, **Summa**, **Y_2345 (**язык С++ различает прописные и строчные буквы).

Длина идентификатора по стандарту не ограничена. Идентификатор создается на этапе *объявления переменной*, функции, типа и т.п., после этого его можно использовать в последующих операторах программы. При выборе идентификатора необходимо иметь в виду следующее:

- идентификатор не должен совпадать с ключевыми словами и именами используемых стандартных объектов языка;
- не рекомендуется начинать идентификаторы с символа подчеркивания;
- на идентификаторы, используемые для определения *внешних переменных*, налагаются ограничения компоновщика.

**Ключевые слова** - это зарезервированные идентификаторы, которые имеют специальное значение для компилятора. Их можно использовать только в том смысле, в котором они определены. В таблице 1.4 приведен список ключевых слов С++ в алфавитном порядке.

**Таблица 1.4 Ключевые слова языка С++**

| | | | | |
|---|---|---|---|---|
| **asm** | **do** | **if** | **return** | **typedef** |
| **auto** | **double** | **inline** | **short** | **typeid** |
| **bool** | **dynamic _ cast** | **int** | **signed** | **typename** |
| **break** | **else** | **long** | **sizeof** | **union** |
| **case** | **enum** | **mutable** | **static** | **unsigned** |
| **catch** | **explicit** | **namespace** | **static_cast** | **using** |
| **char** | **export** | **new** | **struct** | **virtual** |
| **class** | **extern** | **operator** | **switch** | **void** |
| **const** | **false** | **private** | **template** | **volatile** |
| **const_cast** | **float** | **protected** | **this** | **wchar_t** |
| **continue** | **for** | **public** | **throw** | **while** |
| **default** | **friend** | **register** | **true** | |

**Комментарий** - это поясняющий значение фрагментов программного кода текст, предназначенный для пользователя. Комментарии компилятор не воспринимает как часть программы. В С++ комментарий можно сделать одним из двух способов:

1. Символы «/ *» начинают комментарий, заканчивающийся символами «* /». Вся эта последовательность символов эквивалентна символу пробела. Этот способ полезно использовать для многострочных комментариев и изъятия частей программы при редактировании:

**#include <iostream. h>** **/*** директива компилятору для включення в программу заголовочного файла **iostream. h** */

2. Символы «//» предваряют комментарий, который заканчивается в конце данной строки:

**int a = 17;** // инициализация переменной

### 8.Типы данных в языке С++

Основная цель любой программы состоит в обработке данных. Данные, с которыми работают машинные команды, хранятся в оперативной памяти. Компилятору для формирования

команд необходимо точно знать, сколько места занимают данные, как именно закодированы и какие действия с ними можно выполнять. Все это задается при описании данных с помощью типа. Каждая константа, переменная, результат вычисления выражения или функции должны иметь конкретный тип.

**Тип данных** однозначно определяет:

1. Множество их возможных значений (связанное с внутренним представлением данных в памяти компьютера);
2. Допустимые действия над данными (операции и функции).

Программист задает тип каждой величины, используемой в программе, исходя из характеристик реальных объектов, которые представляют эти величины. Обязательное указание типа позволяет компилятору проверять, правильно ли используется объект в конструкциях языка. От типа величины зависят машинные команды, которые будут использоваться для обработки данных.

Объем памяти, выделяемый для величин, тоже зависит от типа. По стандарту единицей памяти в С++ является байт.

Типы данных в языке С++ делятся на элементарные (базовые или основные) и составные. **Элементарные** типы данных являются неделимыми и позволяют описывать целые, вещественные, символьные и логические величины. На основе этих типов программист может конструировать составные типы. К составным типам относятся массивы, структуры, объединения, перечисления, ссылки, указатели и классы.

**Базовые типы данных** в языке С++ часто называют арифметическими, поскольку их можно использовать в арифметических операциях. Существует пять базовых типов данных:

- **bool** (логический);
- **char** (символьный);
- **int** (целый);
- **float** (действительный);
- **double** (действительный с двойной точностью).

Существует четыре ключевых слова, уточняющих внутреннее представление и диапазон значений стандартных типов:

- **short** (короткий);
- **long** (длинный);
- **signed** (знаковый);
- **unsigned** (беззнаковый).

Сочетания перечисленных ключевых слов формируют 14 различных арифметических типов. Например, **char, signed char** и **unsigned char** - это три равноправных различных типа.

Рассмотрим базовые типы данных языка С++ подробнее.

**Логический тип.** Величины логического типа могут принимать только значения **true** и **false** (истина и ложь), являющиеся ключевыми словами. Величины логического типа могут участвовать в арифметических операциях. При преобразовании к целому типу **true** имеет значение 1, **false** - 0. Размер логического типа в стандарте не определен и зависит от реализации.

**Символьный тип**. В стандарте языка С++ определено три различных символьных типа: **char**, **signed char** и **unsigned char**. Внутренним представлением символа является его код - целое число. Под величину любого символьного типа отводится одна единица памяти - байт:

**sizeof(char) = sizeof(signed char) = sizeof(unsigned char) = 1 .**

Размер байта зависит от реализации, однако этот размер должен быть достаточен, чтобы вместить код любого символа из набора символов реализации для данного компьютера. Наличие знака у типа **char** тоже зависит от реализации: он может совпадать либо с **signed char**, либо с **unsigned char.**

Величины символьных типов применяются также для хранения целых чисел, не превышающих границы указанных диапазонов, и могут участвовать в арифметических операциях, поэтому их также относят к целым типам.

**Целый тип.** В языке С++ определено 8 типов для хранения целочисленных величин: четыре знаковых (**signed char**, **short int**, **int**, **long int**) и четыре беззнаковых (**unsigned char**, **unsigned short int**, **unsigned int**, **unsigned long int**). По умолчанию все целочисленные типы считаются

знаковыми, поэтому спецификатор **signed** можно не указывать. Ключевое слово **unsigned** позволяет представлять неотрицательные целые числа. Диапазоны целых чисел приведены в таблице 1.5.

**Таблица 1.5** **Целый тип данных в С++**

| Тип | Размер | Диапазон числа без знака | Диапазон числа со знаком |
|---|---|---|---|
| **short int** | 2 байта | 0.... 65535=$2^{16}$-1 | -$2^{15}$...$2^{15}$-1=-327678... 32767 |
| **int** | 4 байта | 0.... 4294967295 = $2^{32}$-1 | -$2^{31}$... $2^{31}$-1 |
| **long int** | 4 байта | 0.... 4294967295 = $2^{32}$-1 | -$2^{31}$... $2^{31}$-1 |

Действительные числа записываются в обычной десятичной либо экспоненциальной форме. Например, **123.5**, **3.14**, **0.95**.

Число **123.5** можно записать в экспоненциальной форме как совокупность числа, называемого мантиссою, и порядка, который представляет степень 10-ти. Например, **$123.5 * 10^0 = 12.35 * 10^1 = 1.235 * 10^2 = 1235 * 10^{-1}$.**

В языке С++ вместо цифры 10 записывается буква Е:

**123.5E0 = 12.35E1 = 1.235E2 = 1235E-1**.

**Действительный тип** – это множество рациональных и иррациональных чисел с десятичной точкой. В языке С++ используются следующие действительные типы: **float** – числа з плавающей точкой одинарной точности, **double** - числа з плавающей точкой двойной точности, **long double – для** вычислений особой точности. Диапазон чисел действительного типа приведен в таблице 2.5.

**Таблица 2.5** **Действительный тип данных в С++**

| Тип | Размер | Диапазон числа |
|---|---|---|
| **float** | 4 байта | -$2^{31}$... $2^{31}$-1 |
| **double** | 8 байт | -$2^{63}$... $2^{63}$-1 |
| **long double** | 8 байт | -$2^{63}$... $2^{63}$-1 |

## 9. Объявление переменных и констант в языке С++

Основное назначение всех компьютерных программ – операции с данными с целью получения некоторого результата. Условием этого процесса является наличие фрагмента памяти запоминающего устройства компьютера, в котором можно хранить элемент данных и к которому можно обращаться по некоторому имени.

**Переменная** – это фрагмент памяти, в котором хранится элемент данных и к которому можно обращаться по некоторому имени. **Имена переменных** могут включать буквы латинского алфавита **A** – **z** (в верхнем или нижнем регистре), цифры от 0 до 9 и знак подчеркивания. Имена переменных должны начинаться либо с буквы, либо со знака подчеркивания. В С++ принято назначать имена переменных с прописных букв, а классов – с заглавных. Компилятор С++ различает прописные и строчные буквы, например, **Sum** и **sum** означают разные переменные.

**Объявление** переменной с одновременным заданием типа хранимого под ее именем элемента данных осуществляется следующим образом:

**ТипПеременной ИмяПеременной;**

Например, строка **int arg;** объявляет целую переменную с именем **arg.**

Имена переменных не должны совпадать с ключевыми словами.

Объявляя переменную, можно сразу **присвоить** ей начальное значение. Например

**int sum=0;** или **int sum(0);**

**float a=2.7;** или **float a(2.7);**

Переменная объявляется **перед** тем местом, где она будет впервые задействована. Подробнее о месте объявления переменных в С++ будет сказано в следующих лекциях.

**Константа** – это переменная, не меняющая свое значение в программе. В С++ объявление константы выглядит следующим образом:

**const Тип ИмяКонстанты = Значение; .**

Например, объявить постоянную $\pi$ можно следующим образом:

**const float pi = 3.14159;** .

## 10. Литералы и другие типы данных

Литералы – это данные, находящиеся непосредственно в тексте программы. Литералы могут быть числовые, символьные и строковыми.

**Числовые** литералы (числа) могут быть целыми и действительными. Целый литерал — это целое число, записанное в тексте программы. Число может быть записано в десятичной, восьмеричной или шестнадцатеричной системах. Для того чтобы отличить, в какой системе счисления записано число, перед ним в восьмеричной системе записывают ноль, а в шестнадцатеричной – еще и латинскую букву «x». Например:

**4523, 679134** - числа в десятичной системе счисления;

**067, 01497** - числа в восьмеричной системе счисления;

**0x73, 0x948** - числа в шестнадцатеричной системе счисления.

**Символьный литерал** (символьная константа) используется для представления одного символа. Это любой символ, заключенный в одинарные кавычки:

**‘W’, ‘&’, ‘3’, ‘Ф’.**

**Строковый литерал** – это любая последовательность символов, заключенная в парные кавычки:

**“ALPHA”, “ТАБЛИЦЯ”, “Это строка \n”, “123/5”.**

**Массивы** - структурированные наборы данных.

**Указатели – содержат адреса ячеек памяти, в которых содержатся данные.**

**Строки** – особая форма массивов, содержащих символьные данные.

**Записи** – структуры, связывающие элементы разных типов в один объект.

**Файлы** – структуры, представленные хранимыми данными в совокупности с операциями передачи.

### Вопросы для самоконтроля

1. Что такое алгоритм?
2. Что обозначает ромб на схеме алгоритма?
3. Какая последовательность решения задачи на ЭВМ?
4. Что обозначает прямоугольник на схеме алгоритма?
5. Как изображается начало алгоритма в схеме?
6. Что обозначает параллелограмм на схеме алгоритма?
7. Какая конструкция используется для изображения разветвления в схемах алгоритмов?
8. Что такое простые операции?
9. Что такое составные операции?
10. Какая конструкция используется для изображения цикла?
11. Как правильно записать действительное число на языке *C*++?
12. В результате решения задачи на ПК на экране появилось число 7.45600000*E*+01. Какое это число?
13. Как производится запись символьной константы в языке *C*++?
14. Как записывается идентификатор в *C*++?
15. Как записывается константа в *C*++?
16. Перечислите базовые типы данных в *C*++.
17. Дайте определение константы.
18. Дайте определение переменной.
19. Для чего необходимо объявлять переменные перед действиями с ними?
20. Каков формат объявления переменной?