Flow of Control

The bulk of this lecture is still in power-point form

Compound Statements

In many of the syntax definitions we use to define flow-of-control you will see the term *statement*

It is important to understand that whenever you see it it can mean

1. a single statement
2. a block of statements (surrounded by `{ }`)

These can be slightly tricky to define. Here's a partial grammar for various kinds of statements

We've seen the declaration-statement in the section on variables.

statements :

1. declaration-statement:

Type *Identifier* **;** **|**
Type *Identifier* **=** *Value* **;**

2. executable-statement:

$\textbf{\textit{expression}}_{\textbf{\textit{opt}}}$ **;|**
return $expression_{opt}$ **;|**
other executable statement

3. single-statement:

declaration-statement **|**
executable-statement

4. statement-sequence:

single-statement **|**
statement-sequence
single-statement

5. statement-block:

{*statement sequence*$_{opt}$**}**

6. statement:

executable-statement **|**
statement-block

Example:

```
{   // open block
    double pi=3.14159; //dec
    cout << pi;     // exec
}   // close block
```

Interpretation: The second line is a declaration-statement. The 3rd line is an executable statement. 2 and 3 together constitute a statement-sequence and lines 1-4 are a statement-block. Thus line 3 is a statement as is lines 1-4 taken together.

Let's just take a little more extensive example:

```
double pi = 3.14159;
double r = 2.4;
double y;
y = 4*pi*r*r/3;
cout<<"The area is " << y << endl;
```

Now we try to categorize it according to the grammar

1. By Rule 1 the 1st and 2nd lines are *declaration statements* (of the second type)
2. By Rule 1 the 3rd line is a *declaration statement* of the first type
3. By Rule 2 the 4th line is an *expression* followed by a `;` and so is an *executable statement* of the first type.
4. The 5th line is less obvious. Actually it's the same as the 4th because `cout<<"The area is " << y << endl` *is technically an expression!*

Now it gets interesting

5. By Rule 3 every one of the five lines is also a *single statement.*
6. By Rule 4 (1st type) the 1st line (and actually all the others) is a *statement sequence*.
7. By Rule 5 (2nd type) line 1 & 2 together also form a *statement sequence*
8. Applying Rule 5 *recursively*, we see that lines 1,2 & 3 also form a *statement sequence* and we keep going until we run out of lines at which point
9. All 5 lines together form a *statement sequence*

Here are a bunch of examples. Try to decide, based on the grammar above, exactly what each of them corresponds to (it may be more than one).

```
{}

;

{
y=3.5;
}

x
```

1. declaration-statement
2. executable-statement
3. single statement
4. statement-sequence
5. block

6. statement
7. none of the above (syntax error?)

This page last updated on Monday, February 2, 2004