AS1 LEVEL

FACT FILES

Software Systems Development

For first teaching from September 2013

For first AS award in Summer 2014

For first A Level award in Summer 2015

Introduction to Object Oriented Development Part (v) – Data Structures

FACT FILE

software systems development

FACT FILE

software systems development

Part (v) – Data Structures

Learning Outcomes

Students should be able to:

- explain the need to store and organise data efficiently within specific structures;
- explain, design and use the following data structures:
  - strings; and
  - static arrays (of simple type and of objects).

Content

Data Structures – String

When working with data, it is often useful to be able to group more than one individual unit of data together. By doing so programs can refer to a set of data objects using just one identifier (or name). An example of this is the **string** data structure. Using a string we can store a long list of **char** data types with just one identifier.

```
// Declaring a name using Char data types

char char_name_1 = 'J';

char char_name_2 = 'e';

char char_name_3 = 'f';

char char_name_4 = 'f';

// Declaring a name using a string Data Structure

string str_name = "Jeff";
```

In C#, a string can be defined as type **string** or **String**.
The type **String** adheres to the Object Oriented concept of a class name beginning with a capital letter. The functionality of the class is available through methods. Methods are available to split strings apart, shorten strings or change case.

For example. **ToUpper( )** is a method that changes a string to uppercase.

An individual character can be extracted from a string. In many languages strings which are numbers stored as text, such as "123", can be implicitly converted to corresponding integers. C# must explicitly convert the string as shown:

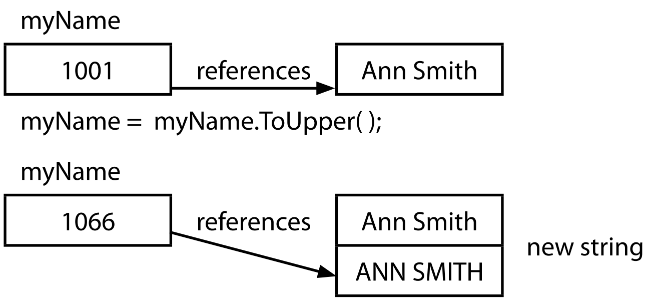
```
int no;
string num = "123";
no = Int16.Parse(num);
```

The type **string** has been maintained for legacy reasons but is an alias for the type **String**. It behaves in a similar fashion and will access all methods of the class String.

Strings are so common that the overloaded operator **+** can be applied to them and is generally used rather than the method Concat( ).

```
{
    string saying;
    string part1 = " A stitch in time";
    string part2 = " saves nine";

    Console.WriteLine(part1 + part2);
    Console.WriteLine("The length of the first string is " + part1.Length.ToString() + "  characters");
    saying = part1+part2;
   // saying = String.Concat(part1,part2);
}
```

It is important to remember that any change to a string results in a new string being formed. An example of this is myName = myName.ToUpper( ).



Note the address in myName has changed to reference the new string.

Note that the other primitive types in C# can access the methods provided by the base class Object.

Arrays

An array is an organised arrangement of data in rows and columns. The most basic data array is an organised arrangement of primitive data types such as int or char.

```
// Declare a single-dimensional array
   int [ ] nos;   // Declares the intention to use an array of  type int called nos
   int [ ] numbers = new int [2];   // Creates an array that can hold two integers  not yet assigned
   int [ ] hoursOfSunshine = {6,5,9,5,4,1,2};   // Creates and array called hoursOfSunshine that is
                                                //   populated with seven numbers indexed from 0 to 6
                                                // hoursOfSunshine [3] refers to the number 5
// Alternatively for hoursOfSunshine
    int [ ] hoursOfSunshine =  new int [7]  {6,5,9,5,4,1,2};
```

Each element in an array is referenced, indexed and updated individually. It is very straightforward to loop through the elements in an array using the **for** and **foreach** commands.

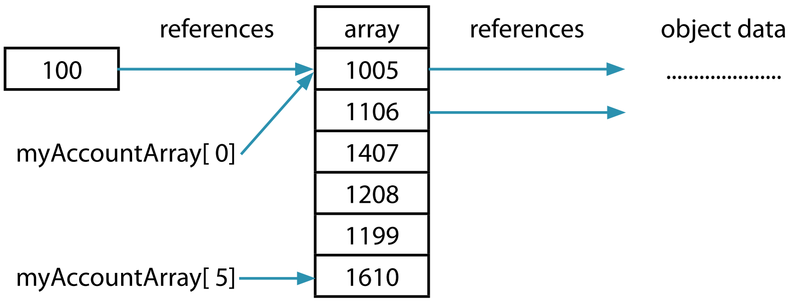
```
int [ ] hoursOfSunshine = {6,5,9,5,4,1,2};

for ( int x =0;  x<hoursOfSunshine.Length; x++)

{

   Console.WriteLine(hoursOfSunshine[x]);

}
```

```
int [ ] hoursOfSunshine = {6,5,9,5,4,1,2};

foreach ( int itm in hoursOfSunshine)

{

   Console.WriteLine(itm);

}
```

Arrays of Objects
Arrays are fixed length groups of type related data. The data could be objects of type String or objects instantiated from a user-defined class such as Account.

An array for objects holds the addresses of where the objects are stored in memory.
Consider a simple class Account with three fields :–
int accountNo,   string accountType,   decimal amount

An array, myAccountArray could be declared to hold details of customers' accounts.
myAccountArray



The object array is declared in the same format:-

```
Account [ ]  myAccountArray;   //or
Account [ ]  myAccountArray  =  new  Account [ 5];
```

The object array must have each object instantiated.

```
myAccountArray [0]  =  new   Account( 112000, "Cash Isa",  5000.00);
myAccountArray [1]  =  new   Account( 112001, "Current",  120.00);
```

Note that strings do not require **explicit instantiation.** This is automatically taken care of.

Two Dimensional Arrays

If a one-dimensional array can be thought of as a column, then a two-dimensional array could be thought of as a grid or table.

```
 // Declare a two-dimensional array for Sudoko Game

int [,] int_array_game_board = new int[3,3];

int_array_game_board[0,0] = 3;
int_array_game_board[0,1] = 2;
int_array_game_board[0,2] = 1;

int_array_game_board[1,0] = 5;
int_array_game_board[1,1] = 8;
int_array_game_board[1,2] = 9;

int_array_game_board[2,0] = 7;
int_array_game_board[2,1] = 0;
int_array_game_board[2,2] = 6;
```

Larger arrays can be defined for more complex problems.

Questions

Q1 Which of these statements correctly declares a two-dimensional integer array in C#?

a. int[,] myArray;
b. int[][] myArray;
c. array int [][];
d. int[2] myArray;
e. object [2] [2] int myArray;

Q2 Which option accurately reflects the output of the program?

```
String str_a = "12345";
String str_b = "54321";
String str_c = "13524";
String str_d;
str_d = str_a[1] + str_b[3] + str_c[2];
Console.WriteLine(str_d);
```

a. 1 3 3
b. 133
c. 7
d. 3 3 5
e. 335
f. 11
g. 5 1 4
h. 514
i. 10
j. 2 2 5
k. 225
l. 9
m. 1 3 5
n. 134
o. 8

Q3 What is the output from the program listing given?

```
int[ ]  x = new int [9];

x[0] = 7;
x[1] = 3;
x[2] = 4;

Console.WriteLine ( (x[0] + x[1]) + " " + x[5] );
```

a. 730
b. 73 0
c. 100
d. 10 0

Q4 What does a Two-Dimensional Array do?

a. It creates an array which can be seen by two programs.
b. It forms a table-like array with more than one column of data.
c. It creates two arrays at once.
d. It creates an array with only two data elements.