Конспект первой лекции

Приходько Павел

05.09.2025

1 Переменные: определение и объявление

```
int x = 9;    // определение
int y;        // объявление
extern int z; // объявление ( о том что переменная есть в другом файле)
```

- Глобальная переменная по умолчанию инициализируется 0.
- Определение создаёт переменную и выделяет память.
- Объявление сообщает компилятору, что переменная есть.
- Extern говорит компилятору, что переменная находится в другом файле.

2 Функции: объявление и определение

```
void f(); // объявление
void f(); // повторное объявление

int g()
{
    return 5;
}
```

- Объявление функции сообщает компилятору имя и параметры. (Допустимы множественные объявления, в отличие от определений)
- Определение функции содержит реализацию.
- Возвращаемый тип не входит в сигнатуру (нельзя перегружать только по return type).

3 Пространства имён (namespace)

```
namespace test // Тестовый неймспейс
{
    int x = 0;
}

namespace test // Дополнение
{
    int y = 0;
}

namespace     // Анонимный неймспейс
{
    int z = 111;
}
```

- namespace помогает организовывать код и избегать конфликтов имён.

- Одно пространство имён можно расширять (см. два блока namespace test).
- Существуют анонимные namespace, ограничивающие область видимости внутри файла. Т.е. z будет доступно только внутри текущего файла.

4 Главная функция `main`

4.1 Переменные из локальной, глобальной области и namespace

```
int main()
{
    int x = 5;                          // local переменная
    std::cout << x << std::endl;        // вывод local переменной
    std::cout << ::x << std::endl;      // вывод global переменной
    std::cout << test::x << std::endl; // вывод переменной из namespace
}
```

- Локальная переменная перекрывает глобальную с тем же именем.
- Чтобы обратиться к глобальной переменной, используется оператор `::`.
- Для переменных в namespace указываем `имя_пространства::имя`.

5 Изменение переменной из namespace

```
int main()
{
    test::x = 1;
    test::x += 1;
    std::cout << test::x << std::endl;
}
[Output]: 2
```

- Переменные внутри namespace можно изменять и к ним можно обращаться через `имя_пространства::`.

6 Определение функции (которую мы раньше объявили)

```
void f()
{
    std::cout << "Keke" << std::endl;
}
```

- Определение функции `f`, которую мы объявили в начале файла, может находиться даже в конце файла.

7 Вызов функции с `void`-возвратом

```
    std::cout << f(); // нельзя
```

- Функции, возвращающие `void`, нельзя передавать в `std::cout`.
- Для вызова такой функции достаточно написать `f();`.