

Detailed Design

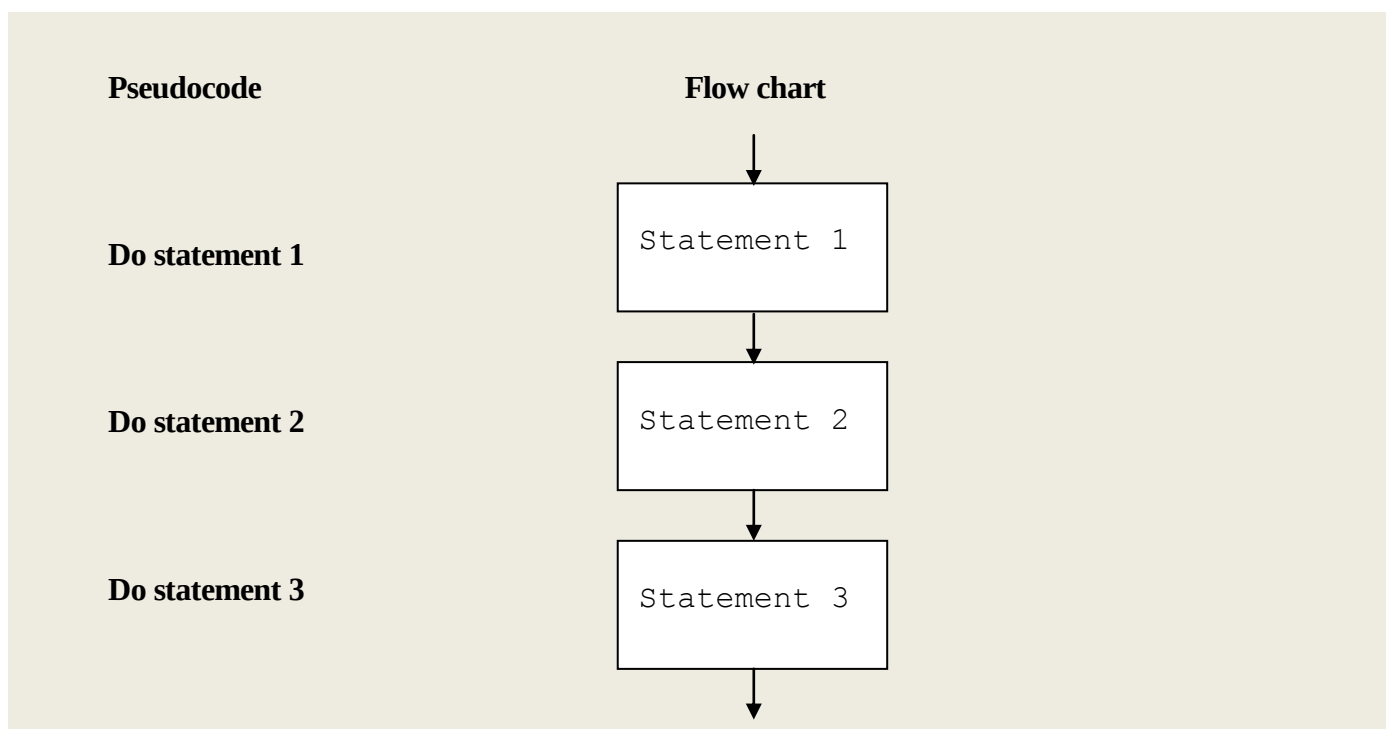
At this stage the details of how the system is going to be built are considered. An algorithm is developed which shows how the problem is to be solved. Before an algorithm can be developed a standard notation is required, and some basic rules must be followed. Algorithms can be presented using a number of notations. On this course pseudocode is going to be used. This is a form in which the structure of a high level language is used together with a limited natural language vocabulary. The rules to be followed are those developed by Bohm and Jacopini in the mid-sixties. They showed that any algorithm can be developed by combining three primitive structures :

- 1) sequence
- 2) selection
- 3) repetition

The pseudocode for each of these structures is given below, together with the flow chart representations.

Sequence

This is the simplest of the structures, each of the commands is executed in turn.

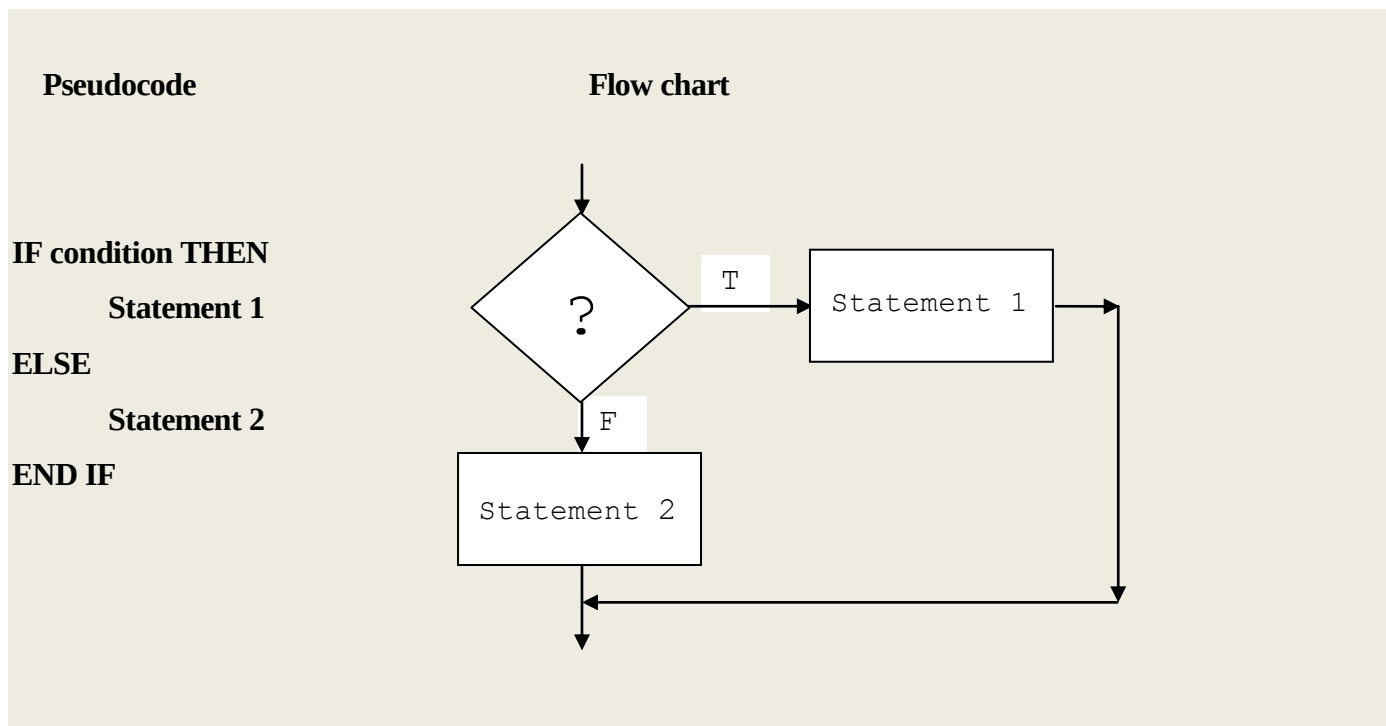


Selection

This construct allows the computer to make a decision, to choose between two or more possibilities. Two forms are given here :

- 1) the IF statement is normally used to select one of two possible alternatives.
- 2) the CASE statement is normally used when there are more than two possibilities.

IF statement



The condition in the IF statement above will be a Boolean expression, i.e. an expression which is either true or false.

Statement 1 will be executed when the expression is true.

Statement 2 will be executed when the expression is false.

CASE statement

Pseudocode

CASE expression OF

A : statement 1

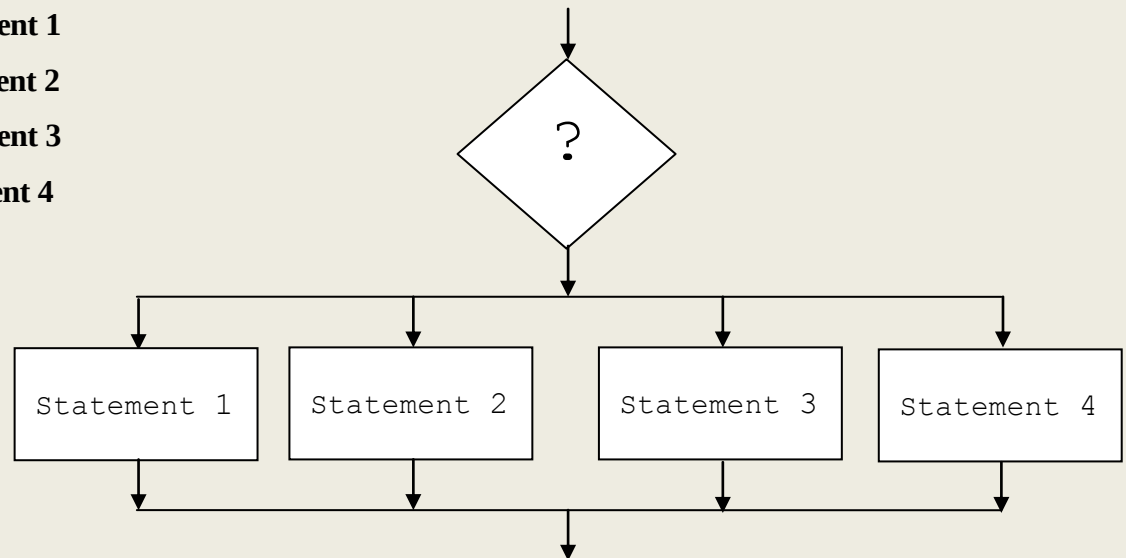
B : statement 2

C : statement 3

D: statement 4

END CASE

Flow chart



The statement executed depends upon the value of the expression.

If this value matches A then statement1 would be executed.

If this value matches A then statement1 would be executed.

If this value matches A then statement1 would be executed.

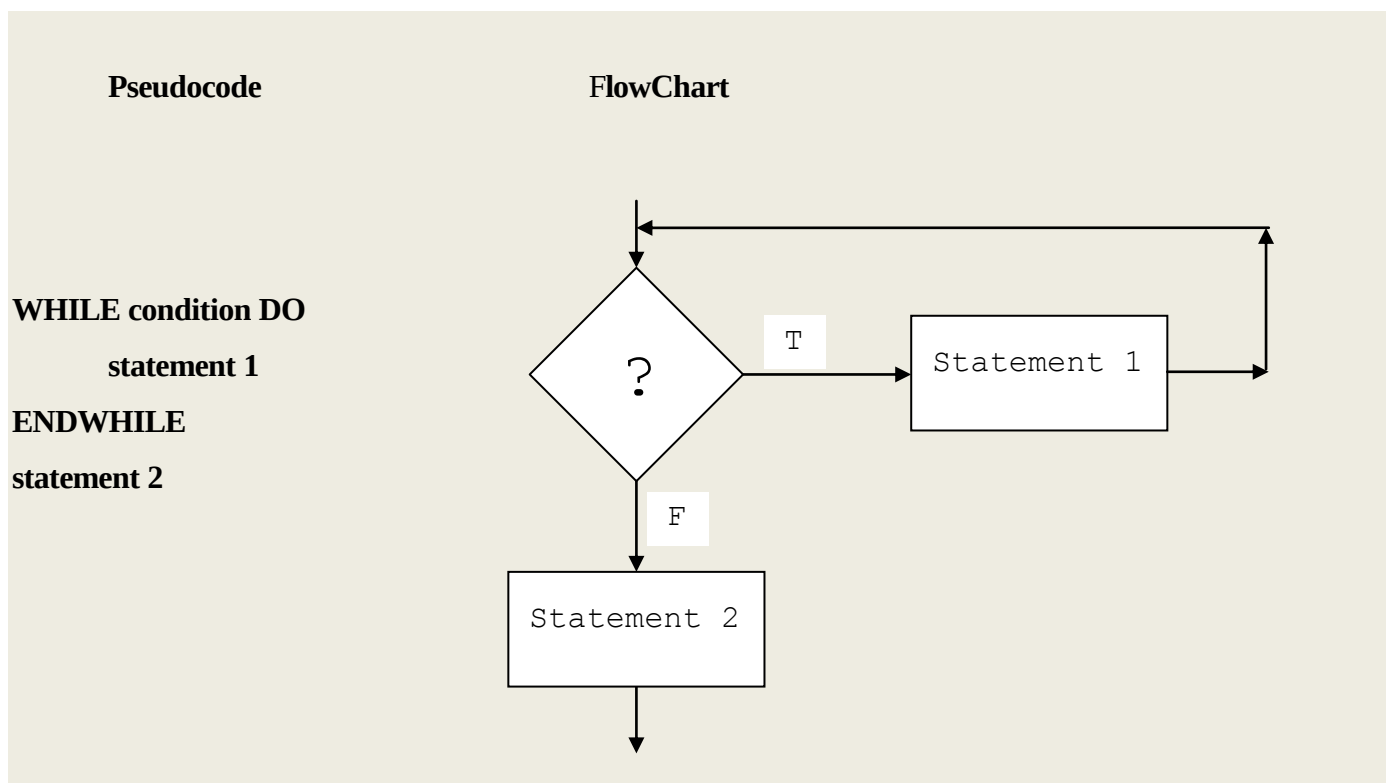
If this value matches A then statement1 would be executed.

Repetition

This construct allows a statement or block of statements to be repeated a number of times. Again, two forms are given :

- 1) WHILE loop - an entry condition loop, where the number of iterations may vary between 0 and n.
- 2) REPEAT loop - an exit condition loop, where the number of iterations may vary between 1 and n.

WHILE loop



The condition, a Boolean expression, is tested prior to entering the loop.

The statement inside the loop, statement 1 is executed as long as the condition is true.

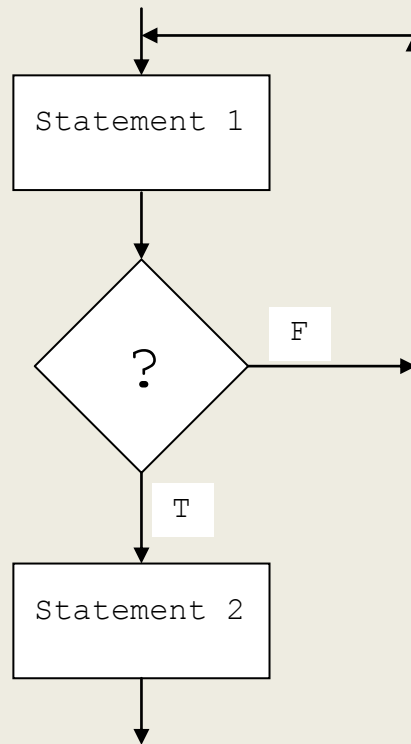
Statement 2 will only be executed after the loop is exited, i.e. the condition has become false.

REPEAT loop

Pseudocode

```
REPEAT  
    statement 1  
UNTIL condition  
statement 2
```

Flow Chart



In this form, the statement within the loop, statement 1, is executed before the condition is tested. Statement 1 will continue to be executed as long as the condition is false. Statement 2 will only be executed after the condition becomes true, and the loop is exited.

Boolean Expressions

A Boolean expression is one which returns a value of true or false. No other results are possible. They are used to control the flow of execution through the program, i.e. they determine which statement will be executed next.

The standard relational operators are used to produce Boolean expressions.

==	equal to
!=	not equal to
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to

For example,

```
IF x < 10 THEN
```

In the statement above, $x < 10$ is the Boolean expression, which will be true provided the value stored in the variable x is less than 10.

Consider an algorithm for a program which is going to convert a temperature either from Celsius to Fahrenheit or from Fahrenheit to Celsius. One solution to this problem could be

```
Display " Enter the temperature to be converted :"  
Read temp  
Display "Is this temperature in Celsius or Fahrenheit ?"  
Display "Enter C for Celsius, F for Fahrenheit"  
Read unit  
IF unit == C THEN  
    conv = temp * 9/5 + 32  
ELSE  
    conv = (temp - 32) * 5/9  
ENDIF  
Display "The converted temperature is " conv
```

In C++, the equivalent code (including variable declarations etc.) is

```
#include <iostream>
using namespace std;
void main(void)
{
    float temp, conv;
    char unit;
    cout<<"Enter the temperature to be converted : ";
    cin>>temp;
    cout<<"Is this temperature in Celsius or Fahrenheit ?\n";
    cout<<"Enter C for Celsius, F for Fahrenheit ";
    cin>>unit;
    if (unit == 'C')
        conv = temp * 9/5 + 32;
    else
        conv = (temp -32) *5/9;
    cout<< "The converted temperature is "<< conv;
}
```