

II.3. Program Flow

statements

Statements control the sequence of execution of a program or evaluate an expression. A statement can be on several lines, but must always end with a semi-colon.

blocks

Any place you can put a statement, you can also put a block of statements. Blocks start with a '{' and end with a '}', with statements in between. Blocks do not need to end with a semi-colon.

For clarity, always indent statements in a block.

Any variables declared in a block have “local scope” (i.e. can only be used within that block or its sub-blocks).

the “if” statement

The “if” statement allows you to test a condition and branch to different code, depending on the result. The syntax is:

```
if (expression)
    statement;    // executed if expression evaluates to true

or

if (expression)
    statement1;    // expression evaluates to true
else
    statement2;    // expression evaluates to zero (false)
```

Try out this program to print a message based on the letter that is input. Call it “food.cc”.

```
#include <iostream.h>

using namespace std;

int main()
{
    char food_letter;
    cout << "Type in a letter between 'a' and 'd': ";
    cin >> food_letter;
    if (food_letter == 'a')
        cout << "Apple";
    else if (food_letter == 'b')
        cout << "Banana";
    else if (food_letter == 'c')
```

```

    {
        cout <<"Carrot";
        cout << " and Corn";
    }
    else if (food_letter == 'd')
        cout <<"Donut";
    else
        cout <<"I don't know any foods that";

    cout << " start(s) with '" << food_letter << "'." << endl;
    return 0;
}

```

looping

Looping is used to repeat the same code many times, usually with an incremental change in parameter values. The syntax for a “for” loop is:

```

for (initialization; test; action)
    statement;

```

The initialization code is called first, then the test condition is evaluated. If it evaluates to a non-zero value then (i) the statement is executed, (ii) the action is executed and the (iii) test is evaluated again. This continues until the test evaluates to zero (false).

```

#include <iostream.h>

using namespace std;

int main()
{
    char food_letter;

    for (food_letter='a'; food_letter <= 'e'; food_letter++)
    {
        if (food_letter == 'a')
            cout << "Apple";
        else if (food_letter == 'b')
            cout <<"Banana";
        else if (food_letter == 'c')
        {
            cout <<"Carrot";
            cout << " and Corn";
        }
        else if (food_letter == 'd')
            cout <<"Donut";
        else
            cout <<"I don't know any foods that";
        cout << " start(s) with '" << food_letter << "'." << endl;
    }
    return 0;
}

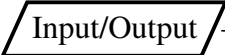
```

If you do not require the initialization code or action, look into using the “while” statement. If you require the statement to be executed at least once, look into using the “do-while” statement.

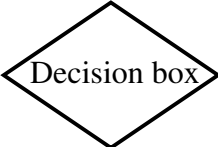
flow charts

Program flow can be represented by a flow chart.

 – 1 start box and 1 or more end boxes:

 – reads/prints variable values:

 – performs calculation or sub-task:

 – branches according to a condition:

E.g. food.cc can be represented by: