FACTFILE:
GCE DIGITAL TECHNOLOGY

Rewarding Learning

AS1: APPROACHES TO SYSTEMS DEVELOPMENT: PROGRAMMING

Programming Structure 2: Sequence, Selection and Iteration

Learning Outcomes

Students should be able to:

- Describe the fundamental programming concepts of sequence, selection and iteration, including count-controlled and condition-controlled loops.

Sequence

Normally, the instructions contained in a program or solution are performed in the order in which they are listed. That is in sequence. This is the simplest control structure. The "Hello world" program and other examples seen in Programming Structure 1 are examples of programs implementing sequential control. The sequence of instructions can contain any number of actions and none can be omitted.

Selection

In many programs, the sequence of statements to be performed is determined by the input data. Decisions must be made, based on the values of certain variables, as to which sequence of statements is to be performed. Such decisions require the evaluation of a condition. The result of this evaluation determines which statements are to be executed next.

What is a condition?

A condition normally describes a particular relationship between a pair of variables or a variable and a constant.

Examples:

Conditions	Using mathematical symbols
GROSS greater than MIN	GROSS > MIN
X equal to Y	X = Y
X not equal to 0	X <> 0
COUNT less than or equal to 10	COUNT <= 10

Relational-Operator used in conditions	Meaning
=	Equal to
< > or !=	Not equal to
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to

The value of a condition is true if the specified relationship holds for the current variable values; otherwise the condition is false. True can be represented by a 1 and false by a 0.

Conditions can be combined using the following logical operators:

- AND
- OR
- NOT

Selection

IF THEN

A statement or sequence of statements is executed only if a condition is true.

Examples

```
IF amount_spent > 200 THEN
    discount = 10%
END IF

IF mark >= 40 THEN
    grade = 'PASS'
    increment pass_count
END IF
```

IF THEN ELSE

One statement or sequence of statements is executed if a condition is true. An alternative statement or sequence of statements is executed if the condition is false. The alternative can itself be an IF THEN or an IF THEN ELSE statement.

Examples

```
IF mark < 40 THEN
      grade = 'FAIL'
      increment fail_count
ELSE
      grade = 'PASS'
      increment pass_count
END IF

IF mark < 40 THEN
      grade = 'FAIL'
      increment fail_count
ELSE IF mark < 70 THEN
            grade = 'MERIT'
            increment merit_count
        ELSE
            grade = 'DISTINCTION
            increment distinction_count
END IF
```

Example
A program is required to calculate the discount that a customer will receive based on the amount spent as follows:

Amount Spent (£)	Discount
0–99	5%
100-200	7.5%
>200	10%

Algorithm

```
IF amount_Spent >=0 and amount_Spent<100 THEN
    Discount=5%
    ELSE IF amount_Spent >=100 and amount_Spent<=200 THEN
        Discount=7.5%
        ELSE IF amount_Spent >200 THEN
            Discount=10%

END-IF
```

c# code to represent algorithm above:

```
if (amount_Spent >=0) && (amount_Spent<100)
            Discount=5
            else
            If (amount_Spent >=100) && (amount_Spent<=200)
                        Discount=7.5
             else
                        If (amount_Spent >200)
                                    Discount=10
```

Iteration (Repetition) – Loops

A statement or sequence of statements can be executed more than once (or even not at all) through the use of a loop which can be **count-controlled** or **condition-controlled**.

Count-controlled loops

The statements within the loop are executed a number of times which is determined by a loop counter.

Example
Algorithm to calculate and output the average of 10 integers input by the user.

```
Begin
            Set sum to 0
            Repeat 10 times
                        Input number
                        Add number to sum
            End repeat
            average = sum / 10
            Output average
End
```

Different programming languages implement count-controlled loops in different ways. Usually, however, a count-controlled loop is implemented using a FOR statement which typically identifies the following:

- The variable which will control the loop.
- The initial value of the variable.
- The final value of the variable.
- The increment or decrement which is applied to the variable each time the loop is executed which usually defaults to 1.

A sample c# FOR Loop which will repeat 10 times.

```
for (i=0 ; i<=10 ; i++)
{
 Console.WriteLine( i*i);
}
```

A sample Python FOR loop which will repeat 10 times.

I Sample c# while loop incorporating if-statment

```
For counter in range (1, 10):
        print counter
```

Condition-controlled loops

Frequently the number of times the statements in a loop must be executed cannot be controlled by a counter.

A condition-controlled loop can be implemented in one of two ways:

Until

The loop is executed until a condition becomes TRUE with the condition being tested at the end of the loop.

While

The loop is executed while a condition is TRUE with the condition being tested at the start of the loop

Example
Algorithm to calculate how many times the integer 1 must be doubled before the result first exceeds one million.

```
Begin
        Set number to 1
        Set count to 0
        Repeat
                number = number * 2
                Increment count
        Until number > 1000000
        Output count
End
```

Example
Algorithm to calculate and output the average of a sequence of positive integers input by the user. The user indicates the end of the sequence by inputting the dummy value -1. It can be assumed the user inputs at least one positive integer.

Begin
    Set sum to 0
    Set count to 0
    Input number
    While number <> -1
        sum = sum + number
        increment count
        Input number
    End while
    average = sum / count
    Output average
End

Different programming languages implement condition-controlled loops in different ways.

```
int number=0;
        while (number<1 || number>20)
        {
            Console.WriteLine("Enter a number in the range 1 to 20");
            number = Convert.ToInt32(Console.ReadLine());
            if (number < 1 || number > 20)
                Console.WriteLine("Number out of range");

        }
```