**Control statements:**

1. Conditional Statements:
2. if
3. if else
4. nested if
5. elsif
6. switch
7. Loops:
8. While
9. Do while
10. For
11. Foreach
12. Branching Statements/ Unconditional Statements:
13. Break
14. Continue
15. Return

Ex:- foreach

Prog:

Namespace ConsoleApplication1

Class Program

{

Static void Main (String args[])

{

List<String> names= new List<String>;

Names.add(“chitti”);

Names.add(“pavan”);

Names.add(“sweety”);

Names.add(“cherry”);

Foreach( String item **in** names)

{

Console.writeLine(item);

}

Console.ReadLine( );

}

}

**Array:** Collection of homogenise elements is called array. Array is stored same data type elements. An array is a start with in index is ‘0’.

Types of arrays:

1. Single Dimensional array (1’D)
2. Two Dimensional array (2’D)
3. Jagged array :- Not fixed size of array

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | | | |
|  |  |  |  |  | | |
|  |  |  |  |  |  | |
|  |  |  |  |  |  |  |

**Jagged array**

**OOP’s:**

1. Class
2. Object
3. Abstraction
4. Encapsulation
5. Inheritance
6. Polymorphism

**Class:**  Group of member functions &Data members.

Or

Collection of variables and methods is called class.

Ex:-

Namespace ConsoleApplication1

Class Program

{

Int num;

String name;

Void Percentage()

{

}

Static void Main (String [] args)

{

}

}

**Object:** Object is a real time entity ,it means real world existing things. Object is a instance of a class.

Syntax:- ClassName Object = new ClassName;

Example: Program Obj=new Program();

**Abstraction/Data hiding:-** Hiding unnecessary information, providing necessary information.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Access Specifies/Modifiers** | **Within class** | **Derived class with in Namespace** | **With in Namespace** | **Derived class other Namespace** | **Anywhere** |
| **Private** | Y | N | N | N | N |
| **Protected** | Y | Y | N | N | N |
| **Internal** | Y | Y | Y | N | N |
| **Protected Internal** | Y | Y | Y | Y | N |
| **Public** | Y | Y | Y | Y | Y |

**Encapsulation:-** Wrapping data into single unit.

EX:- Class , struct….,

Struct StructureName

{

Int num;

String Name;

Int rank;

Public void show( );

{

Console.Write();

}

}

|  |  |
| --- | --- |
| Value | Reference |
| Int x=10; | Int x=y; y=10; |
| Store Stack Memory | Heap Memory |
| x->10 | x->y->10 |
| Ex:- Structure | Ex:- Class |

**Inheritance:-**

One class is derived to another class

Parent class-Base class-Super class

Child class-derived class-sub class

1. Single level Inheritance
2. Multi level
3. Multiple\*\*\*not supported in C#...
4. Hybrid
5. Hyrarical

**Single level inheritance:- A**  base class is derived to sub class is called Single level inheritance.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApplication1

{

class Addition

{

public static void Add(int a, int b)

{

Console.WriteLine("Addition of the numbers is:" + (a + b));

}

}

class Subtraction:Addition

{

public static void Sub(int a, int b)

{

Console.WriteLine("Subtraction of the numbers is:" + (a - b));

}

static void Main()

{

Add(10, 20);

Sub(10, 20);

Console.ReadKey();

}

}

}

**Multiple inheritance: - by**  default multiple inheritance doesn’t support bcz of it face confusion .

We can achieve with the help of interface.

**Interface:** Interface we have a method declaration but not implementation by default all interface members are public .

* All interface members must be implemented in derived class.
* We can’t create Object for interface.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApplication1

{

interface IAddition

{

void Add(int a, int b)

}

interface IMultification

{

void Mul(int a, int b)

}

class Calculation: IAddition, IMultification

{

public static void Sub(int a, int b)

{

Console.WriteLine("Subtraction of the numbers is:" + (a - b));

}

public void Add(int a, int b)

{

Console.WriteLine("Addition of the numbers is:" + (a + b));

}

public void Mul(int a, int b)

{

Console.WriteLine("Multiplication of the numbers is:" + (a \* b));

}

static void Main()

{

IAddition Iadd = new Calculation();

Iadd.Add(10, 20);

IMultification Imul = new Calculation();

Imul.Mul(10, 20);

Sub(10, 20);

Console.ReadKey();

}

}

}