

CO453 Application Programming

Week 2 - C# Part 5

Arrays and sorting algorithms

What is an array?

Data Structures

We can combine simple data types into more complex structures

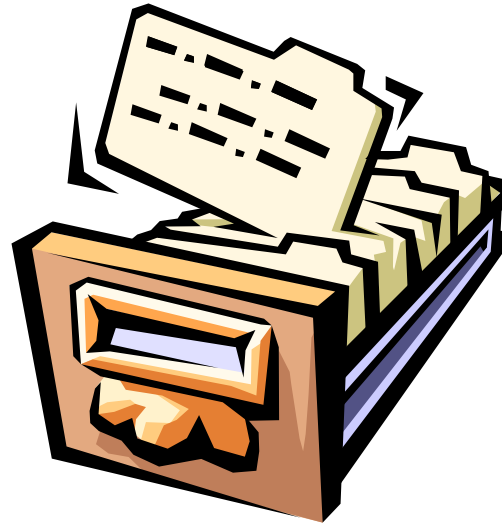
Array

a numbered list of similar data types



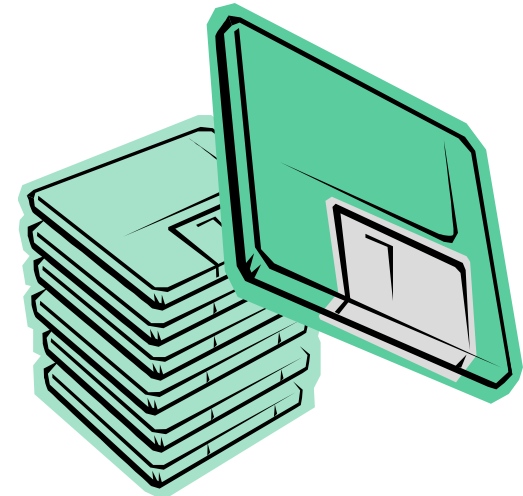
Class

a single package to hold data and functions (methods) for an object



File

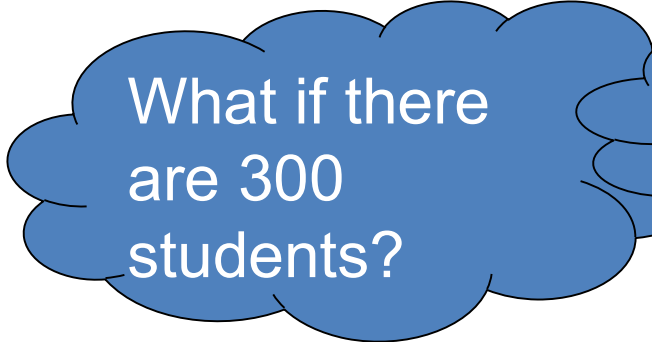
long-term storage for data



Problem: storing lots of data

Imagine that we want to store the ages of a class of 30 students

```
int age1 ;  
int age2 ;  
int age3 ;  
int age4 ;  
etc.
```



What if there
are 300
students?



There must be
a better way!

```
age1 = dialog ( "Enter age of student 1" );  
age2 = dialog ( "Enter age of student 2" );  
age3 = dialog ( "Enter age of student 3" );  
  
etc.
```

Arrays as a solution

Arrays allow us to store lots of data as a collection of **elements**

```
int age1 = 23;  
int age2 = 32;  
int age3 = 43;  
int age4 = 54;  
etc.
```

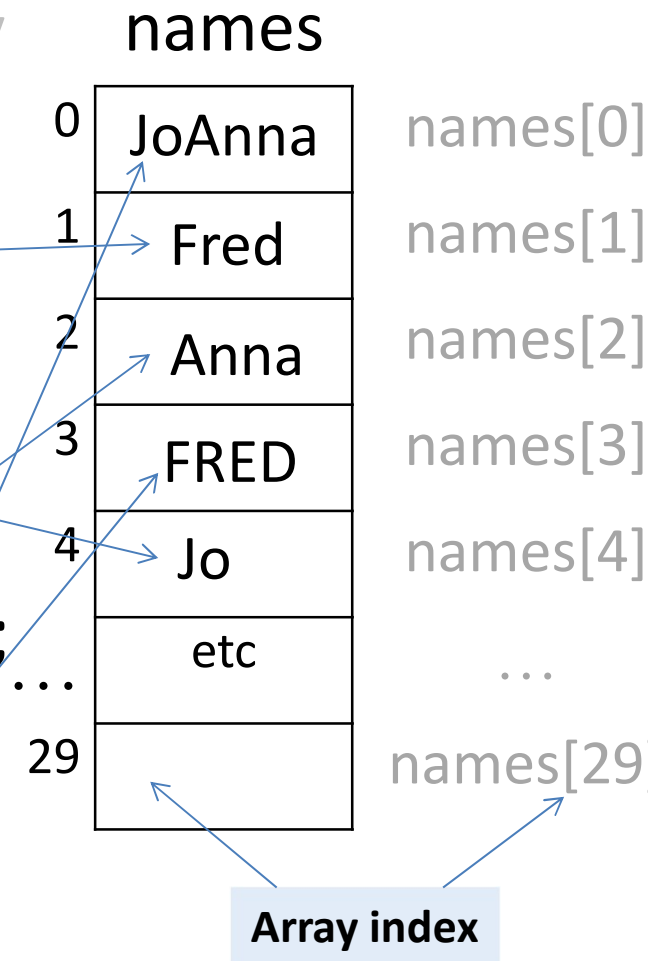
age	
0	23
1	32
2	43
3	54
4	etc

Each element acts like a variable (storage space) but is referenced as being part of an array (*age in this case*)

Array method

```
string[] names; //define array  
names = new string[30]; //create array
```

```
names[1] = "Fred";  
names[4] = "Jo";  
names[2] = Console.ReadLine();  
names[0] = names[4] + names[2];  
names[3] = names[1].ToUpper();
```



Defining an array of 30 integers would be written: `int age[30];`

Question: are the elements numbered...

1 – 30 ?

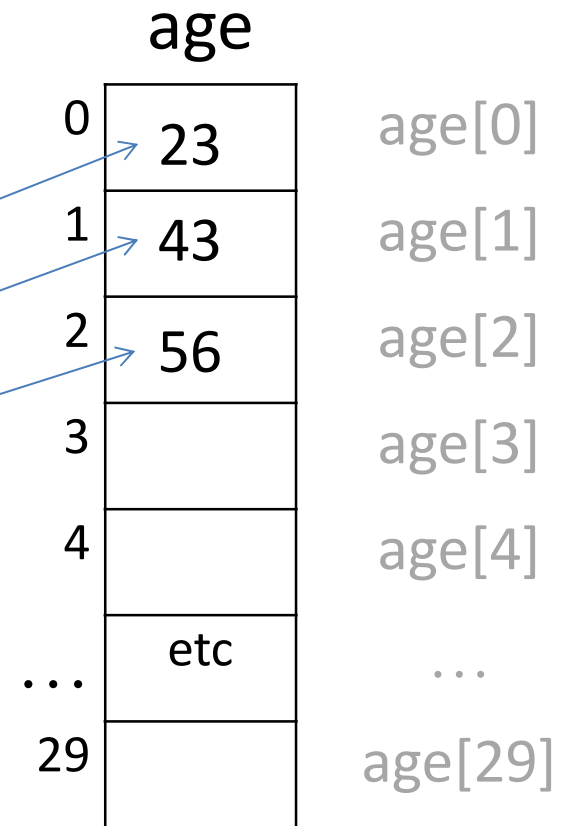
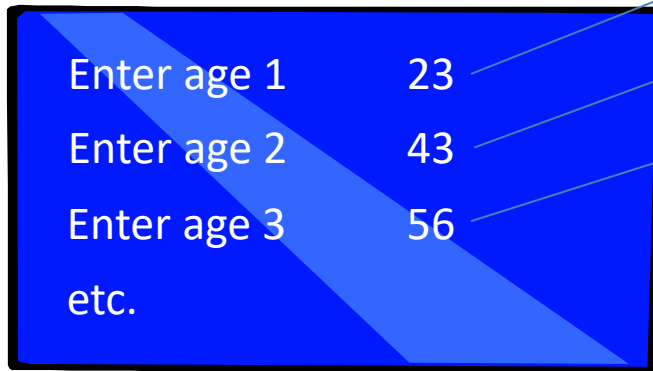
or

0 – 29

Input using a loop

A for loop can be used to input the contents of the whole array

```
for (int i = 0; i < 30; i++) {  
    Console.WriteLine("Enter age " + (i+1));  
    age[i] = Convert.ToInt32(Console.ReadLine());  
}
```

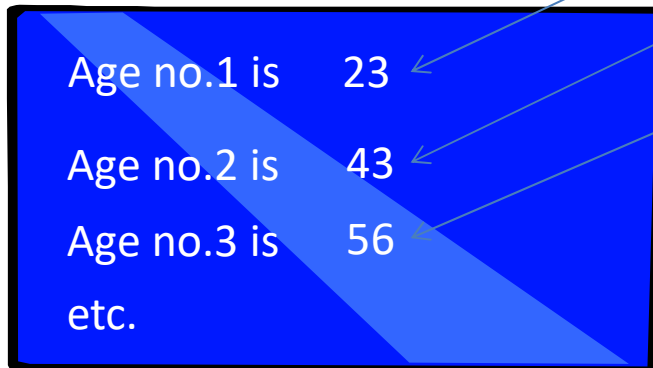


Why use a for loop?

Output using a loop

A for loop can be used to print the contents of the whole array

```
for (int i = 0; i < 30; i++) {  
    Console.WriteLine("Age no. " + (i+1)  
        + " is " + age[i]);  
}
```



```
Age no.1 is 23  
Age no.2 is 43  
Age no.3 is 56  
etc.
```

age		
0	23	age[0]
1	43	age[1]
2	56	age[2]
3		age[3]
4		age[4]
...	etc	...
29		age[29]

Other Types of Array

We can produce arrays using any of the usual data types: e.g.

```
int age[30];           // defines an array of 30 integers
```

```
float wage[20];      // an array of 20 float numbers
```

```
object items[100];   // an array of 100 objects
```

```
string names[20];    // an array of 20 strings
```

Each element of an array can be accessed using the array index (the integer variable *i* in the previous slides)



```
age [ i ]  
wage [ i ]  
items [ i ]  
names [ i ]
```

Initialising Arrays

Arrays can be initialised as they are declared e.g.

```
int arr1[5] = { 1, 2, 6, 0, 4 } ;
```

initialises arr1
with 5 values

```
double arr2[10] = { 3.2, 4.1, 6.7 } ;
```

initialises arr2
with 3 values
(the rest are 0)

```
int arr3[100] = { 0 } ;
```

initialises all arr3
elements to 0

```
int arr4[ ] = { 1, 2, 6, 0, 4 } ;
```

Array arr4 is
created of correct
size

Games Scores

```
class Game  
{
```

```
    int[] scores;           // define an array for scores  
    const int MAX = 6;     // set a maximum size
```

```
public static void Main()  
{  
    Game myGame = new Game();  
    myGame.getScores();  
    myGame.showScores();  
}
```

**Main()
method**

```
public Game()  
{  
    scores = new int[ MAX ];  
}
```

**Game class
constructor**

**Here an empty scores array
of size MAX is created**

```
public void getScores ()
{
    Console.WriteLine("Game Scores Entry");
    Console.WriteLine("=====");
    for (int i=0; i < MAX ; i++)
    {
        Console.Write("Enter Score " + (i+1));
        scores[i] = Convert.ToInt32(Console.ReadLine());
    }
}
```

All MAX scores are here
entered in the scores array

```
public void showScores ()
{
    Console.WriteLine("Game Scores");
    Console.WriteLine("=====");
    for (int i=0; i < MAX ; i++)
    {
        Console.WriteLine("Score " + (i+1)
            + " is " + scores[i]);
    }
}
```

Two-Dimensional Arrays (tables)

Arrays with 2 Dimensions

2-dimensional arrays look like tables with rows and columns

Define the Array

```
int[,] marks = new int [3,5];
```

Input to the array

```
marks [0, 1] = 24;
```

```
marks [1, 4] = 41;
```

```
marks [2, 3] = Console.ReadLine();
```

```
marks [1, 0] = marks [0, 1] + marks [2, 3];
```

marks array

	0	1	2	3	4
0		24			
1	43				41
2				19	

Output from the array

```
message( "The mark for student 5 of class 2 is " + marks [1][4] );
```

Filling the 2-D marks Array

To fill the whole marks array, we need to use 2 for-loops

```
for ( i = 0; i < 3; i++)           // 3 classes
  for ( j = 0; j < 5; j++)         // 5 students per class
  {
    Console.WriteLine("Enter mark for class " + i + " student
    " + j );
    marks[i , j] = Convert.ToInt32(Console.ReadLine());
  }
```

marks array

	0	1	2	3	4
0	64	78	56	77	83
1	46				
2					

Enter mark for class 0 student 0 64
Enter mark for class 0 student 1 78
Enter mark for class 0 student 2 56
Enter mark for class 0 student 3 77
Enter mark for class 0 student 4 83
Enter mark for class 1 student 0 46
etc.

the code uses
nested
for-loops

Sorting Algorithms

- Sorting algorithms order items in an array so certain values can be found quicker
 - An ordered list is much easier to search than an unordered list
- There are many sorting algorithms
 - There isn't one algorithm that is better than the rest
 - It depends on the size of the array and how ordered the list is
 - Some are designed to sort large amounts of data; where as others are only a few lines of code

Bubble Sort Algorithm

loop N times

loop from 0 up to N-1

 if current item > next item

swap the two items

 end if

end loop

end loop

Bubble sort method

```
public void BubbleSort(int[] A)
{
    int temp;
    for (int i = 0; i < A.Length; i++) //loop N times (size of the array)
    {
        for (int j = 0; j < A.Length - 1; j++) // loop from 1 to N-1
        {
            if (A[j] > A[j + 1]) // swap values
            {
                temp = A[j];
                A[j] = A[j + 1];
                A[j + 1] = temp;
            }
        }
    }
}
```

Example program

```
public static void Main()
{
    int[] A = new int[30];        //declare array
    Random r = new Random();
    for (int i=0; i < A.Length; i++) //populate
        A[i] = r.next(100);

    BubbleSort(A); // see previous page
    Display(A);
}
```

```
public void Display()
{
    for (int i=0; i < A.Length; i++)
        Console.WriteLine("Value "+i+" is "+ A[i]);
}
```

The Last Slide



Extra Reading



Passing arrays

To pass an array to a function, **we only need to pass its name**. Any changes made inside the function automatically change the original array

Recap

- The items in an array are called **elements**
- We specify how many elements an array will have when we declare the **size** of the array (if 'fixed-size')
 - Dynamic arrays don't need a size on declaration
- Elements are numbered and can be referred to by number inside the [] is called the **index**
 - This is used when data is input and output
- Can only store data if it **matches the type** the array is declared with

The Constructor

- The constructor is a special method in a class
- It always has the same name as the class
- When an object is created from a class, the constructor is automatically executed
- It is used to initialise the new object

```
public Dice()  
{  
    rand = new Random();  
}
```

This constructor creates a new Random object, used to generate random numbers for the Dice

How can we return more than 1 result from a method or function?

- We can use parameters to make changes to the original variables.
- To do this we can use reference parameters instead of value parameters.
- Reference parameters are defined using ref
e.g. `public void times (ref double n1, ref double n2)`
defines `n1` and `n2` as reference parameters
- Now any change to `n1` or `n2` inside the method will also change the value of the parameter passed to it.
- This is because they are essentially the same variable .. using the same memory address
- Note you must also use ref when you call the method
e.g. `times (ref number1, ref number2);`