Область видимости, время жизни

Область видимости

Область видимости (scope) имени – это часть текста программы, в пределах которой имя может быть использовано.

В языке Си выделяют следующие области видимости

- блок;
- файл;
- функция;
- прототип функции.

Область видимости «блок»

В языке Си *блоком* считается последовательность объявлений, определений и операторов, заключенная в фигурные скобки.

Существуют два вида блоков:

- составной оператор;
- определение функции.

Блоки могут включать в себя составные операторы, но не определения функций.

Область видимости «блок»

- Переменная, определенная внутри блока, имеет область видимости в пределах блока.
- Формальные параметры функции имеют в качестве области видимости блок, составляющий тело функции.

```
double f(double a)  // начало области видимости переменной a
{
    double b;       // начало области видимости переменной b
    ...
    return b;
}                   // конец области видимости переменных a и b
```

Область видимости «файл»

- Область видимости в пределах файла имеют имена, описанные за пределами какой бы то ни было функции.
- Переменная с областью видимости в пределах файла видна на протяжении от точки ее описания и до конца файла, содержащего это определение.
- Имя функции всегда имеет файловую область видимости.

Область видимости «файл»

```
#include <stdio.h>

int max;

void f(void)
{
    printf("%d\n", max);
}

int main(void)
{
    max = 5;
    f();
    ...
}
```

Область видимости «функция»

- Метки - это единственные идентификаторы, область действия которых - функция.
- Метки видны из любого места функции, в которой они описаны.
- В пределах функции имена меток должны быть уникальными.

Область видимости «функция»

```
void foo(void)
{
    int i = 0;

loop:
    printf("%d\n", i);
    i++;
    if (i < 5)
        goto loop;
}
```

Область видимости «прототип функции»

Область видимости в пределах прототипа функции применяется к именам переменных, которые используются в прототипах функций.

```
int f(int i, double d);
```

Область видимости в пределах прототипа функции простирается от точки, в которой объявлена переменная, до конца объявления прототипа.

```
int f(int, double);         // ок
int f(int i, double i);     // ошибка компиляции
```

Правила перекрытия областей видимости

Переменные, определенные внутри некоторого блока, будут доступны из всех блоков, вложенных в данный.

```
{
    int a = 1;
    ...
    {
        int b = 2;
        ...
        printf("%d %d\n", a, b); // ок
    }

    printf("%d %d\n", a, b); // ошибка компиляции
}
```

Правила перекрытия областей видимости

Возможно определить в одном из вложенных блоков переменную с именем, совпадающим с именем одной из "внешних" переменных.

```
{
    int a = 1;

    {
        double a = 2.0;

        printf("%g\n", a);      // 2
    }

    printf("%d\n", a);          // 1
}
```

Время жизни

Время жизни (storage duration) – это интервал времени выполнения программы, в течение которого «программный объект» существует.

В языке Си время жизни «программного объекта» делится на три категории

- глобальное (по стандарту - статическое (англ. static));
- локальное (по стандарту - автоматическое (англ. automatic));
- динамическое (по стандарту - выделенное (англ. allocated)).

Глобальное время жизни

Если «программный объект» имеет глобальное время жизни, он существует на протяжении выполнения всей программы.

Примерами таких «программных объектов» могут быть функции и переменные, определенные вне каких либо функций.

Локальное время жизни

Локальным временем жизни обладают «программные объекты», область видимости которых ограничена блоком.

Такие объекты создаются при каждом входе в блок, где они определяются. Они уничтожаются при выходе из этого «родительского» блока.

Примерами таких переменных являются локальные переменные и параметры функции.

Динамическое время жизни

Время жизни «выделенных» объектов длится с момента выделения памяти и заканчивается в момент ее освобождения.

В Си нет переменных, обладающих динамическим временем жизни.

Динамическое выделение выполняется программистом «вручную» с помощью соответствующих функций. Единственные способ «добраться» до выделенной динамической памяти – использование указателей.