Rewarding Learning

AS LEVEL

FACT FILES

Software Systems Development

AS1 Introduction to Object Oriented Development

Part iii – PROGRAM CONTROL STRUCTURE

For first teaching from September 2013

For first AS award in Summer 2014

For first A Level award in Summer 2015

software systems development

FACT FILE

software systems development

Introduction to Object Oriented Development

Part iii – PROGRAM CONTROL STRUCTURES

Learning Outcomes

Students should be able to:

- apply and evaluate the basic principles of control structures in terms of:
  - sequence (sequential functions, methods);
  - repetition (unconditional, conditional);
  - selection (decision IF, nested IF, switch);

Course Content

Sequence

Sequence refers to the order in which software instructions are carried out. In a sequential control structure, instructions are executed in the order in which they appear in the program.

Sequential Functions and Methods

A common sequence of instructions can be defined as a function or a method. A function or a method can be declared as void where a return value is not required. Where a return value is required, its type must be defined. Functions and Methods are executed in the order in which they are called.

Repetition

Computers have the capacity to perform repeated calculations at great speed. Software designers usually implement **repetition** or **iteration** through the use of loops. A loop allows a statement or a group of statements, to be executed more than once. A loop can be repeated either:

- a set number of times (for example 5 times);
- until a condition has been met (for example **until a is greater than int b**).

Unconditional Repetition

A For loop can be used where the number of repetitions of a loop is known. In this structure, the counter is initialised, the end value for the counter is declared and the increment for the counter set.

```
for (int i = 0; i < 5; i ++)
{
    Console . Writeline (i);
}
```

The loop will repeat the **WriteLine** instruction five times, each with a different value of integer.

Conditional Repetition

When it is not known how many times a loop should be repeated, a condition can be tested at the beginning or at the end of the loop. The loop will be repeated while the condition is met. For example, to divide 2030 by 7, a computer can repeatedly subtract 7 from 2030, each time incrementing a counter. When the result of the subtraction is 0 or less, then the counter will hold the (whole number) answer to 2030/7. This can be coded using a **while** instruction rather than a **for** instruction.

Selection

Within a program, it is possible to decide which statements to execute using a **selection**. This is normally achieved using an **if** or a **switch** statement.

If Statements

An if statement allows control to branch to the execution of a set of instructions only if the condition evaluates to TRUE. A condition is an expression, or a combination of expressions joined by the logical operators && and/or ||, which evaluates to TRUE or FALSE.

```
if (lives_left = = 0)
{
    Save_current_score ( );
    End _game( );
}
```

In the example above, if the current value of lives_left is 0, a function is called to save the players score, followed by a function to end the game. If lives_left is not 0, control branches to the end of the **if structure.**

if-else Statements

An if-else statement will also include statements to be executed if the evaluation of the condition is FALSE.

```
lives_left  - =  1;   // short for lives_left = lives_left -1
if (lives_left = = 0)
{
    Save_current_score ( );
    End _game( );
}
else
{
     Decrease_score  ( );
}
```

When the decision is evaluated, if the current value of lives_left is 0, a function is called to save the players score, followed by a function to end the game. Otherwise the control will branch to the else section and the function will be called to decrease the players score.

Nested if

Use of nested if statements is common practice for the solution of more complex problems.

```
if ( category  = = 'A')
{
   If (result  >= 85)
      grade = " Distinction";
   else
       if (result >= 60)
           grade ="Merit";
        else
            if (result >= 40)
                grade = "Pass";
            else
                grade ="Fail";
}
else
{
    if (result >= 40)
        grade = "Pass";
    else
        grade ="Not Achieved";
}
```

Switch Statements

Another type of selection construct is known as the **switch** statement. Rather than use a simple TRUE/FALSE condition to make a selection, the **switch** construct allows multiple decisions to be effected. Switch statements, if relevant, result in more efficient code than multiple **if** statements.

Control is transferred to the **case** statement which matches the value of the switch variable. The switch statement can include any number of unique case statements. Execution proceeds until the **break** statement transfers control out of the switch structure. A branch statement such as a **break** is normally required after each case block. C# does not support an implicit fall through from one case label to another. The one exception is if the case statement has no code. If no case expression matches the switch value, then control is either, transferred to the statement block that follows the default label or exits if no default is included.

```
a = 20;
switch (a)
{
  case 10 : Console.WriteLine ("\n\t\tswitch on value 10");  break;
  case 20 :
  case 40 : Console.WriteLine ("\n\t\tswitch on value 20 or value 40");  break;
}
```

Note that the output from the code above is switch on value 20 or value 40 as it fell through case 20 because it contained no code. If 'a' had the value 50 assigned, then there would be no output as a default statement is not included. Note that to preserve the logic of multiple statements for decisions and repetition, they must be enclosed in { } parenthesis'

Activities/Questions

Q1 Insert the correct term from the options below in the blank spaces of each statement.

sequence **statement** **function/method**
iteration **selection**

- A _____ is a control structure in which a set of instructions is executed in order.
- A _____ is a single instruction within a program.
- _____ occurs when a control structure has more than one potential path and this path is determined using choices based on conditions in the program.
- A _____ is a set of instructions that performs a specific process for a program as often as necessary.
- _____ is a control structure in which a group of instructions is executed more than once.

Q2 The body of a while loop can consist of:

a. a single instruction:
b. a block of instructions within curly braces;
c. comments only;
d. either **a** or **b**;
e. neither a nor b.

Q3 What is the output of the following code segment?

```
int a = 3
int b = 4;
if ( a = = b)
   Console.Write ("Black ");
   Console.WriteLine ("White");
```

software systems development

COUNCIL FOR THE **CURRICULUM EXAMINATIONS** AND **ASSESSMENT**
29 Clarendon Road, Clarendon Dock, Belfast BT1 3BG
✆ +44 (0)28 9026 1200 🖷 +44 (0)28 9026 1234 ☎ +44 (0)28 9024 2063 ✉ info@ccea.org.uk 🌐 www.ccea.org.uk
Rewarding Learning