

# Тема 1-5.

## Локальные и глобальные переменные

для АСУБ и ЭВМб



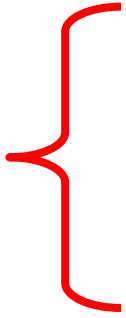
# Определения

- **Переменная** – это именованная область памяти.
- **Имя переменной** – это *ссылка* на некоторую область *памяти*.
- **Типизированная переменная**:
  - диапазон возможных значений;
  - способ хранения в памяти;
  - допустимые операции.





Локально



# Переменные по области действия

- **Область действия (ОД) идентификатора (имени)** - это часть программы, в которой можно обращаться к данному имени (сфера действия имени в программе):
  - **Глобальная переменная** — переменная, определённая вне функций и доступная из разных функций. **ОД** глобальных переменных - вся программа от точки их определения или от точки их описания
  - **Локальная переменная** — переменная, определённая внутри некоторого блока и доступная только из него. **ОД** от точки определения до конца блока, либо весь блок (не в C++)

# Переменные по области видимости

- **Область видимости (ОВ)** – это часть программы, в которой обращение к имени переменной позволяет обратиться к участку памяти, связанному с данной переменной:
  - Переменная внутри вложенного блока может иметь то же имя, что и переменная внутри внешнего блока. Когда подобное случается, то переменная во вложенном (внутреннем) блоке «скрывает» внешнюю переменную. Это называется **сокрытием имен**.
- **ОВ≤ОД**

# Области действия и ВИДИМОСТИ

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

int x = -1; //глобальная

void func(int x); // в прототипе

int main()
{
    setlocale(LC_ALL, "Russian");

    int x = -100;
    func(100);
    func(200);

    cout << "из main: " << x << endl;
    cout << "глобально: " << ::x << endl;

    return 0;
}

void func(int x) // в определении
{
    //int x; //переопределение!!!
    cout << "входной параметр из func: " << x << endl;
    {
        int x; //инициализация
        cout << "в фиктивном блоке из func : " << x << endl;
        for (int x = 10; x < 13; x++) //в блоке
        {
            cout << "в блоке оператора " << x << endl;
        }
    }
}
```



Переменная – именованная область памяти

Идентификатор

Как работать с памятью?  
• Есть компьютер и программист  
• Есть этап создания и этап выполнения

Область действия (ОД) идентификатора в программе

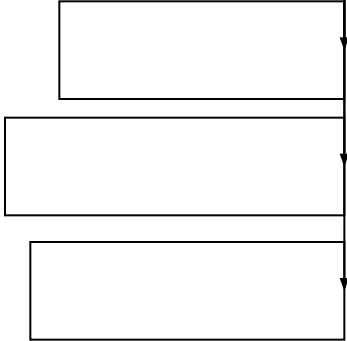
Повторное определение идентификатора:  
**ОВ ≤ ОД**

Область видимости (ОВ) идентификатора

Локально



- Блок
- В определении функции
- В прототипе функции
- Глобально
- Метки операторов



Переменная – именованная область памяти

Идентификатор

Класс хранения – способ выделения памяти под переменную, также определяет время её жизни

Область действия (ОД) идентификатора в программе

Повторное определение идентификатора:  
 $ОВ \leq ОД$

Статическое

Автоматическое

Динамическое

Блок

В определении функции

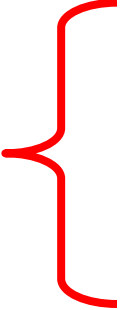
В прототипе функции

Глобально

Метки операторов

Область видимости (ОВ) идентификатора

Локально



- Память под статические переменные распределяется **компилятором** и выделяется при запуске программы, а освобождается при завершении программы
- Все глобальные переменные — статические
- Для определения статической локальной переменной используется ключевое слово **static**

# + СТАТИЧЕСКАЯ ПЕРЕМЕННАЯ

- `#include <iostream>`
- `using namespace std;`
- `int x = -1; //глобальная`
- `void func(int x); // в прототипе`
  
- `int main()`
- `{`
- `setlocale(LC_ALL, "Russian");`
  
- `int x = -100;`
- `func(100);`
- `func(200);`
  
- `cout << "из main: " << x << endl;`
- `cout << "глобально: " << ::x << endl;`
  
- `return 0;`
  
- `}`
  
- `void func(int x) // в определении`
- `{`
- `//int x;`
- `cout << "входной параметр из func: " << x << endl;`
- `{`
- `static int x; //инициализация`
- `cout << "в фиктивном блоке из func и статичная : " << x++ << endl;`
- `for (int x = 10; x < 13; x++) //в блоке`
- `{`
- `cout << "в блоке оператора " << x << endl;`
- `}`
- `}`
- `}`
- `}`



# Глобальные переменные

- Оператора разрешения области видимости (::)
  - Если используются одинаковые имена для локальных и глобальных переменных

```
#include <iostream>
```

```
int value(4); // глобальная переменная
```

```
int main()
```

```
{
```

```
    int value = 8; // эта переменная (локальная) скрывает значение глобальной пере
```

```
    value++; // увеличивается локальная переменная value (не глобальная)
```

```
    ::value--; // уменьшается глобальная переменная value (не локальная)
```

```
    std::cout << "Global value: " << ::value << "\n";
```

```
    std::cout << "Local value: " << value << "\n";
```

```
    return 0;
```

```
}
```

# Глобальные переменные

- Если вы хотите сделать глобальную переменную **внутренней** (которую можно использовать только внутри одного файла) — используйте **ключевое слово static**:

```
#include <iostream>
```

```
static int g_x; // g_x - это статическая глобальная переменная,  
               // которую можно использовать только внутри этого файла
```

```
int main()  
{  
    return 0;  
}
```

# Глобальные переменные

- Если вы хотите сделать глобальную переменную внешней (которую можно использовать в любом файле программы) — используйте **ключевое слово extern**:

```
#include <iostream>
```

```
extern double g_y(9.8); // g_y - это внешняя глобальная переменная и её  
можно использовать и в других файлах программы
```

```
int main()  
{  
    return 0;  
}
```



# Глобальные переменные

- По умолчанию, **неконстантные** переменные, объявленные вне блока, считаются внешними.
- Однако константные переменные, объявленные вне блока, считаются внутренними.

# Глобальные переменные

- Чтобы использовать внешнюю глобальную переменную, которая была объявлена в другом файле, нужно записать предварительное объявление переменной с использованием ключевого слова **extern** (без инициализируемого значения)
- Если предварительное объявление находится вне **блока**, то оно применяется ко всему файлу. Если же внутри блока, то оно применяется только к нему.
- Если переменная объявлена с помощью ключевого слова **static**, то получить доступ к ней с помощью предварительного объявления не получится.

# Глобальные переменные

global.cpp

```
// Определяем две глобальные переменные  
int g_m; // неконстантные глобальные переменные имеют внешнюю связь  
         //по умолчанию  
int g_n(3); // неконстантные глобальные переменные имеют внешнюю связь  
           //по умолчанию  
// g_m и g_n можно использовать в любом месте этого файла
```

main.cpp

```
#include <iostream>
```

```
extern int g_m; // предварительное объявление g_m. Теперь g_m можно  
                //использовать в любом месте этого файла
```

```
int main()
```

```
{
```

```
  extern int g_n; // предварительное объявление g_n. Теперь g_n можно  
                  //использовать только внутри main()
```

```
  g_m = 4;
```

```
  std::cout << g_n; // должно вывести 3
```

```
  return 0;
```

```
}
```

# Глобальные переменные

- Переменные `g_m` и `g_n` имеет *файловую область видимости* внутри `global.cpp`.
- Доступ к этой переменной вне файла `global.cpp` отсутствует.
- Хотя эта переменная и используется в `main.cpp`, сам `main.cpp` не видит её, он видит только предварительные объявления).
- Компоновщик отвечает за связывание определения `g_m` и `g_n` в `global.cpp` с использованием `g_m` и `g_n` в `main.cpp`.

# Связи функций

- Функции имеют такие же свойства связи, что и переменные. По умолчанию они имеют внешнюю связь, которую можно сменить на внутреннюю с помощью ключевого слова `static`
- Предварительные объявления функций не нуждаются в ключевом слове `extern`. Компилятор может определить сам (по телу функции): определяете ли вы функцию или пишете её прототип.