Составной оператор, локальные и глобальные переменные

Во всех управляющих структурах, где нужны вложенные операторы, таких, как, например, условный оператор или операторы цикла, разрешается использование в качестве вложенного лишь одного оператора. Однако часто нужно выполнить в случае истинности некоторого условия несколько операторов, или в цикле выполнить более одного оператора. Для этих случаев в C++ предусмотрена конструкция, называющаяся «составной оператор». Она состоит из последовательности операторов, заключенной в фигурные скобки. Действие такого оператора состоит в последовательном выполнении входящих в его состав операторов.

Операторы в составном операторе не разделяются ничем (еще раз напомним, что в C++ символ «;» не разделяет операторы, как в Pascal, а завершает оператор, сделанный из выражения, и ряд других операторов). В частности, это означает, что обе «;» в следующем фрагменте необходимы:

```
{ i = 0; j = 1; }
```

Правда, используя операцию последования, этот фрагмент можно переписать эквивалентным образом и без составного оператора:

```
i = 0, j = 1;
```

В отличие от Pascal, в C++ можно определять переменные в любом составном операторе (почти в любом месте), а не только в теле какой-либо функции.

Так определенные переменные видимы (на них можно сослаться) только в том составном операторе, где они объявлены, почему они и называются локальными.

Локальные переменные бывают двух видов:

а) Автоматические (по умолчанию) — такие переменные создаются (и инициализируются) в начале выполнения того составного оператора, в котором они определены, и уничтожаются, когда этот составной оператор закончит свою работу.

б) Статические (в декларации перед типом стоит ключевое слово static) — такие переменные создаются в начале работы программы; они инициализируются, когда первый раз выполняется их декларация (когда она выполняется второй и более раз, инициализация игнорируется); они уничтожаются, когда программа завершит свою работу. В частности, они сохраняют свои значения между периодами работы такого составного оператора. Так что при, скажем, повторном выполнении их составного оператора в начале их значения будут такими, какими они были в конце работы первого его выполнения.

Если один составной оператор вложен в другой, то вполне возможна ситуация, когда в каждом из них определена локальная переменная с одним и тем же именем. В этом случае, во внутреннем составном операторе имя этой переменной будет означать ту из этих переменных, которая определена во внутреннем составном операторе и скрывает ту, которая определена во внешнем.

Здесь уместно сказать и о так называемых глобальных переменных. Под глобальными переменными понимаются переменные, определенные в одном из исходных файлов программы, вне любого составного оператора. Обычно, такие переменные доступны во всей программе, но чтобы пользоваться ими в других исходных файлах, в них должно встречаться объявление такой переменной (та же декларация, но перед ней стоит ключевое слово extern и нет инициализации — обычно помещается в соответствующий заголовочный файл).

Если есть одноименные глобальная и локальная переменные, в своем составном операторе локальная переменная скрывает глобальную, но к глобальной можно обратиться, поставив перед ее именем операцию разрешения области видимости (::). Увы, если есть два составных оператора, один из которых вложен в другой, и в обоих определены одноименные переменные, из внутреннего невозможно обратиться к переменной из внешнего.

Наконец, скажем здесь и о внутреннем и внешнем связывании. Это понятие применяется к именам глобальных переменных и функций, определенных в программе. Если какое-то имя глобальной переменной подлежит внешнему связыванию, то такая переменная будет доступна из других файлов программы (конечно, если она объявлена в них). Но если у нас есть две одноименные глобальные переменные, каждая из которых подлежит внешнему связыванию, даже из разных исходных файлов, компоновщик выдаст ошибку — конфликт имен. Чтобы этого не происходило, можно сделать так, чтобы по крайней мере одна из конфликтующих переменных подлежала внутреннему связыванию, поставив перед ее определением ключевое слово static (это ключевое слово для локальных и глобальных переменных означает совсем разные вещи). Такое решение проблемы конфликта имен, правда, приведет к тому, что та глобальная переменная, которая подлежит внутреннему связыванию, будет видна только в том файле, в котором она определена.

Что нужно знать в результате (проверяется на зачете):

- Синтаксис и смысл составного оператора.
- Зачем нужен составной оператор?

- Способ обойтись без составного оператора, где это возможно.
- Что такое локальные переменные и какими они бывают?
- Определение, область видимости и время жизни автоматических локальных переменных.
- Определение, область видимости и время жизни статических локальных переменных.
- Что такое глобальные переменные и какими они бывают?
- Когда одна переменная скрывает другую?
- Когда и как можно обратиться к скрытой переменной?
- Что такое внутреннее и внешнее связывание, как его осуществить для конкретной глобальной переменной?
- Конфликт имен и способ его избежать, последствия применения данного способа.

Что нужно уметь в результате (задача на контрольной работе):

- Писать в программе составные операторы.
- Определять в составных операторах автоматические и статические локальные переменные.
- Определять и объявлять в программе глобальные переменные внешнего и внутреннего связывания.
- Обращаться из составного оператора к скрытым глобальным переменным.