Problem Statement

Given three numbers, sort the numbers in order.

Problem Analysis

Input : first, second, third - three input numbers

Output: first, second, third - numbers sorted in order

Assumptions: The numbers are integers
Sort the numbers in ascending order

Program Design

Approach: Compare and swap the number until they are in the right order.

Initial Algorithm:

1. Read in first, second and third.
2. Compare and swap first and second.
3. Compare and swap second and third.
4. Compare and swap first and second.
5. Output the sorted numbers first, second and third.

Refinement of the initial algorithm

1. Read in first, second and third.

2. Compare and swap first and second.
   - If first > second then swap first and second.

3. Compare and swap second and third.
   - If second > third then swap second and third.

4. Compare and swap first and second.
   - If first > second then swap first and second.

5. Output the sorted numbers first, second and third.

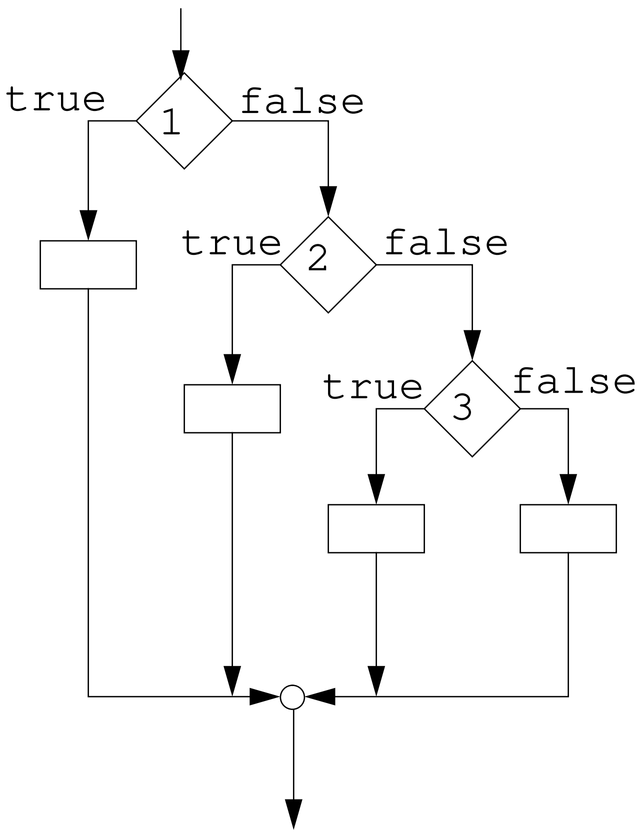
Implementation:

```
#include <iostream.h>

void main ( ) {
        int first, second, third;              // input numbers
        int temp;                              // temporary space

        // Read in first, second and third.
        cout << "Enter three numbers: ";
        cin >> first >> second >> third;

        // Compare and swap first and second .
        if (first > second) {                  // swap two numbers if needed
                temp = first;
                first = second;
                second = temp;
        }

        // Compare and swap second and third.
        if (second > third) {                  // swap two numbers if needed
                temp = second;
                second = third;
                third = temp;
        }

        // Compare and swap first and second.
        if (first > second) {                  // swap two numbers if needed
                temp = first;
                first = second;
                second = temp;
        }

        // Output the sorted numbers first, second and third.
        cout << "The sorted numbers are "
                << first << " , " << second << " , " << third << endl;
}
```

If Statement (Cont'd)

- Syntax (Cont'd)

```
if (<condition 1>) {
  <sequence of statements>
} else if (<condition 2>) {
  <sequence of statements>
} else if (<condition 3>) {
  <sequence of statements>
•••
else {
  <sequence of statements>
}
```



If Statement (Cont'd)

- Example

```
if (TotalScore >= 80)
  Grade = "A";
else if (TotalScore >= 70)
  Grade = "B";
else if (TotalScore >= 60)
  Grade = "C";
else if (TotalScore >= 50)
  Grade = "D";
else /* TotalScore < 50 */
  Grade = "F";
```

DANGLING else PROBLEM

- Consider the following statement:

```
if <C1>  if <C2>  <A>  else <B>
```

  - Does it mean this?

```
if <C1>  {
   if <C2>  <A> else <B>
}
```

  - Or this?

```
if <C1> {
      if <C2>  <A>
 }
 else <B>
```

- C++ uses the first interpretation: an else is associated with the most recent if. However, adding redundant { } often helps to clarify the meaning.

Nested if statements (example)

E.g. If the driver is male then
    If the driver is under 21,
        the insurance premium is $800
    If the driver is 21 or older,
        the insurance premium is $400

Two possible interpretations:

```
if  (gender == 'M')
    if (age < 21)
        premium = 800.0;
else
    premium = 400.0;
```

```
if  (gender == 'M')
    if (age < 21)
        premium = 800.0;
    else
        premium = 400.0;
```

If the driver is male then
    If the driver is under 21,
        the insurance premium is $800
If the driver is female,
        the insurance premium is $400

Nested If statement

Example:

```
C = 1;
B = 1;
if (A > 0)
     if (B > 0)
          C = 2;
else
     C = 3;
```

```
Suppose A = -1, what is the value of C?
```

Short-circuit Evaluation

A && B = if A then B else FALSE

(This avoids evaluation of B in the case where A isfalse.)

A || B : if A then TRUE else B

E.g.

(I != 0) && (K/I > L)

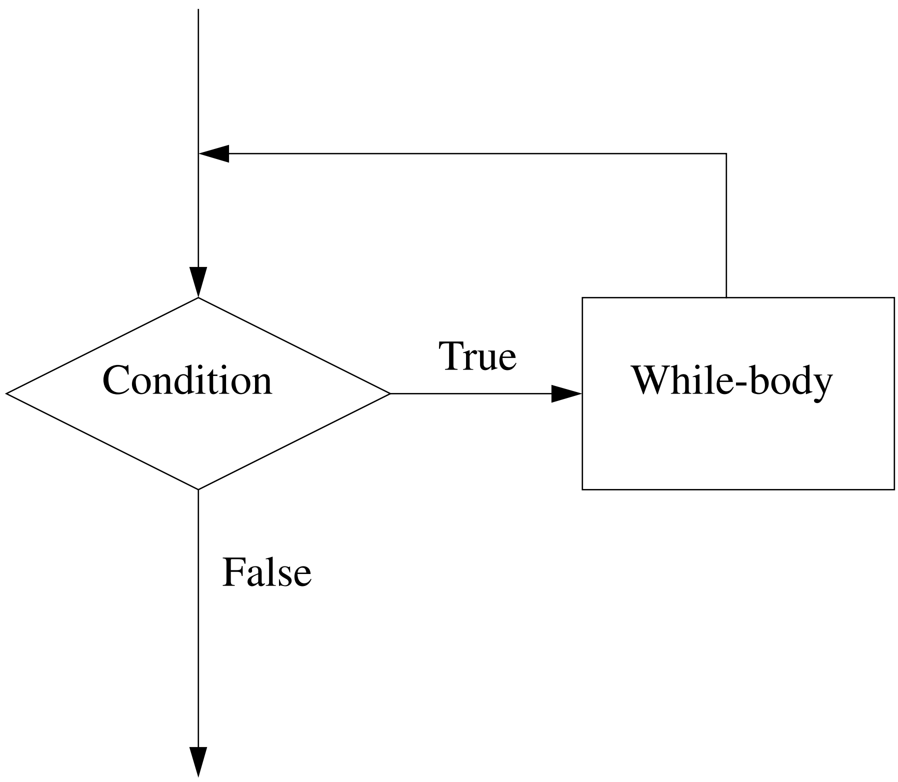
- In some programming languages (e.g. Pascal) a logical expression is always fully evaluated.

WHILE Loop

while (< logical expression >) {

< sequence of statements >

}



1. The logical expression is evaluated. If its value is TRUE, then the body of the while loop will be executed.

2. At the end of the body of the while loop, the control will go back to the beginning of the while loop.

3. Step 1 and Step 2 will iterate until the logical expression evaluates to FALSE.

While Loop (Example)

```
// Program to find the maximum value of a list of numbers
// Input:    A list of numbers
// Output:  Maximum value of the numbers entered

void main( )
{
int  value,                                  //Input value
     maxval;                                 //Maximum value

      value = 0;                             // Initialize value
      maxval = 0;                            // Initialize maxval

      while  (value != -1) {
        cout << "Enter a value (-1 to stop): ";
        cin >> value;

        if (value > maxval)
           maxval = value;                   // new maximum
      }

      cout << "The maximum value found is "
          << maxval << endl;
}
```