Тема: Структура программы на C++.

Введение в структуру программы на C++

Программа на C++ — это **набор инструкций**, которые компилятор выполняет **построчно**. Каждая программа обязательно должна содержать **функцию main()**, с которой начинается выполнение.

Основные элементы структуры программы

```cpp
#include <iostream>          // подключение библиотеки ввода/вывода
using namespace std;         // позволяет использовать std без префикса

int main() {                 // главная функция — с неё начинается выполнение
    cout << "Hello, world!";  // вывод текста на экран
    return 0;                // завершение программы
}
```

- **#include <iostream>** — директива препроцессора для подключения библиотеки.
- **using namespace std** — позволяет не писать std::cout, а просто **cout**.
- **int main()** — **главная функция**, с неё начинается выполнение.
- **return 0**; — возвращает значение, сигнализирующее об успешном **завершении программы**

Пространство имён:

```
using namespace std;
```

```
std :: cout << ”Hello, World!”;
```

Позволяет не писать std::cout, std::string и т.д.
Если не использовать это:

Подключение заголовочных файлов:

```
#include <cmath>      // для математических функций, например sqrt()
#include <string>     // работа со строками
```

Заголовочные файлы содержат в себе **определения функций и объектов**, которые можно использовать в коде

Типы данных в C++

Типы данных позволяют компьютеру определить, **какой вид информации** хранится в **переменной**. В C++ существуют следующие основные типы:

int — целые числа (например, 5, -12)

float — числа с плавающей точкой (одинарной точности)

double — более точные вещественные числа

char — один символ (например, 'A')

bool — логический тип (true или false)

string — строка текста (например, "Привет")

Примеры объявления переменных

float — рекомендуется добавлять f в конце числа

char пишется в **одинарных** кавычках

string требует подключение #include <string>

```
int age = 18;
float height = 1.75f;
double pi = 3.14159;
char grade = 'A';
bool passed = true;
string name = "Anna";
```

Число 3.14 **без суффикса —** это **double**, а не float. Чтобы задать именно float, добавь **f**:
float x = 3.14f; // правильно
Без f будет лишнее преобразование типа.

Типы данных

Тип данных	Размер/байт	Диапазон значений	Пример использования
bool	1	true / false	bool isOn = true;
char	1	-128 до 127 (или 0 до 255 беззнаковый)	char letter = 'A';
int	4	-2,147,483,648 до 2,147,483,647	int age = 25;
long	4 или 8	зависит от системы: 32 или 64 бита	long h= 1000000;
long long	8	-9,223,372,036,854,775,808 до 9,223,372,036,854,775,807	long long b= 1e12;
float	4	~ ±3.4e±38 (точность ~6-7 знаков)	float pi = 3.14f;
double	8	~ ±1.7e±308 (точность ~15-16 знаков)	double e = 2.71828;
long double	8, 12, 16	зависит от компилятора/платформы	long double p= 1.618L;
unsigned int	4	0 до 4,294,967,295	unsigned int c= 100;

Переменные и типы данных

Каждая **переменная** имеет **тип данных**. C++ - **строго** типизированный язык.

```cpp
int a = 10;
float b = 3.14;
char c = 'A';
bool d = true;
string name = "Alice";
```

```cpp
#include <iostream>
using namespace std;
int main() {
    int x;
    cout << "Введите число: ";
    cin >> x;
    cout << "Вы ввели: " << x;
    return 0;
}
```

Ввод и вывод

- **cin** — **ввод** данных с клавиатуры.
- **cout** — **вывод** на экран.
- << и >> — операторы **потока**.

Ввод и вывод переменных

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

int main() {
    int age;
    string name;

    cout << "Введите имя: ";
    cin >> name;

    cout << "Введите возраст: ";
    cin >> age;

    cout << "Привет, " << name << "! Возраст: " << age << " лет." << endl;
    return 0;
}
```

- **cin** используется для ввода
- **cout** используется для вывода
- **<<** — оператор вывода
- **>>** — оператор ввода

Особенности string и getline()

```
cout << "Введите возраст: ";
    cin >> age;
    cin.ignore(); // Очищаем символ новой строки из буфера

    cout << "Введите полное имя: ";
    getline(cin, fullName);
```

Если строка содержит **пробелы**, используем **getline()**:
После cin, перед getline() желательно добавить cin.ignore().