Лекция№2

Основы программирования на языке C/C++

План

1. Особенности языка C++.
2. Стандарты.
3. Структура программы на языке C++.
4. Встроенные типы данных.
5. Переменные.

Основные особенности языка C++

C++ - компилируемый язык программирования;

Парадигмы программирования: процедурное, объектно-ориентированное;

Статическая система типов данных;

Использование препроцессора для абстрагирования однотипных операций;

Синтаксис C++ унаследован от языка C. Совместим с C.

Однако C++ не является в строгом смысле надмножеством C;

Алфавит языка

1. Все буквы латинского алфавита.

2. Цифры: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

3. Специальные символы: **,** (запятая), **;**, . (точка), **+**, **-**, *****, **^**, **&** (амперсанд), =, **~** (тильда), **!**, **/**, **<**, **>**, (,), {, }, [,], |, %, ?, **'** (апостроф), **"** (кавычки), **:** (двоеточие), **_** (знак подчёркивания), ****, **#**.

Символы служат для построения лексем. Существует пять видов лексем:

- идентификаторы;
- ключевые слова;
- знаки (символы) операций; ·
- литералы; ·
- разделители (скобки, точка, запятая, пробел, табуляция).

Ключевые слова — это зарезервированные идентификаторы, которые имеют специальное значение для компилятора.

auto	enum	operator	throw
bool	explicit	private	true
break	export	protected	try
case	extern	public	typedef
catch	false	register	typeid
char	float	reinterpret_cast	typename
class	for	return	union
const	friend	short	unsigned
const_cast	goto	signed	using
continue	if	sizeof	virtual
default	inline	static	void
delete	int	static_cast	volatile
do	long	struct	wchar_t
double	mutable	switch	while
dynamic_cast	namespace	template	

Синтаксические правила языка программирования определяются нормативными документами, которые обычно называют стандартами языка.

Стандарты языка с течением времени меняются.

Необходимо обращать внимание на то, чтобы компиляторы поддерживали данные изменения.

Стандарты языка c++

В 1985 году вышло первое издание «Языка программирования C++», обеспечивающее первое описание этого языка.
В 1989 году состоялся выход C++ версии 2.0.
C++98
C ++11
C++14
C++17
C++20

Структура программы на языке С++

Программа на языке С++ строится по блочному принципу, как совокупность нескольких независимых программных блоков-функций.

Функция – обособленный программный блок, реализующий собственный алгоритм функционирования, обрабатывая входные параметры и формируя как результат своей работы некоторое значение.

Пример структуры программ на языке C++

```
#include <iostream>
int func()
{//тело функции
}
int main()
{
//тело функции
return 0;
}
```

Тело функции – набор инструкций (операторов) языка C++, реализующих алгоритм работы функции.

main() – главная функция программы, она обязательно должна быть.

Выполнение программы всегда начинается с первого оператора функции main. Когда будет выполнена последняя инструкция этой функции, программа завершит свою работу.

Тело функции заключается в { }, которые выполняют роль операторных скобок (аналогично begin и end в языке Pascal)

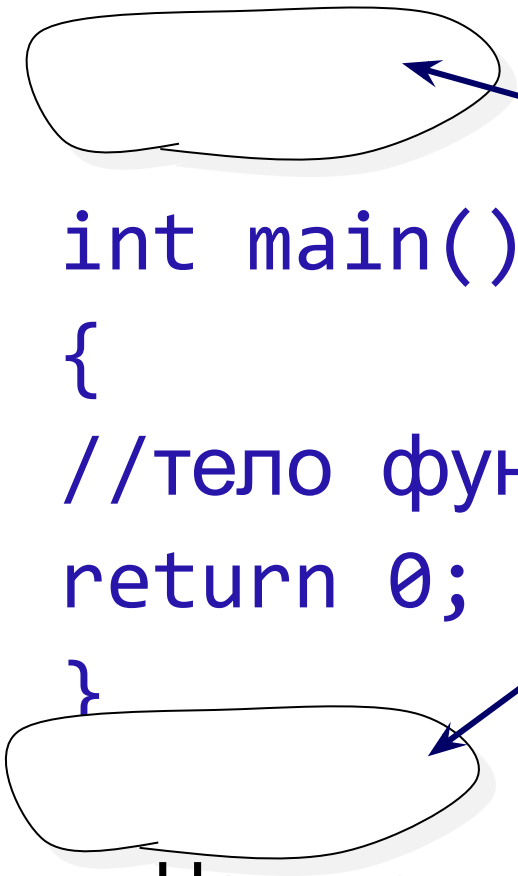
Для функции main используется тип возвращаемого значения int

```
int main()
{//тело функции
return 0;
}
```

Это означает, что в результате своей работы программа должна сформировать и вернуть вызывающей программе целочисленное значение – код статуса своего завершения

.

Вызов оператора return 0 завершает программу с кодом 0, что общепринято означает успешное завершение работы программы



Внешний уровень программы

Внешний уровень программы включает те ее части, которые лежат вне какой-либо функции

На внешнем уровне программы могут находиться инструкции: директивы препроцессора, определения и описания новых типов данных и объектов, комментарии.

Структура простой программы на языке C++

На внешнем уровне программы нельзя помещать операторы

```
#include <iostream>
int x=0;
int main()
{
x=x+10;
cout<<x;
return 0;
}
```

```
#include <iostream>
int x;
x=x+10; //Ошибка
int main()
{
cout<<x;
return 0;
}
```

```cpp
#include <iostream>

using namespace std;

int main()
{
    cout << "Hello World!\n";
   return 0;
}
```

Инструкция using namespace std представляет собой команду для использования стандартного *пространства имен* std.

Пространство имен — это некое абстрактное хранилище, позволяющее избегать конфликта имен. При составлении программного кода нам необходимо указать, какое именно пространство имен будет использовано для «хранения» названий создаваемых в программе утилит.

Комментарии

В текст программы можно добавлять комментарии. Текст комментария не подчиняется синтаксису языка C++.

Назначение комментария – пояснить некоторый фрагмент программы, хорошим стилем считается наличие 1 комментария на 10 строк кода

Комментарий до конца строки:

```
//текст комментария
```

```
Sum=Sum+Price; //увеличиваем сумму на величину Price
```

Комментарии

Многострочные комментарии выделяются символами /* и */.

Текст, заключённый в служебные символы /* и */ (в этом порядке), полностью игнорируется компилятором.

Препроцессор и директивы

Препроцессор C++ — программа, подготавливающая код программы на языке C++ к компиляции.

Директивы препроцессора представляют собой инструкции, записанные в тексте программы на C++, и выполняемые до трансляции программы. Директивы начинаются с символа # и не заканчиваются точкой с запятой.

Несколько ключевых слов:

define — создание константы или макроса;

undef — удаление константы или макроса;

include — вставка содержимого указанного файла.

Стандартная Библиотека означает коллекцию классов и функций.

Стандартная Библиотека поддерживает несколько основных контейнеров, функций для работы с этими контейнерами, объектов-функции, основных типов строк и потоков (включая интерактивный и файловый ввод-вывод).

Стандартная библиотека включает 39 заголовочных файлов (C++14), например, fstream, iostream, complex и т.д.

Стандартная библиотека c++.

Стандартная Библиотека языка C++ также включает в себя спецификации стандарта ISO C90 **стандартной библиотеки языка Си** (24 заголовочных файла).
Пример: cmath, ctime, cstdio и т.д.

Стандартная библиотека шаблонов (STL) — подмножество стандартной библиотеки C++ и содержит контейнеры, алгоритмы, итераторы, объекты-функции и т. д.
Примеры: map, set, vector и т.д.

iostream (от англ. Input/Output Stream («поток ввода-вывода»)) — заголовочный файл с классами, функциями и переменными для организации ввода-вывода в языке программирования C++. Он включён в стандартную библиотеку C++.

cmath — заголовочный файл стандартной библиотеки языка программирования C++, разработанный для выполнения простых математических операций

Вывод-ввод информации

В языке C++ для вывода информации используется специальный объект **cout,** для ввода - **cin**.

cout – **C**onsol **Out**put (вывод в консоль)
cin – **C**onsol **in**put (ввод в консоль)

Чтобы воспользоваться данными объектами необходимо подключить заголовочный файл <iostream>.

Для вывода в консоль кириллицы можно воспользоваться следующими процедурами:

```
setlocale(LC_ALL, "Russian");
```

```
system("chcp 1251 >nul");
```

Если в программе предполагается ввод и вывод русского текста, то необходимо подключить заголовочный файл <windows.h>

```
SetConsoleCP(1251);
SetConsoleOutputCP(1251);
```

Для того чтобы консоль не закрывалась сразу после выполнения программы можно воспользоваться процедурой:

```
system("pause>nul");
```

Основной тип объектов, хранящих данные в программе – переменная.

Переменная - это ячейка в памяти компьютера, которая имеет имя, тип и хранит некоторое значение.

Значение переменной может меняться во время выполнения программы. При записи в ячейку нового значения старое стирается.

Тип переменной определяет:

1) Размер ячейки памяти в байтах.
2) Интерпретацию двоичного кода.
3) Допустимые операции.

Процесс проверки и накладывания ограничений на типы используемых данных называется контролем типов или **типизацией программных данных.**

Различают следующие виды типизации:

Статическая типизация — переменная связывается с типом в момент объявления и тип не может быть изменён позже.

Динамическая типизация — приём, используемый в языках программирования, при котором переменная связывается с типом в момент присваивания значения, а не в момент объявления переменной.

Язык C++ поддерживает статическую типизацию, и типы всех используемых в программе данных должны быть указаны перед ее компиляцией.

Различают: простые (встроенные), составные и прочие типы данных.

Простые: целочисленные, с плавающей точкой (вещественные), символьные, логические.

Составные (сложные) данные:

Массив — индексированный набор элементов одного типа.

Строковый тип — массив, хранящий строку символов.

Пользовательские типы данных (структуры, классы и т. д.) .

Другие типы данных:

Указатель — хранит адрес в памяти компьютера, указывающий на какую-либо информацию, например, указатель на переменную.

Основные типы данных в C++

int — целочисленный тип данных.
float — тип данных с плавающей запятой.
double — тип данных с плавающей точкой двойной точности.
char — символьный тип данных.
bool — логический тип данных.

Целочисленные типы данных

Тип	Размер	Числовой интервал
int	4	−2147483 648÷ +2147483647
short	2	От –32 768 до +32 767
long	4	−2147483 648÷ +2147483647
long long	8	От –9 223 372 036 854 775 808 до 9 223 372 036 854 775 807
unsigned int	4	От 0 до 4 294 967 295
unsigned short	2	От 0 до 65 535

Модификатор типа – специальное ключевое слово, которое указывается перед идентификатором типа и влияет на размер памяти, выделяемой для переменной данного типа.

Модификаторы:

signed (числовое значение со знаком);
unsigned (числовое значение без знака);
long (расширение диапазона);
short (сокращение диапазона).

Узнать, сколько байт выделяется для значений определенного типа или для какой-то определенной переменной, можно с помощью оператора **sizeof**.

Пример:

```
cout << sizeof(unsigned) << endl;//4
```

Синонимы целочисленных типов данных:

short, short int, signed short, signed short int
unsigned short, unsigned short int
int, signed, signed int
unsigned, unsigned int
long, long int, signed long, signed long int
unsigned long, unsigned long int
long long, long long int, signed long long, signed long long int
unsigned long long, unsigned long long int

Целочисленные типы можно определять в битах: _int8, _int16, _int32, _int64.

Любую переменную, которая будет использована в программе, необходимо объявлять.

Определяя новую переменную в программе, программист должен присвоить ей **идентификатор**.

Идентификатор — это имя переменной, функции, класса или другого объекта в языке C++.

В языке C++ идентификатор может состоять из латинских букв (не более 32-х), арабских цифр и символа подчеркивания “_”, при этом он не может начинаться с цифры.
Можно использовать верхний и нижний регистр.

Верные идентификаторы: x, Sum1, _Count, F_I_O

Недопустимые идентификаторы: Sum$, Сумма, My Summ, 1Sum.

Объявление переменной предполагает указание имени переменной и ее типа (int и т. п.). Если при объявлении переменной одновременно выделяется память под нее, то происходит **определение переменной.**

В приведенных примерах переменные объявляются и определяются.

Примеры :

```
int x;

int main()
{  short y, z;
   …
}
long c;
```

В программе переменная объявляется в любом месте (но до первого ее использования).

Область доступности переменной определяется блоком, в котором она объявлена. Блок, в свою очередь, ограничивается парой фигурных скобок.

Если переменная объявлена в главной функции программы, она доступна в любом месте главной функции.

В C++ точка, в которой объявлено имя, является точкой, в которой он становится видимым для компилятора.

Нельзя ссылаться на переменные, объявленные в более поздней точке в единице компиляции.

Переменные должны быть объявлены как можно ближе до точки, в которой они используются.

Инициализация переменной происходит когда в переменную записывается первое значение.

1) Копирующая инициализация

```
int y=8;
```

2) Инициализация по умолчанию

```
int y;//неопределенное значение, если инициализировать в главной функции
y=10;// копирующее присваивание
```

3) Прямая инициализация с помощью круглых скобок ()

```
int x(4);
```

4) uniform-инициализация

```
int x{4};

short x{};
```

Инициализация переменной с пустыми фигурными скобками указывает на инициализацию по умолчанию (переменной присваивается 0).

В uniform-инициализации есть преимущество: нельзя присвоить переменной значение, которое не поддерживает её тип данных — компилятор выдаст предупреждение или сообщение об ошибке.

Тип данных с плавающей точкой

Тип	Размер	Числовой интервал	Точность
float	4	−2147483 648÷ +2147483647	6-9 цифр
double	8	от $\pm 5{,}0 \times 10^{-324}$ до $\pm 1{,}7 \times 10^{308}$	15–17 цифр

Примеры создания переменных типа данных с плавающей точкой

```
double pi=3.14;

float x;
x = -5.2;

long float y=4.9;
```

Модификаторы short, unsigned и signed для типа данных с плавающей точкой не используются.

Символьный тип данных

Тип данных char используется для хранения символа (управляющего или печатного) в определенной системе кодировки.

Как правило, в таких переменных хранят символ из таблицы ASCII.

При инициализации переменной типа char символ заключается в апострофы

Приме

р:

```
char y='d';
```

Для представления символов в C++ типу данных char отводится один байт. Таким образом, можно хранить в переменных символьного типа до 256 значений.

Каждому символу соответствует определённое число из диапазона [0, 255].

Логический тип данных

Переменные логического типа могут принимать только два значения: true(истина) и false(ложь).
Если присвоить логической переменной число 0, то переменная принимает значение false.
Если логической переменной присвоить любое отличное от нуля число, то она примет значение true.

```
bool y=true;
bool z=false;
```

При выводе в консоль переменная y примет значение 1, переменная z значение 0.

Логический тип данных

Переменные логического типа могут принимать только два значения: true(истина) и false(ложь).

Если присвоить логической переменной число 0, то переменная принимает значение false.

Если логической переменной присвоить любое отличное от нуля число, то она примет значение true.

Чтобы в консоли было выведено значение false или true необходимо перед потоком вывода поставить флаг **boolalpha**

```
bool y=false;
cout << boolalpha;
cout << y << '\n';
```

Константы

Константы – еще один тип объектов, способных хранить данные в программе.

Также как и переменные, константы имеют тип, идентификатор, но значения констант нельзя изменить.

Определение констант в программе:

const имя_типа идентификатор1 = конст.выражение1 [, идентификатор2 =конст. выражение2] [,…] ;

Примеры определения переменных в программе

```
const int N=5;
const int Count=100*N;
const double pi=3.14;
```

Значение констант нельзя изменять