1. ЭЛЕМЕНТЫ ЯЗЫКА

1.1. Первые программы

Напишем простейшую программу:

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

void main( ){
cout<<"Hello, World!\n"; }
```

Здесь в первой строке расположена так называемая **директива препроцессора include** (включающая директива). Выполнение этой директивы приведет к тому, что вместо первой строки в программу будет вставлено содержимое файла iostream. Файл iostream относится к так называемым **хедерам**, или заголовочным файлам. Заголовочные файлы содержат тексты на языке C++, и компилятор после выполнения директивы include будет обрабатывать новый полученный текст программы. Вторая строка определяет так называемое пространство имен **std**.

В третьей строке исходного текста программы находится заголовок функции с именем **main**. Пустые круглые скобки говорят о том, что эта функция не имеет аргументов, а **ключевое слово void** означает, что функция main не возвращает никакого значения. В фигурных скобках находится блок, который часто называют телом программы.

В третьей строке стоит оператор, действие которого – вывод в поток **cout** (на английском звучит как **see-out**), который здесь ассоциируется с экраном дисплея. В результате на экране появится строка:

```
Hello, World!
```

Символ '\n' в конце текста в кавычках сообщает компилятору, что после вывода текста на экран нужно перейти на новую строку.

Теперь приведем пример простой программы и диалога, который появляется на экране компьютера, если пользователь запускает эту программу и вводит соответствующие данные. В дальнейшем **пользователем** мы будем называть человека, который использует программу. Данные, введенные пользователем, выделены в тексте

диалога жирным шрифтом. **Программистом** мы будем в дальнейшем называть, естественно, автора программы.

```
#include <iostream> using
namespace std;
void main( ){ int m, n, sum;                                    //
Описания.
cout << "Для ввода чисел задайте два числа на клавиатуре\n";
cout << " (через пробелы) и нажмите Enter.\n"; cin >> m >> n;
                                  // Ввод чисел. sum = m + n; cout <<
"При m =  " << m << "  и  n = " << n
<< "   их  сумма равна " << sum << ".\n";      // Вывод результата.
}
```

При выполнении этой программы на экране монитора появится следующий диалог:

```
Для ввода чисел задайте два числа на клавиатуре
(через пробелы) и нажмите Enter. 20
45
При m = 20  и  n = 45    их  сумма равна 65.
```

В этой программе за символами // стоят **комментарии**. В третьей строке **описываются** целые переменные с **именами** m, n и sum.

Оператор, начинающийся со слова **cin**, сообщает компилятору, что введенные пользователем значения, равные 20 и 45, нужно поместить соответственно в **переменные** m и n. Объект cin (читается как **see-in**) – поток ввода – здесь понимается как клавиатура, а стрелки << и >> указывают направление, в котором перемещаются данные.

Перейдем теперь к описанию языка C++ и его возможностей.

1.2. Алфавит языка

В алфавит языка входят:

1. Прописные латинские буквы A...Z.
2. Строчные латинские буквы a...z.
3. Арабские цифры 0...9.
4. Символ подчеркивания _ (рассматривается как буква).
5. Пробельные символы.
6. Знаки пунктуации и специальные символы (табл. 1).

Символы группы 1–4 используются для образования ключевых слов и имён языка.

Имя есть последовательность букв и цифр, начинающаяся с буквы и не являющаяся ключевым словом. (Символ _ в начале имени ставить не рекомендуется.)

В C++ прописные и строчные буквы различаются, поэтому имена ARG1 и arg1 являются различными.

Таблица 1

Знаки пунктуации и специальные символы

Символы	Наименование	Символы	Наименование
,	запятая	{	открывающая скобка
.	точка	}	закрывающая скобка
;	точка с запятой	<	меньше
:	двоеточие	>	больше
?	знак вопроса	[	открывающая скобка
'	апостроф	]	закрывающая скобка
!	восклицательный знак	#	номер или решетка
|	прямая черта	%	процент
/	слеш	&	амперсанд
\	обратный слеш	^	НЕ-логическое
~	тильда	-	минус
*	звездочка	=	равенство
(	открывающая скобка	"	кавычки
)	закрывающая скобка	+	плюс

К пробельным символам относятся: пробел, символы табуляции, перевода строки, возврата каретки, перевода страницы. Эти символы отделяют друг от друга **лексемы** языка. Любая последовательность пробельных символов рассматривается при компиляции как один пробел.

1.3. Комментарии

Короткие комментарии можно написать так: //

символы до конца строки.

Комментарии, не умещающиеся в одну строку, пишутся следующим образом: /* символы

символы

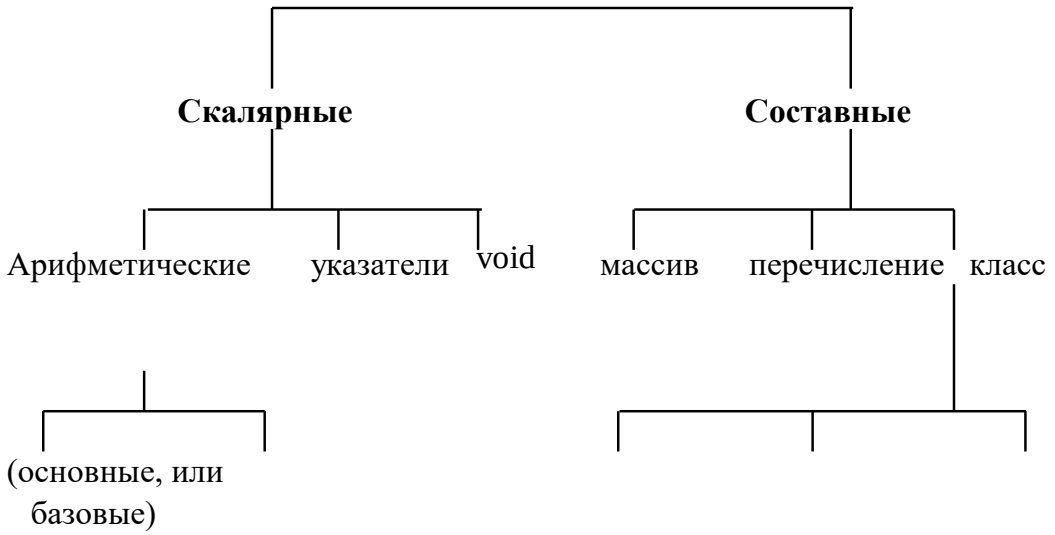
. . .

символы */

В комментариях символы – это не только литеры из алфавита языка C++, но и любые возможные символы, включая русские буквы.

1.4. Типы данных

Все типы данных можно разделить на две категории: скалярные и составные.



целые плавающие структура класс объединение

Ключевыми словами, используемыми при объявлении основных типов данных, являются

- для целых типов: **char, int, short, long, signed, unsigned**;
- для плавающих типов: **float, double, long double**;
- для классов: **structure, union, class**;
- для перечисления: **enum**; для типа void: **void** (пустой).

Целые типы данных

Тип char, или символьный

Данными типа char являются различные символы, причем значением этих символов является численное значение во внутренней кодировке ЭВМ.

Символьная константа – это символ, заключенный в апострофы, например, '&', '4', '@', 'a'. Символ '0', например, имеет в кодировке ASCII значение 48.

Существуют две модификации этого типа: signed char и unsigned char.

Данные char занимают один байт и меняются в диапазоне: signed char (или просто char) -128 ... 127; unsigned char 0...255.

Отметим, что если необходимо иметь дело с переменными, принимающими значения русских букв, то их тип должен быть unsigned char, так как коды русских букв > 127 (в кодировке ASCII).

Символы, в том числе и неграфические, могут быть представлены как символьные константы с помощью так называемых управляющих последовательностей.

Управляющая последовательность – это специальные символьные комбинации, которые начинаются с символа '\', за которым следует буква или комбинация цифр (см. табл. 2).

Последовательности '\0ddd' и '\0xdd' позволяют представлять любой символ из набора ЭВМ как последовательность восьмеричных или шестнадцатеричных цифр соответственно. Например, символ возврата каретки можно задать так: '\r', или '\015', или '0xD'.

Таблица 2

Специальные управляющие последовательности

Управляющая последовательность	Наименование
\a	Звонок
\b	Возврат на шаг
\t	Горизонтальная табуляция
\n	Перевод строки
\v	Вертикальная табуляция
\r	Возврат каретки
\f	Перевод страницы
\"	Кавычки
\'	Апостроф

\\	Обратный слеш

Тип short (эквивалент short int)

Данные типа short занимают 2 байта и принимают целые значения из диапазона -32768…32767.

Тип unsigned short

Данные такого типа занимают **2** байта, и диапазон их значений 0…65535.

Тип int

Данные типа **int** в разных системах могут занимать:

- либо **2** байта и принимать, соответственно, целые значения в диапазоне -32768…32767;
- либо 4 байта, и тогда их диапазон значений будет -2147483648…2147483647.

Тип long (long int)

Такие данные занимают **4** байта и изменяются в диапазоне -2147483648 … 2147483647.

Тип unsigned long (int)

Такие данные занимают **4** байта и изменяются в диапазоне 0 … 4298876555.

Отметим, что если целая константа выходит из диапазона **int**, то она автоматически становится константой типа **long** или даже **unsigned long**.

Так, 32768 имеет (при двухбайтном int) тип **long**, 2676768999 имеет тип **unsigned long**.

Задать тип константы можно и явно с помощью суффиксов 'U' и 'L': -6L 6U 33UL.

Заметим, что в стандарте языка определено лишь, что sizeof(char)=1 и sizeof(char)<=sizeof(short)<=sizeof(int)<=sizeof(long).

Здесь **sizeof**(**type**) – операция, определяющая размер типа **type** в байтах.

Целая константа, которая начинается с нуля, является восьмеричной константой, а начинающаяся символами 0x – шестнадцатеричной константой. Например,

031 0750 01 – восьмеричные константы; 0x17 0xA9 0xFF – шестнадцатеричные константы.

Тип bool (логический тип)

Этот тип относится к целым. Данные типа bool занимают один байт. Диапазон допустимых значений – целые числа от 0 до 255. Данные типа bool используются для хранения результатов логических выражений. У логического выражения может быть один из двух результатов: **true** или **false**; true – если логическое выражение истинно, false – если логическое выражение ложно.

Так как диапазон допустимых значений типа данных bool от 0 до 255, то нужно сопоставить этот диапазон с определёнными в языке логическими константами true и false. Константе true эквивалентны все числа от 1 до 255 включительно, тогда как константе false эквивалентно только одно целое число – 0.

Плавающие типы данных Данные о плавающих типах, которые представляют в ЭВМ вещественные числа, приведены в табл. 3 (в среде разработки Visual Studio C++).

Таблица 3

Тип	Длина	Диапазон	Десятичных цифр
float	4	±3.4e-38 … 3.4e38	7
double	8	±1.7e-308 … 1.7e308	15
long double	8	±1.7e-308 … 1.7e308	15

По умолчанию плавающие константы имеют тип **double**, если они не выходят из соответствующего диапазона. Так, константы

1.0 .3 -6. 2.3e-6 (*означает* $2{,}3\cdot10^{-6}$) -3E19 (*означает* $-3\cdot10^{19}$)

1.2 – типа **double**. Суффикс ℓ говорит о том, что плавающая константа имеет тип **long double**: 3ℓ, -4E8ℓ, 1.6e-19ℓ, 1.3e-200ℓ.

1.5. Константы-строки, или литералы

Константа-строка – это последовательность символов, взятая в кавычки:

"строка", "125 руб.", "\"a + b = c\" – это равенство".

Строковый литерал представляется в памяти как массив элементов типа **char**.

При компиляции в конец каждой строки автоматически добавляется так называемый нуль-символ '\0', являющийся признаком конца строки. Таким образом, в памяти литерал «три» занимает не три, а четыре байта.

Значением строки является адрес ее начала.

1.6. Директива препроцессора define

Директива препроцессора **define** имеет вид **#define**

имя текст_подстановки

Примеры:

#define nmax 1000

#define km(nmax 3+1)

Имя, которое указано в **#define**, заменяется в тексте программы текстом подстановки. Таким образом, вместо имени nmax в тексте везде появится 1000, а вместо km – (1000 * 3+1).

Отметим, что есть еще одна форма директивы #define – с параметрами.

1.7. Описание

Все переменные должны быть описаны до их использования. Описание (часто употребляют «объявление») состоит из спецификатора типа и следующего за ним списка переменных, которые будут иметь указанный тип:

```
int i, j, k, pmax; float
radix, a, b, s_m;
double k, kr; char ch,
ch1;
char symbol;
```

При описании переменная может быть инициализирована некоторым значением, например: char t = 't', BACKSLASH = '\\'; int i = 0, j, k, s = 1; float ro, eps = 1e-6;

1.8. Модификатор *const*

Если в объявлении имени присутствует модификатор **const**, то объект, с которым сопоставлено данное имя, рассматривается в области существования этого имени как константа:

```
const int i = 50;
const double pi = 3.14159265358979;
```

Такие именованные константы в программе изменять нельзя. Использовать эти константы можно так же, как и обычные.