9. ОБЛАСТЬ ВИДИМОСТИ

Если, используя имя, можно получить доступ к элементу, с которым это имя сопоставлено, то говорят, что данное имя находится в области своей видимости.

Область видимости является подобластью области существования имени. Если элемент языка, чьё имя находится в области своего существования, тем не менее недоступен по этому имени, то будем говорить, что это имя **скрыто** или **замаскировано**.

Глобальные имена видимы от точки их объявления до конца файла, если они не замаскированы локальными именами.

Переменные из объемлющих блоков, как и глобальные, видимы во внутренних блоках.

Если переменная, объявленная внутри блока, имеет то же имя, что и переменная объемлющего уровня, то имя объемлющего уровня маскируется, и определение переменной в блоке заменяет определение объемлющего уровня на протяжении всего блока. Видимость замаскированной переменной восстанавливается при выходе из блока. Метки в функции видимы во всём теле функции.

```
int i = 3;
{
int c = i;        // c становится равным 3;
…
int i = 0;        // имя i маскирует внешнее имя i;
cout <<"c = "<< c
<<",   i= "
<< i <<".\n";
}                 // конец области существования имен i, c из блока;
                  // опять видно имя i, объявленное перед блоком. cout
<<" i = "<< i << ".\n";
…
```

Здесь будет выведено:

```
 c = 3,   i = 0.
i = 3.
```

Если скрытым именем является глобальное имя функции или объекта, то обратиться к нему можно, используя операцию **разрешения доступа**, или **разрешения контекста ::**.

```
int i=5;          // Глобальная переменная;

void main( ){ int i=1;      // Локальная
переменная. i++; ::i++;
cout << "i = " << i << ",\n";
cout << "глобальное i = " << ::i<< ".\n";
}
```

Здесь будет напечатано:

```
i = 2, глобальное
i = 6.
```

С помощью операции **::** нельзя обратиться к скрытому локальному объекту.