Connection

static void Main(string[] args)

{

SqlConnection sqlConnection;

string Connectionstring = @"Data Source=DESKTOP-B4EEIPP\SQLEXPRESS;Initial Catalog=user;Integrated Security=True";

try

{

sqlConnection = new SqlConnection(Connectionstring);

sqlConnection.Open();

Console.WriteLine("success");

Console.WriteLine("enter your name");

string username = Console.ReadLine();

Console.WriteLine("enter your age");

int age = int.Parse(Console.ReadLine());

string insertQuery = @"insert into info values('"+username+"','"+age+"')";

SqlCommand insertCommand = new SqlCommand(insertQuery, sqlConnection);

insertCommand.ExecuteNonQuery();

sqlConnection.Close();

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

}

!st Programm

Console.WriteLine("Hello World!");

Console.ReadLine(); //return a string

Reading and writing

Console.WriteLine("Enter Your Name !");

string name = Console.ReadLine();

Console.WriteLine("your name is " + name); //concatination

Console.ReadLine();

Other way

Console.WriteLine("Enter Your Name !");

string name = Console.ReadLine();

string age = Console.ReadLine();

//Console.WriteLine("your name is " + name + "" + age); //concatenation syntex

Console.WriteLine("your name is {0} and age{1}",name,age); //plaseholder syntex

Console.ReadLine();

Sum of two numbers

static void Main(string[] args)

{

Console.WriteLine("enter your number ");

int num1 = int.Parse(Console.ReadLine()); //convert into int by int.parse

Console.WriteLine("enter your secound number ");

int num2 = int.Parse(Console.ReadLine());

int sum = num1 + num2;

Console.WriteLine("your addition is {0}", sum);

Console.ReadLine();

}

BUILT IN DATA TYPES

Integral type

Signed integers (which take negative and positive values)

Sbyte -128 to 127 signed 8bit integer

Short -32.768 to 3w.767 16 bit integer

Int 32bit integer

Long 64 bit integer

Unsigned integer(which takes only positive values)

Byte 0-255 unsigned 8bit integer

Ushort unsigned 16 bit integer

Uint unsigned 32 bit integer

Ulong unsingned 64 bit integer

And

Char Unicode 16 bit character

Console.WriteLine(int.MinValue);

Console.WriteLine(int.MaxValue);

Console.WriteLine(uint.MaxValue);

Console.WriteLine(uint.MinValue);

Console.WriteLine(float.MinValue);

Console.WriteLine(float.MaxValue);

Console.WriteLine(byte.MaxValue);

Console.WriteLine(byte.MinValue);

Console.WriteLine(sbyte.MaxValue);

Console.WriteLine(sbyte.MinValue);

Console.WriteLine(short.MaxValue);

Console.WriteLine(short.MinValue);

Console.WriteLine(ushort.MaxValue);

Console.WriteLine(ushort.MinValue);

Console.WriteLine(long.MaxValue);

Console.WriteLine(long.MinValue);

Console.WriteLine(ulong.MaxValue);

Console.WriteLine(ulong.MinValue);

Console.ReadLine();

-2147483648

2147483647

4294967295

0

-3.4028235E+38

3.4028235E+38

255

0

127

-128

32767

-32768

65535

0

9223372036854775807

-9223372036854775808

18446744073709551615

0

BUILT IN DATA TYPES

Boolean data type

Boolean data type which only store true or false

int a = 30;

int b = 40;

bool c = a > b;

Console.WriteLine(c);

False

int a = 30;

int b = 40;

bool c = a < b;

Console.WriteLine(c);

True

Float double and decimal data type

C# .net type size precision

Float system.single 4byte 7digit

Double system.double 8byte 15-16 digit

Decimal system.decimal 16 byte 28-29 digit

float a = 23.23456f;

Console.WriteLine(a);

23.23456

Or

float a = 23.2345634354f;

Console.WriteLine(a);

23.234564

double a = 23.2344d;

Console.WriteLine(a);

23.2344

Or

double a = 23.23444444565655753d;

Console.WriteLine(a);

23.234444445656557

decimal a = 23.23444444565655753123456789m;

Console.WriteLine(a);

23.23444444565655753123456789

Or

decimal a = 23.2344444456565575312345678912m;

Console.WriteLine(a);

23.234444445656557531234567891

String and character data type

String stores multiple character in a single variable

Double quotes will be used with string data type

Char store single character at a time in a variable

Single quotes will be used for char data type

string a = "welcome to in";

Console.WriteLine(a);

welcome to in

escape sequence

string a = "\"welcome to in\"";

Console.WriteLine(a);

"welcome to in"

string a = "\\welcome to in\\";

Console.WriteLine(a);

\welcome to in\

string a = "one \n two \n three";

Console.WriteLine(a);

one

two

three

verbatim literal

string a = "D:\\beast\\tutorials\\csharp";

Console.WriteLine(a);

D:\beast\tutorials\csharp

string a = @"D:\beast\tutorials\csharp";

Console.WriteLine(a);

D:\beast\tutorials\csharp’

Verbatim literal make escape sequence translate as normal

Data type conversion

There is two types of conversion implicit and explicit conversion

Implicit conversion done by the compiler

int a = 20;

float b = a; //implicit conversion

Console.WriteLine(b);

20

float a = 20.4566f;

int b = (int)a; //explicit conversion

Console.WriteLine(b);

20

Or

float a = 20.4566f;

int b = Convert.ToInt32(a); //explicit conversion

Console.WriteLine(b);

But we loss the data after from .

string a = "20";

string b = "30";

int c = Convert.ToInt32(a) + Convert.ToInt32(b); //explicit conversion

Console.WriteLine(c);

50

Or

string a = "20";

string b = "30";

int c = int.parse(a) + int.parse(b); //explicit conversion

Console.WriteLine(c);

50

Parse class is only take a string

string a = "20.34";

string b = "30.45";

float c = float.parse(a) + float.parse(b); //explicit conversion

Console.WriteLine(c);

50.79

Console.WriteLine("enter you 1st number");

float a = float.Parse(Console.ReadLine());

Console.WriteLine("enter you 2nd number");

float b = float.Parse(Console.ReadLine());

float c = a + b;

Console.WriteLine(c);

enter you 1st number

5.454

enter you 2nd number

5.6

11.054

CONSTANT

Constant mean we cannot change the constant variable

Constant variable can be declare in class and method both

class Program

{

public const double a = 3.14;

static void Main(string[] args)

{

Console.WriteLine(a);

}

}

3.14

int a; //declaration

a = 10; // initialization

int a = 10; // initialization

const variable initialization is mendetory you cannot declare the const variable

class Program

{

static void Main(string[] args)

{

const double a = 3.14;

Console.WriteLine(a);

}

}

3.14

Date time format specifiers

Formate specifier

D long date

d short date

f full date time short time

F full date time long time

g general date/time short time

DateTime dt = DateTime.Now;

Console.WriteLine("{0}", dt);

14-Mar-22 2:34:42 PM

DateTime dt = DateTime.Now;

Console.WriteLine("{0:d}", dt);

14-Mar-22

DateTime dt = DateTime.Now;

Console.WriteLine("{0:D}", dt);

Monday, March 14, 2022

DateTime dt = DateTime.Now;

Console.WriteLine("{0:f}", dt);

Monday, March 14, 2022 2:36 PM

DateTime dt = DateTime.Now;

Console.WriteLine("{0:F}", dt);

Monday, March 14, 2022 2:39:02 PM

DateTime dt = DateTime.Now;

Console.WriteLine("{0:g}", dt);

14-Mar-22 2:41 PM

DateTime dt = DateTime.Now;

Console.WriteLine("{0:D}", dt);

Console.WriteLine("{0:f}", dt);

Console.WriteLine("{0:F}", dt);

Console.WriteLine("{0:g}", dt);

Console.WriteLine("{0:G}", dt);

Console.WriteLine("{0:m}", dt);

Console.WriteLine("{0:M}", dt);

Console.WriteLine("{0:T}", dt);

Console.WriteLine("{0:t}", dt);

Console.WriteLine("{0:d}", dt);

Console.WriteLine("{0:Y}", dt);

Console.WriteLine("{0:y}", dt);

Console.WriteLine("{0:ddd}", dt);

Console.WriteLine("{0:dddd}", dt);

Console.WriteLine("{0:FF}", dt);

Console.WriteLine("{0:HH}", dt);

Console.WriteLine("{0:MM}", dt);

Console.WriteLine("{0:MMM}", dt);

Console.WriteLine("{0:ss}", dt);

Console.WriteLine("{0:HH:mm:ss tt}", dt);

Console.WriteLine("{0:dd-MM-yyy}", dt);

Monday, March 14, 2022

Monday, March 14, 2022 2:58 PM

Monday, March 14, 2022 2:58:38 PM

14-Mar-22 2:58 PM

14-Mar-22 2:58:38 PM

March 14

March 14

2:58:38 PM

2:58 PM

14-Mar-22

March 2022

March 2022

Mon

Monday

03

14

03

Mar

38

14:58:38 PM

14-03-2022

Statement expression and operators

Operand and operator called expression

Arithmetic operators are binary operators because arithmetic operator is work with two operends

int a = 10, b = 5, c, d, e, f, g;

c = a + b;

d = a - b;

e = a / b;

f = a \* b;

g = a % b;

Console.WriteLine("{0} \n {1} \n {2} \n{3} \n{4}",c,d,e,f,g);

Relational and comparison operator

int a = 10;

int b = 20;

bool c = a == b; //equal to

bool d = a != b; //not equal to

bool e = a > b; //greater than

bool f = a < b; //less than

bool h = a >= b; //greater than equal

bool g = a <= b; // less than equal

Console.WriteLine("{0} \n{1}\n {2}\n {3}\n {4}\n {5}",c,d,e,f,g,h);

False

True

False

True

True

False

Logical and conditional operators

There are two types of conditional operator

And &&

Or ||

And

True true true

True false false

False true false

False false false

int a = 10;

int b = 20;

false false

bool c = (a > b) && (b < a);

Console.WriteLine(c);

False

int a = 10;

int b = 20;

false true

bool c = (a > b) && (a < b);

Console.WriteLine(c);

False

int a = 10;

int b = 20;

true true

bool c = (b > a) && (a < b);

Console.WriteLine(c);

True

Or operator ||

True true true

True false true

False true true

False false false

int a = 10;

int b = 20;

bool c = (b > a) || (a < b);

Console.WriteLine(c);

True

int a = 10;

int b = 20;

bool c = (b < a) || (a > b);

Console.WriteLine(c);

False

Assignment operator

There is two type of operator

Simple assignment operator

Coumpound assignment operator

+=

-=

\*=

/=

%=

int a = 5;

a += 1;

Console.WriteLine(a);

6

int a = 5;

a -= 1;

Console.WriteLine(a);

4

int a = 5;

a \*= 2;

Console.WriteLine(a);

10

int a = 5;

a /= 2;

Console.WriteLine(a);

2

int a = 5;

a %= 2;

Console.WriteLine(a);

1

Increment and decrement

int a = 5;

Console.WriteLine(a++); //post increment

5

int a = 5;

Console.WriteLine(a++); //post increment

Console.WriteLine(a);

6

int a = 5;

Console.WriteLine(a++); //pre increment

6

Decrement

int a = 5;

Console.WriteLine(a--); //post decrement

5

int a = 5;

Console.WriteLine(a--); //post decrement

Console.WriteLine(a);

4

int a = 5;

Console.WriteLine(--a); //pre decrement

4

Urinary operators work with one operand

Bninary operators work with two operands

Ternary operators work with three operands

Ternary operator use for decision making operators

?:

int a = 5;

string b = (a > 10) ? "a is greater" : "a is lesser";

Console.WriteLine(b);

A is lesser

int a = 20;

string b = (a > 10) ? "a is greater" : "a is lesser";

Console.WriteLine(b);

A is greater

int a = int.Parse(Console.ReadLine());

string b = (a >= 18) ? "your are eligible" : "you are not eligible";

Console.WriteLine(b);

your are eligible

precedent of operators

priority sequence

/\*+-

int a = 8 / 2 - 3 + 2 \* 2;

Console.WriteLine(a);

5

int a = 8 / 2 - 3 + 2 \* 2;

//4-3+2\*2

//4-3+4

//1+4

//5

Console.WriteLine(a);

Highest precedent lever is ()

int a = 8 / 2 - (3 + 2) \* 2;

//8/2-5\*2

// 4-5\*2

// 4-10

// -6

Console.WriteLine(a);

-6

Decision making if and else condition

Decision making constructor

If else

If else if

Nested if

Switch case

Nested switch case

If

int a = 20; //if condition is true then print a is greater

if (a > 10)

{

Console.WriteLine("a is greater");

}

a is greater

int a = 5;

if (a > 10)

{

Console.WriteLine("a is greater");

}

else

{

Console.WriteLine("a is lesser");

}

a is lesser

int a = int.Parse(Console.ReadLine());

if (a > 10)

{

Console.WriteLine("a is greater");

}

else

{

Console.WriteLine("a is lesser");

}

With string

string name = Console.ReadLine();

if (name == "mangal singh")

{

Console.WriteLine("welcome mangal singh");

}

else

{

Console.WriteLine("wrong details");

}

Username and password

Console.WriteLine("enter username");

string username = Console.ReadLine();

Console.WriteLine("enter your password");

string password = Console.ReadLine();

if(username == "mangal" && password == "mangal")

{

Console.WriteLine("login successfully");

}

else

{

Console.WriteLine("incorect username and password");

}

If else if

int percent = int.Parse (Console.ReadLine());

if (percent >= 80)

{

Console.WriteLine("A grade");

}

else if (percent >= 60)

{

Console.WriteLine("b grade");

}

else if (percent >= 40)

{

Console.WriteLine("c grade");

}

else

{

Console.WriteLine("fail");

}

Nested if

int a = 4;

if (a > 5)

{

if(a == 10)

{

Console.WriteLine("a is 10");

}

else

{

Console.WriteLine("a is not 10");

}

}

else

{

Console.WriteLine("a is lesser");

}

A is lesser

int a = 8;

if (a > 5)

{

if(a == 10)

{

Console.WriteLine("a is 10");

}

else

{

Console.WriteLine("a is not 10");

}

}

else

{

Console.WriteLine("a is lesser");

}

A is not 10

int a = 2;

if (a > 5)

{

if(a == 10)

{

Console.WriteLine("a is 10");

}

else

{

Console.WriteLine("a is not 10");

}

}

else

{

if (a == 3)

{

Console.WriteLine("a is 3");

}

else

{

Console.WriteLine("a is not 3");

}

}

A is not 3

Nested email and password

Console.WriteLine("please enter your email");

string email = Console.ReadLine();

if (email == "mangal@gmail.com")

{

Console.WriteLine("please enter your password");

string password = Console.ReadLine();

if (password == "1234")

{

Console.WriteLine("login successfully");

}

else

{

Console.WriteLine("incorrect password");

}

}

else

{

Console.WriteLine("incorrect email");

}

Swit

Switch case

int week\_number = 4;

switch (week\_number)

{

case 1:

Console.WriteLine("monday");

break;

case 2:

Console.WriteLine("tuesday");

break;

case 3:

Console.WriteLine("webnesday");

break;

case 4:

Console.WriteLine("thursday");

break;

case 5:

Console.WriteLine("friday");

break;

case 6:

Console.WriteLine("saturday");

break;

case 7:

Console.WriteLine("sunday");

break;

default:

Console.WriteLine("invalid number");

break;

}

Console.WriteLine("after switch");

string name = "mangal";

switch (name)

{

case "lal":

Console.WriteLine("welcome lal");

break;

case "mangal":

Console.WriteLine("welcome mangal");

break;

case "vishu":

Console.WriteLine("welcome vishu");

break;

default:

Console.WriteLine("invalid name");

break;

}

int age = int.Parse(Console.ReadLine());

switch (age)

{

case 15:

case 16:

case 17:

Console.WriteLine("you are not eligible for vote");

break;

case 18:

Console.WriteLine("now you can vote");

break;

}

Nested switch case

string pizza = Console.ReadLine();

switch (pizza)

{

case "fajita":

Console.WriteLine("you have slected fajita pizza");

break;

case "cheese":

Console.WriteLine("enter Your Cheese quantity");

string Cheese = Console.ReadLine();

switch (Cheese)

{

case "extra Cheese":

Console.WriteLine("you have selected extra cheese");

break;

case "normal Cheese":

Console.WriteLine("you have selected normal cheese");

break;

default:

Console.WriteLine("invalid key");

break;

}

break;

case "veg":

Console.WriteLine("Enter you vegetable");

string veg = Console.ReadLine();

switch (veg)

{

case "tomato":

Console.WriteLine("you have selected tomato");

break;

case "onion":

Console.WriteLine("you have selected onion");

break;

case "ginger":

Console.WriteLine("you have selected ginger");

break;

default:

Console.WriteLine("this vegetable is not available");

break;

}

break;

case "non-veg":

Console.WriteLine("you have selected non-veg");

break;

default:

Console.WriteLine("invalid flavour");

break;

}

Marksheet Project

Console.WriteLine("enter your name:");

string name =Console.ReadLine();

Console.WriteLine("enter your Roll Number:");

string roll = Console.ReadLine();

Console.WriteLine("enter your standard:");

string standard = Console.ReadLine();

Console.WriteLine("-----------Enter Your Subject marks-----------");

Console.WriteLine("Hindi");

int h = int.Parse(Console.ReadLine());

Console.WriteLine("English:");

int e = int.Parse(Console.ReadLine());

Console.WriteLine("Physics:");

int p = int.Parse(Console.ReadLine());

Console.WriteLine("Chemestry:");

int c = int.Parse(Console.ReadLine());

Console.WriteLine("Math:");

int m = int.Parse(Console.ReadLine());

Console.WriteLine("-------------Marksheet-----------");

Console.WriteLine("Name:{0}", name);

Console.WriteLine("Name:{0}", roll);

Console.WriteLine("Name:{0}", standard);

int obt = h + e + p + c + m;

int per = obt \* 100 / 500;

Console.WriteLine("Total Marks {0}/500",obt);

Console.WriteLine("Percentage {0}/100", per);

if (per >= 80)

{

Console.WriteLine("Grade: A" );

}

else if (per >= 60)

{

Console.WriteLine("Grade: B");

}else if (per >= 40)

{

Console.WriteLine("Grade: C");

}

else if(per >= 35)

{

Console.WriteLine("Grade: E");

}

else

{

Console.WriteLine("Grade: F");

}

int kt = 0;

if (h < 35)

{

int hmarks = 35 - h;

Console.WriteLine("Your Are Fail In Hindi with {0}",hmarks);

kt++;

}

if (e < 35)

{

int emarks = 35 - e;

Console.WriteLine("Your Are Fail In English with {0}", emarks);

kt++;

}

if (p < 35)

{

int pmarks = 35 - p;

Console.WriteLine("Your Are Fail In Physics with {0}", pmarks);

kt++;

}

if (c < 35)

{

int cmarks = 35 - c;

Console.WriteLine("Your Are Fail In Chemestry with {0}", cmarks);

kt++;

}

if (m < 35)

{

int mmarks = 35 - m;

Console.WriteLine("Your Are Fail In Maths with {0}", mmarks);

kt++;

}

if(per >= 80 && kt <= 0) {

Console.WriteLine("Remarks: Excellent");

}

else if(per >= 60 && kt <= 0)

{

Console.WriteLine("Remarks: VeryGood");

}

else if(per >= 40 && kt <= 0)

{

Console.WriteLine("Remarks: Good");

}

else if(per >= 35 && kt <= 0)

{

Console.WriteLine("Remarks: Poor");

}

else

{

Console.WriteLine("Remarks: Fail");

}

if (kt > 0)

{

Console.WriteLine("you have kt in {0} subject", kt);

}

if ((h >= 35) && (e >= 35) && (p >= 35) && (c >= 35) && (m >= 35))

{

Console.WriteLine("Hoorrey You have passed");

}

Loop Statement

Loop statement is also called as iteration statement

Ther is 4 types of loops

For loop

While loop

Do while loop

Foreach loop

For Loops

There have 3 thing in for loop

Initialization

Condition

Increment/decrement

for(

int i =0; //Initialization

i <10; //Condition

i++ //increment

)

{

Console.WriteLine(i);

}

Print number table

int number = int.Parse(Console.ReadLine());

for(

int i =0;

i <= 10;

i++

)

{

Console.WriteLine(number + " X " +i+"= "+ number\*i);

}

Decrement

for (int i = 10; i > 0; i--)

{

Console.WriteLine(i);

}

While Loop

int i = 0;

while (i < 10)

{

Console.WriteLine(i);

i++;

}

Number table

int number = 2;

int i = 0;

while (i <= 10)

{

Console.WriteLine(number+ " X " + i +" = "+ number\*i);

i++;

}

Do While Loop

int i = 0;

do

{

Console.WriteLine(i);

i++;

} while (i <= 10);

int i = 1;

do

{

Console.WriteLine(i);

i++;

} while (i < 0);

Nested for loop

for(int i = 0; i <= 3; i++)

{

Console.WriteLine("hellow");

for(int j =0; j < 3; j++)

{

Console.WriteLine("hey ");

}

}

Jump Statement

Types of jump

Break

Continue

Goto

Return

Break statement

for(int i = 0; i <= 10; i++)

{

if(i == 5)

{

break;

}

Console.WriteLine(i);

}

Console.WriteLine("loop terminated");

Continue statement

With even number

for(int i = 1; i <= 10; i++)

{

if(I % 2 == 1)

{

continue;

}

Console.WriteLine(i);

}

Console.WriteLine("loop terminated");

}

Odd number

for(int i = 0; i <= 10; i++)

{

if(I % 2 == 0)

{

continue;

}

Console.WriteLine(i);

}

Console.WriteLine("loop terminated");

}

GOTO statement

for(int i = 0; i <= 10; i++)

{

if(i == 5)

{

goto Stop;

}

Console.WriteLine(i);

}

Console.WriteLine("loop terminated");

Stop:

Console.WriteLine("this function will work");

How to restart c# programm with user input

string confirm;

do

{

Console.WriteLine("enter your first number");

int num1 = int.Parse(Console.ReadLine());

Console.WriteLine("enter your second number");

int num2 = int.Parse(Console.ReadLine());

int add = num1 + num2;

Console.WriteLine("your addition is {0}", add);

Console.WriteLine("do you want to repeat yes or no ");

//confirm =Console.ReadLine();

confirm =Console.ReadLine().ToLower();

}while(confirm == "yes");

With while loop

while (true)

{

Console.WriteLine("enter your first number");

int num1 = int.Parse(Console.ReadLine());

Console.WriteLine("enter your second number");

int num2 = int.Parse(Console.ReadLine());

int add = num1 + num2;

Console.WriteLine("your addition is {0}", add);

Console.WriteLine("do you want to repeat yes or no ");

string confirm = Console.ReadLine().ToLower();

if (confirm == "yes")

{

continue;

}

else

{

break;

}

}

Arrays

An array is collection of elements of single data

Type stored in adjacent memory locations

1st method

int[] m\_array = new int[4];

m\_array[0] = 33;

m\_array[1] = 44;

m\_array[2] = 55;

m\_array[3] = 66;

Console.WriteLine(m\_array[3]);

66

string[] m\_array = new string[4];

m\_array[0] = "mangal";

m\_array[1] = "amit";

m\_array[2] = "rakesh";

m\_array[3] = "gautam";

Console.WriteLine(m\_array[3]);

Gautam

2nd method

string[] m\_array = new string[4] { "mangal", "rakesh", "amit", "gautam" };

Console.WriteLine(m\_array[3]);

string[] m\_array = new string[] { "mangal", "rakesh", "amit", "gautam" };

Console.WriteLine(m\_array[3]);

3rd method

string[] m\_array = { "mangal", "rakesh", "amit", "gautam" };

Console.WriteLine(m\_array[3]);

Array length

string[] m\_array = { "mangal", "rakesh", "amit", "gautam" };

Console.WriteLine(m\_array.length);

Foreach loop with array

Arrays with for

1st method

string[] names = new string[5];

names[0] = "mangal";

names[1] = "amit";

names[2] = "gautam";

names[3] = "rakesh";

names[4] = "shivam";

for (int i = 0; i < names.Length; i++)

{

Console.WriteLine(names[i]);

}

2nd method

string[] names = new string[] {

"mangal",

"rakesh",

"vishu",

"mahi",

"renu",

"tripti"

};

for (int i = 0; i < names.Length; i++)

{

Console.WriteLine(names[i]);

}

3rd method

string[] names = {

"mangal",

"rakesh",

"vishu",

"mahi",

"renu",

"tripti"

};

for (int i = 0; i < names.Length; i++)

{

Console.WriteLine(names[i]);

}

Foreach method

static void Main(string[] args)

{

string[] names = {

"mangal",

"rakesh",

"vishu",

"mahi",

"renu",

"tripti"

};

foreach (string items in names)

{

Console.WriteLine(items);

}

int[] names = {

2,

3,

5,

6,

9,

10

};

foreach (int items in names)

{

Console.WriteLine(items);

}

Types of arrays single and multi dimensional

Single dimensional array data store contegus memory

static void Main(string[] args)

{

string[] names = {

"mangal",

"rakesh",

"vishu",

"mahi",

"renu",

"tripti"

};

foreach (string items in names)

{

Console.WriteLine(items);

}

Multi dimensional array

Rectangular array

Jagged array

Rectangular array

int[,] number = new int[3, 4]

{

{ 1,2,3,4},

{ 5,6,7,8},

{9,10,11,12 },

};

Console.WriteLine(number[0,2]);

Array length

int[,] number = new int[3, 4]

{

{ 1,2,3,4},

{ 5,6,7,8},

{9,10,11,12 },

};

Console.WriteLine(number.GetLength(1));

Rank dimensions

int[,] number = new int[3, 4]

{

{ 1,2,3,4},

{ 5,6,7,8},

{9,10,11,12 },

};

Console.WriteLine(number.Rank);

2

Nested for loop for multidimensional arrays

int[,] number = new int[3, 4]

{

{ 1,2,3,4},

{ 5,6,7,8},

{9,10,11,12 },

};

for (int i =0; i < number.GetLength(0);i++)

{

for (int j = 0; j < number.GetLength(1); j++)

{

Console.WriteLine(number[i,j] + " ");

}

}

int[,] number = new int[3, 4]

{

{ 1,2,3,4},

{ 5,6,7,8},

{9,10,11,12 },

};

for (int i =0; i < number.GetLength(0);i++)

{

for (int j = 0; j < number.GetLength(1); j++)

{

Console.Write(number[i,j] + " ");

}

Console.WriteLine();

}

int[,] number = new int[,]

{

{ 1,2,3,4},

{ 5,6,7,8},

{9,10,11,12 },

};

for (int i =0; i < number.GetLength(0);i++)

{

for (int j = 0; j < number.GetLength(1); j++)

{

Console.Write(number[i,j] + " ");

}

Console.WriteLine();

}

int[,] number =

{

{ 1,2,3,4},

{ 5,6,7,8},

{9,10,11,12 },

};

for (int i =0; i < number.GetLength(0);i++)

{

for (int j = 0; j < number.GetLength(1); j++)

{

Console.Write(number[i,j] + " ");

}

Console.WriteLine();

}

Working with foreach

int[,] number = new int[3, 4]

{

{ 1,2,3,4},

{ 5,6,7,8},

{9,10,11,12 },

};

foreach (var item in number)

{

Console.WriteLine(item);

}

Jagged array

int[][] numbers = new int[3][];

numbers[0] = new[] {1,2,3,4,5,6 };

numbers[1] = new[] { 7, 8, 9, 10, 11, 12 };

numbers[2] = new[] { 13, 14, 15, 16, 17, 18 };

for (int i = 0; i < numbers.GetLength(0); i++)

{

for (int j = 0; j < numbers[i].Length; j++)

{

Console.WriteLine(numbers[i][j]);

}

}

Foreach method jagged array

int[][] numbers = new int[3][];

numbers[0] = new[] {1,2,3,4,5,6 };

numbers[1] = new[] { 7, 8, 9, 10, 11, 12 };

numbers[2] = new[] { 13, 14, 15, 16, 17, 18 };

foreach (var item in numbers)

{

foreach (var i in item)

{

Console.WriteLine(i);

}

}

Creating array with user input

Console.WriteLine("enter your array number");

int num = int.Parse(Console.ReadLine());

int[] numbers = new int[num];

for (int i = 0; i < num; i++)

{

Console.WriteLine("enter your data");

int data = int.Parse(Console.ReadLine());

numbers[i] = data;

}

foreach (var item in numbers)

{

Console.WriteLine(item);

}

--------------------

String type

Console.WriteLine("enter your array number");

int num = int.Parse(Console.ReadLine());

string[] numbers = new string[num];

for (int i = 0; i < num; i++)

{

Console.WriteLine("enter your data");

string data = Console.ReadLine();

numbers[i] = data;

}

foreach (var item in numbers)

{

Console.WriteLine(item);

}

--------------------------------------------

Console.WriteLine("enter your array number");

int num = int.Parse(Console.ReadLine());

string[] numbers = new string[num];

for (int i = 0; i < num; i++)

{

Console.WriteLine("enter your data" + (i+1));

string data = Console.ReadLine();

numbers[i] = data;

}

foreach (var item in numbers)

{

Console.WriteLine(item);

}

Method or function

A method is a group of statement that together perform a task

It is used to perform the specific task

Methods are reusable s

Methods are function which is declare in class

public void show()//declaration //non static method// instance method

{

Console.WriteLine("hey bro");

Console.WriteLine("bro");

}

static void Main(string[] args)

{

Program program = new Program();

program.show();

}

Function reusable

public void show()//declaration //non static method// instance method

{

Console.WriteLine("hey bro");

Console.WriteLine("bro");

}

static void Main(string[] args)

{

Program program = new Program();

program.show();

program.show();

program.show();

}

Static method

public static void show1() //declaration //static method

{

Console.WriteLine("hey bro");

Console.WriteLine("bro");

}

static void Main(string[] args)

{

Program.show1();

}

Parameterize function

public static void show1(int num1, int num2) //declaration //static method

{

int add = num1 + num2;

Console.WriteLine(add);

}

static void Main(string[] args)

{

Program.show1(20,30);

}

public static void show1(int num1, int num2) //declaration //static method

{

int add = num1 + num2;

Console.WriteLine(add);

}

static void Main(string[] args)

{

Program.show1(20,30);

Program.show1(50, 30);

Program.show1(60, 30);

}

Input type

public static void show1(string name , int age) //declaration //static method

{

Console.WriteLine("your nameis {0} and age is{1} ",name,age);

}

static void Main(string[] args)

{

Console.WriteLine("enter your name");

string name = Console.ReadLine();

Console.WriteLine("enter your age");

int age = int.Parse(Console.ReadLine());

Program.show1(name,age);

}

Default parameter value

public static void show1( int age = 18) //declaration //static method

{

Console.WriteLine("your age is {0}",age);

}

static void Main(string[] args)

{

Program.show1();

}

public static void show1( int age = 18) //declaration //static method

{

Console.WriteLine("your age is {0}",age);

}

static void Main(string[] args)

{

Program.show1(24);

}

24

Named argument

public static void show1( string name, int age) //declaration //static method

{

Console.WriteLine("your name is {0}", name);

Console.WriteLine("your age is {0}",age);

}

static void Main(string[] args)

{

Program.show1(name:"mangal",age:26); //named argument

}

Return calling method

Return statement

public static int show1( int num1, int num2) //declaration //static method

{

int result = num1 + num2;

return (result);

}

static void Main(string[] args)

{

Console.WriteLine(Program.show1(20, 30));

}

public static int show1( int num1, int num2) //declaration //static method

{

int result = num1 + num2;

return (result);

}

static void Main(string[] args)

{

int a = Program.show1(20, 30);

int b = 30;

int result = a + b;

Console.WriteLine(result);

}

Creating a calculator

public static void add(int a, int b)

{

int result = a + b;

Console.WriteLine("your result is {0}",result);

}

public static void sub(int a, int b)

{

int result = a - b;

Console.WriteLine("your result is {0}", result);

}

public static void mul(int a, int b)

{

int result = a \* b;

Console.WriteLine("your result is {0}", result);

}

public static void div(int a, int b)

{

int result = a / b;

Console.WriteLine("your result is {0}", result);

}

static void Main(string[] args)

{

Console.WriteLine("Please enter your Number");

int num1 = int.Parse(Console.ReadLine());

Console.WriteLine("Please enter your sec number");

int num2 = int.Parse(Console.ReadLine());

Console.WriteLine("enter your operators + - \* / ");

string op = Console.ReadLine();

if (op == "+")

{

Program.add(num1,num2);

}else if(op == "-")

{

Program.sub(num1, num2);

}

else if(op == "\*")

{

Program.mul(num1, num2);

}

else

{

Program.div(num1, num2);

}

}

Value type and reference type

Stack and heap is a storage type

Struct --- value type --- stack

Class ---- reference --- heap

Struck value type

struct employee

{

public int age;

public int salary;

}

class Program

{

static void Main(string[] args)

{

employee e = new employee();

e.age = 25;

e.salary = 2000;

employee e1 = e;

employee e2 = e;

e.age = 26;

Console.WriteLine(e1.age);

Console.WriteLine(e2.age);

Console.WriteLine(e.age);

}

}

25

25

26

Class reference type

class employee

{

public int age;

public int salary;

}

class Program

{

static void Main(string[] args)

{

employee e = new employee();

e.age = 25;

e.salary = 2000;

employee e1 = e;

employee e2 = e;

e.age = 26;

Console.WriteLine(e1.age);

Console.WriteLine(e2.age);

Console.WriteLine(e.age);

}

}

26

26

26

Pass by value and pass by reference

Pass by value

Pass by value create a copy sent to passbyvalue

Then change the value in passbyvalue it not effect on main method value where I create a copy

public static void passbyvalue(int a)

{

a = a + 10;

Console.WriteLine(a);

}

static void Main(string[] args)

{

int value = 15;

passbyvalue(value);

Console.WriteLine(value);

}

25

15

Pass by reference

In that case we create a reference main to passbyvalue

If passbyvalue change the value it also change in main

class Program

{

public static void passbyvalue(ref int a)

{

a = a + 10;

Console.WriteLine(a);

}

static void Main(string[] args)

{

int value = 15;

passbyvalue(ref value);

Console.WriteLine(value);

}

}

25

25

Pass By Out

public static void passbyout(out int a)

{

a = 10;

Console.WriteLine(a);

}

static void Main(string[] args)

{

int value;

passbyout(out value);

Console.WriteLine(value);

}

Var and dynamic

--

Clases and object

Class

class Program

{

int rollno;

string name;

int age;

int std;

public void setrollno(int rollno)

{

this.rollno = rollno;

}

public int getrollno()

{

return this.rollno;

}

static void Main(string[] args)

{

Program ali = new Program();

ali.setrollno(20);

ali.getrollno();

Console.WriteLine(ali.getrollno());

}

}

class Program

{

int rollno;

string name;

int age;

int std;

public void setrollno(int rollno,string name, int age,int std)

{

this.rollno = rollno;

this.name = name;

this.age = age;

this.std = std;

}

public void getrollno()

{

dynamic[] myarray =

{

this.rollno,

this.name,

this.age,

this.std

};

int i;

for (i = 0; i < myarray.Length; i++)

{

Console.WriteLine(myarray[i]);

};

}

static void Main(string[] args)

{

Program ali = new Program();

ali.setrollno(20,"mangal",20,7);

ali.getrollno();

}

}

class Program

{

int rollno;

string name;

int age;

int std;

public void setrollno(int rollno,string name, int age,int std)

{

this.rollno = rollno;

this.name = name;

this.age = age;

this.std = std;

}

public void getrollno()

{

dynamic[] myarray =

{

this.rollno,

this.name,

this.age,

this.std

};

int i;

for (i = 0; i < myarray.Length; i++)

{

Console.WriteLine(myarray[i]);

};

}

static void Main(string[] args)

{

Program ali = new Program();

ali.setrollno(20,"mangal",20,7);

Program rahul = new Program();

rahul.setrollno(21, "angal", 0, 1);

ali.getrollno();

rahul.getrollno();

}

}

class Program

{

int rollno;

string name;

int age;

int std;

public void setrollno(int rollno,string name, int age,int std)

{

this.rollno = rollno;

this.name = name;

this.age = age;

this.std = std;

}

public dynamic getrollno()

{

dynamic[] myarray =

{

this.rollno,

this.name,

this.age,

this.std

};

return myarray;

}

static void Main(string[] args)

{

Program ali = new Program();

ali.setrollno(20,"mangal",20,7);

for (int i = 0; i < ali.getrollno().Length; i++)

{

Console.WriteLine(ali.getrollno()[i]);

};

}

}

Constructor

Default constructor

class Program

{

public Program()

{

}

static void Main(string[] args)

{

}

}

Parameterize constructor

class Program

{

int empid;

string emp\_name;

int age;

public Program(int empid, string emp\_name,int age)

{

this.empid = empid;

this.emp\_name = emp\_name;

this.age = age;

}

public dynamic getdata()

{

dynamic[] myarray =

{

this.empid,

this.emp\_name,

this.age

};

return myarray;

}

static void Main(string[] args)

{

Program emp = new Program(11,"mangal",22);

emp.getdata();

foreach (var item in emp.getdata())

{

Console.WriteLine(item);

}

}

}

Many constructor & constructor overloading

Default and parameterized constructor is also called instance constructor

class Program

{

public Program()

{

Console.WriteLine("this is my first constructor");

}

public Program(int a, int b)

{

Console.WriteLine("this is my second constructor {0}", a+b);

}

public Program(int a, int b, int c)

{

Console.WriteLine("this is my first constructor{0}",a+b+c);

}

static void Main(string[] args)

{

Program emp = new Program(20,30);

}

}

Static constructor

Static constructor does not take any parameterize

Static constructor cannot use any access specifier like private public

We can create Only one static constructor in a class

Static constructor runs only one time but default constructor can run many times as per object how much have

Static and instance members of class

Instance variable

class student

{

public int id;

public string name;

public int age;

}

class Program

{

static void Main(string[] args)

{

student ali = new student();

ali.id = 2;

ali.name = "mangal";

ali.age = 25;

Console.WriteLine("you info{0}{1}{2}",ali.id,ali.name,ali.age); }

}

class person

{

public static string personName;

public static int personAge;

static person()

{

personName = "ali";

personAge = 23;

Console.WriteLine("static constructor invoke ");

}

public person()

{

Console.WriteLine("default constructor is invoked ");

}

public static void getdetails()

{

Console.WriteLine("person name:", personName);

Console.WriteLine("person Age:", personAge);

}

}

class Program

{

static void Main(string[] args)

{

person umar = new person();

}

}

Copy constructor

The constructor which create object by coping variable from onother object is called copy constructor

class example

{

string name;

int age;

public example(string name, int age)

{

this.name = name;

this.age = age;

}

public void getdata()

{

Console.WriteLine("your name is {0}", name);

Console.WriteLine("your age is {0}", age);

}

public example(example e) // Copy Constructor

{

this.name = e.name;

this.age = e.age;

}

}

class Program

{

static void Main(string[] args)

{

example obj = new example("mangal", 25);

obj.getdata();

example obj1 = new example(obj);

obj1.getdata();

}

}

Private constructor

Private constructor means its not give you allow to create object and inheritance

If the class have all members are static means we don’t need to create a object

In private constructor can create parameter but there is no use

class Example

{

public static int a;

public string name;

public string fname;

public Example(string name, string fname)

{

this.name = name;

this.fname = fname;

}

private Example()

{

}

public static int getincrement()

{

return ++a;

}

}

class Program

{

static void Main(string[] args)

{

Example obj = new Example("mangal","singh");

Example.a = 20;

Console.WriteLine(Example.getincrement());

}

}

Instance method and variable

This variable create a object copy

class student

{

public int id;

public string name;

public int age;

public void fullinfo()

{

Console.WriteLine("Your Id is {0} and Name {1} and Age{2}",this.id,this.name,this.age);

}

}

class Program

{

static void Main(string[] args)

{

student ali = new student();

ali.id = 2;

ali.name = "mangal";

ali.age = 25;

ali.fullinfo();

Console.WriteLine("---------------------");

student zain = new student();

zain.id = 3;

zain.name = "rohit";

zain.age = 26;

zain.fullinfo();

}

}

Static variable

Static school name

Static variable call with class name like student.schoolname

class student

{

public int id;

public string name;

public int age;

public static string schoolname = "abc school";

public void fullinfo()

{

Console.WriteLine("Your Id is {0} and Name {1} and Age{2} and school {3}",this.id,this.name,this.age,student.schoolname);

}

}

class Program

{

static void Main(string[] args)

{

student ali = new student();

ali.id = 2;

ali.name = "mangal";

ali.age = 25;

ali.fullinfo();

Console.WriteLine("---------------------");

student zain = new student();

zain.id = 3;

zain.name = "rohit";

zain.age = 26;

zain.fullinfo();

}

}

Static function

class student

{

public int id;

public string name;

public int age;

public static string schoolname = "abc school";

public static int fees = 2000;

public void fullinfo()

{

Console.WriteLine("Your Id is {0} and Name {1} and Age{2} and school {3}",this.id,this.name,this.age,student.schoolname);

}

public static int feesinfo()

{

return fees;

}

}

class Program

{

static void Main(string[] args)

{

student ali = new student();

ali.id = 2;

ali.name = "mangal";

ali.age = 25;

ali.fullinfo();

Console.WriteLine(student.feesinfo());

Console.WriteLine("---------------------");

student zain = new student();

zain.id = 3;

zain.name = "rohit";

zain.age = 26;

zain.fullinfo();

Console.WriteLine(student.feesinfo());

}

}

Instance variable use with object

Static variable use with class

Instance function use with object

Static function use with class

static function cannot use in instance variable

In instance method we can use instance and static variable

Static class

Cannot create a object of static class and cannot be inherited

static class product

{

public static int productId;

public static string productName;

public static int productPrice;

static product()

{

productId = 11;

productName = "headphone";

productPrice = 500;

}

public static void getdetails()

{

Console.WriteLine("productID {0}", productId);

Console.WriteLine("productName {0}", productName);

Console.WriteLine("productPrice {0}", productPrice);

}

public static void discount()

{

int discount = productPrice / 10;

Console.WriteLine("your Discount Amount: {0}", discount);

Console.WriteLine("Total Price OF Product: {0}",(productPrice - discount));

}

}

class Program

{

static void Main(string[] args)

{

product.getdetails();

product.discount();

}

}

}

Destructor

class person

{

public string name;

public int age;

public person(string name, int age)

{

this.name = name;

this.age = age;

}

public string getname()

{

return this.name;

}

public int getage()

{

return this.age;

}

~person()

{

Console.WriteLine("destructor invoked ");

}

}

class Program

{

static void Main(string[] args)

{

person p = new person("mangal",25);

Console.WriteLine(p.getname());

Console.WriteLine(p.getage());

Console.WriteLine("-------------");

person n = new person("singh", 33);

Console.WriteLine(n.getname());

Console.WriteLine(n.getage());

}

}

Inheritence

class emp : human

{

public int empid;

}

class person : human

{

public int personid;

}

class human

{

public int id;

public string name;

public int age;

}

class Program

{

static void Main(string[] args)

{

emp e = new emp();

e.age = 11;

person p = new person();

p.name = "mangal";

Console.WriteLine(p.name);

Console.WriteLine(e.age);

}

}

Types of Inheritence

Single inheritance

Hierarchical

Multi level

Multiple

Single inheritance

class baseclass

{

public static string name = "mangal";

}

class derivedclass : baseclass

{

public static int age = 21;

}

class Program

{

static void Main(string[] args)

{

Console.WriteLine(derivedclass.age);

Console.WriteLine(derivedclass.name);

}

}

Hierarchical

class baseclass {

public static int id = 21;

public static string name = "mangal";

public static int salary = 31000;

}

class child1 : baseclass {

public static int DEP\_id = 4;

public static string Dep\_name = "hr";

}

class child2 : baseclass {

public static int Dep\_salary = 4000;

}

class Program

{

static void Main(string[] args)

{

Console.Write(child1.id + " " + child1.name+ " " + child1.salary + " "+ child1.DEP\_id+ " "+ child1.Dep\_name+ " "+ child2.Dep\_salary);

}

}

Multi lever inheritance

class baseclass {

public static int id = 21;

public static string name = "mangal";

public static int salary = 31000;

}

class child1 : baseclass {

public static int DEP\_id = 4;

public static string Dep\_name = "hr";

}

class child2 : child1 {

public static int Dep\_salary = 4000;

}

class Program

{

static void Main(string[] args)

{

Console.WriteLine(Convert.ToString(child2.id)+" "+ child2.name+" "+ child2.salary+" "+child2.DEP\_id+" "+child2.Dep\_name+" "+child2.Dep\_salary);

}

}

Multiple inheritance

This mean create a many class on project folder and we can inheritance to each other by parent and child

class Program : Sales

{

public dynamic tot\_det()

{

Prod\_Name = "mangal";

return Prod\_Name;

}

static void Main(string[] args)

{

Program p = new Program();

Console.WriteLine(p.tot\_det());

}

}

Constructor inheritance

class baseclass

{

public baseclass( string message)

{

Console.WriteLine(message);

}

}

class derivedclass : baseclass

{

public derivedclass() :base("heelloo world")

{

Console.WriteLine("derived class");

}

}

class Program

{

static void Main(string[] args)

{

derivedclass dc = new derivedclass();

}

}

Access specifier

Access modifier

Access Modifier

Public

Private

Protected

Internal

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Applicable to  The Application | Applicable to the current  Class | Applicable to the derived  Class | Applicable to outside the namespace /assembly | Applicable to the outside the  Namespace but in derived class |
| Public | Yes | Yes | Yes | Yes | Yes |
| private | No | Yes | No | No | No |
| protected | No | Yes | Yes | No | Yes |
| Internal | Yes | Yes | Yes | No | No |
|  |  |  |  |  |  |

Encapsulation

Encapsulation is mechanism of wrapping the data and code acting on the data (methods and properties )

As a single unit

class person

{

private int age;

private string name;

public void setperson(int age)

{

this.age = age;

}

public void getperson()

{

Console.WriteLine(this.age);

}

}

class Program

{

static void Main(string[] args)

{

person p = new person();

p.setperson(20);

p.getperson();

}

}

class person

{

private string name;

private int age;

public void setname(string name)

{

if (string.IsNullOrEmpty(name) == true)

{

Console.WriteLine("Name is Required");

}

else

{

this.name = name;

}

}

public void getname()

{

if(string.IsNullOrEmpty(this.name) == true)

{

}

else

{

Console.WriteLine(this.name);

}

}

public void setage(int age)

{

if (age > 0)

{

this.age = age;

}

else

{

Console.WriteLine("Age Is required");

}

}

public void getage()

{

if(this.age > 0)

{

Console.WriteLine(this.age);

}

else

{

Console.WriteLine("invalid age");

}

}

}

class Program

{

static void Main(string[] args)

{

person p = new person();

p.setname("");

p.getname();

p.setage(22);

p.getage();

}

}

Properties with set and get encapsulation

class person

{

private int id;

private string name;

private int age;

public int ID

{

set

{

if(value <= 0)

{

Console.WriteLine("value not zero and negetive");

}

else

{

this.id = value;

}

}

get

{

return this.id;

}

}

public string Name

{

set

{

if(string.IsNullOrEmpty(value) == true)

{

Console.WriteLine("Name value is requred");

}

else

{

this.name = value;

}

}

get

{

return this.name;

}

}

public int Age

{

set

{

if(value < 0)

{

Console.WriteLine("The Age not valid");

}

else

{

this.age = value;

}

}

get

{

return this.age;

}

}

}

class Program

{

static void Main(string[] args)

{

person p = new person();

p.ID = 1;

p.Name = "mangal";

p.Age = 23;

Console.WriteLine(p.Name +" "+p.ID+" "+ p.Age );

}

}

Read only properties

class person

{

private int id;

private string name;

private int age = 22;

public int AGE

{

get

{

return this.age;

}

}

}

class Program

{

static void Main(string[] args)

{

person p = new person();

Console.WriteLine(p.AGE);

}

}

Write only

class person

{

private int id;

private string name;

private int age = 22;

public int ID

{

set

{

this.id = value;

}

}

}

class Program

{

static void Main(string[] args)

{

person p = new person();

p.ID = 2;

}

}

Properties with get and set

class person

{

public string firstname { get; set; }

public string lastname { get; set; }

public person(string fname, string lname)

{

this.firstname = fname;

this.lastname = lname;

}

}

class Program

{

static void Main(string[] args)

{

person p = new person("mangal","singh");

Console.WriteLine(p.firstname + "" + p.lastname);

}

}

class person

{

public string firstname { get; set; }

public string lastname { get; set; }

public person(string fname, string lname)

{

this.firstname = fname;

this.lastname = lname;

}

}

class Program

{

static void Main(string[] args)

{

person p = new person("mangal","singh");

//you can change the value as well

p.firstname = "santosh";

p.lastname = "singh";

Console.WriteLine(p.firstname + "" + p.lastname);

}

}

class person

{

public string firstname { get; private set; }

public string lastname { get; private set; }

public person(string fname, string lname)

{

this.firstname = fname;

this.lastname = lname;

}

}

class Program

{

static void Main(string[] args)

{

person p = new person("mangal","singh");

Console.WriteLine(p.firstname + "" + p.lastname);

}

}

Static properties

class person

{

public static string name;

public static string lname;

public static string \_name {

set {

if(string.IsNullOrEmpty(value))

{

Console.WriteLine("Name is required");

}

else

{

name = value;

}

}

get {

return name;

}

}

}

class Program

{

static void Main(string[] args)

{

person.\_name = "mangal";

Console.WriteLine(person.\_name);

}

}

Polymorphism

Polymorphism in C# concept by which can perform the single task by different ways

Polymorphism is derived from 2 greek word poly and morphs

The word poly means many and morphs means forms

So polymorphism means many forms

Two types of polymorphism

Static polymorphism(compile time polymorphism)

Dynamic polymorphism(run time polumorphism)

Static polymorphism

The mechanism of linking a function with an object during compile time called static polymorphism

Or early binding

This provides two technique

Operator overloading

Method or function overloading

class methodoverloading

{

public void add()

{

int a = 20;

int b = 30;

int c = a + b;

Console.WriteLine(c);

}

public void add(int a , int b)

{

int c = a + b;

Console.WriteLine(c);

}

public void add(string a, string b)

{

string c = a + " " + b;

Console.WriteLine(c);

}

}

class Program

{

static void Main(string[] args)

{

methodoverloading m = new methodoverloading();

m.add();

m.add(20, 20);

m.add("mangal", "singh");

}

Operator overloading

This is use for combine the objects to other object

class operatoroverloading

{

public string str;

public int num;

public static operatoroverloading operator +(operatoroverloading ob1, operatoroverloading ob2)

{

operatoroverloading ob3 = new operatoroverloading();

ob3.str = ob1.str +" "+ ob2.str;

ob3.num = ob1.num + ob2.num;

return ob3;

}

}

class Program

{

static void Main(string[] args)

{

operatoroverloading ob1 = new operatoroverloading();

ob1.str = "mangal";

ob1.num = 25;

operatoroverloading ob2 = new operatoroverloading();

ob2.str = "singh";

ob2.num = 30;

operatoroverloading ob3 = new operatoroverloading();

ob3 = ob1 + ob2;

Console.WriteLine(ob3.str);

Console.WriteLine(ob3.num);

}

}

Polymorphism method overriding

class parent

{

public virtual void show()

{

Console.WriteLine("this is parent");

}

}

Class child: parent

{

public override void show()

{

Console.WriteLine("this is child");

}

}

class Program

{

static void Main(string[] args)

{

Parent p = new child();

p.show();

}

}

Method hiding in inheritance

class parent

{

public void show()

{

Console.WriteLine("this is parent");

}

}

class child: parent

{

public new void show()

{

Console.WriteLine("this is child");

}

}

class Program

{

static void Main(string[] args)

{

child c = new child();

c.show();

}

}

Base class method run

class parent

{

public void show()

{

Console.WriteLine("this is parent");

}

}

class child: parent

{

public new void show()

{

base.show();

}

}

class Program

{

static void Main(string[] args)

{

child c = new child();

c.show();

}

}

Casting

class parent

{

public void show()

{

Console.WriteLine("this is parent");

}

}

class child: parent

{

public new void show()

{

}

}

class Program

{

static void Main(string[] args)

{

child c = new child();

((parent)c).show();

}

}

class parent

{

public void show()

{

Console.WriteLine("this is parent");

}

}

class child: parent

{

public new void show()

{

}

}

class Program

{

static void Main(string[] args)

{

parent p = new child();

p.show();

}

}

Sealed class

Sealed class is class that prevent inheritance

sealed class parent

{

public void show()

{

Console.WriteLine("this is parent");

}

}

sealed class parent

{

public void show()

{

Console.WriteLine("this is parent");

}

}

class child: parent // gave error on this class because parent class is sealed

{

public new void show()

{

}

}

Sealed methods

class A

{

public virtual void print()

{

Console.WriteLine("this is the class a ");

}

}

class B : A

{

public sealed override void print()

{

Console.WriteLine("this is the class b ");

}

}

class C : B

{

public override void print() // here got error because parent class got sealed

{

Console.WriteLine("this is the class c ");

}

}

class Program

{

static void Main(string[] args)

{

C c = new C();

c.print();

}

}

Sealed method always available with method overriding

Only we can sealed those method which are overriding

INDEXER

class employees

{

private int[] employee = new int[3];

public int this[int index]

{

set {

employee[index] = value;

}

get {

return employee[index];

}

}

}

class Program

{

static void Main(string[] args)

{

employees e = new employees();

e[0] = 5;

Console.WriteLine(e[0]);

}

}

Delegates

A person sent or authorized to represent others in particular elect representative sent to a conference

Types of delegates

Multiple delegates

Single cast delegates

Multi cast delegates

Single cast delegates

namespace ConsoleApp1

{

public delegate void calculation(int a, int b);

class Program

{

public static void addition (int a , int b)

{

int result = a + b;

Console.WriteLine("your addtion is {0}", result);

}

static void Main(string[] args)

{

calculation obj = new calculation(Program.addition);

obj.Invoke(20, 30);

}

}

}

Multi cast delegates

namespace ConsoleApp1

{

public delegate void calculation(int a, int b);

class Program

{

public static void addition (int a , int b)

{

int result = a + b;

Console.WriteLine("your addtion is {0}", result);

}

public static void sub(int a, int b)

{

int result = a - b;

Console.WriteLine("your subs is {0}", result);

}

public static void mul(int a, int b)

{

int result = a \* b;

Console.WriteLine("your mul is {0}", result);

}

public static void div(int a, int b)

{

int result = a / b;

Console.WriteLine("your div is {0}", result);

}

static void Main(string[] args)

{

calculation obj = new calculation(Program.addition);

obj.Invoke(20, 30);

obj = sub;

obj(10, 23);

obj = mul;

obj(3, 4);

obj = div;

obj(50, 10);

}

}

}

namespace ConsoleApp1

{

public delegate void calculation(int a, int b);

class Program

{

public static void addition (int a , int b)

{

int result = a + b;

Console.WriteLine("your addtion is {0}", result);

}

public static void sub(int a, int b)

{

int result = a - b;

Console.WriteLine("your subs is {0}", result);

}

public static void mul(int a, int b)

{

int result = a \* b;

Console.WriteLine("your mul is {0}", result);

}

public static void div(int a, int b)

{

int result = a / b;

Console.WriteLine("your div is {0}", result);

}

static void Main(string[] args)

{

calculation obj = new calculation(Program.addition);

obj += sub;

obj += mul;

obj += div;

obj(20, 30);

}

}

}

Multi delegates

namespace ConsoleApp1

{

public delegate void calculation(int a, int b);

public delegate void show\_d();

public delegate void mul\_(int a);

class Program

{

public static void show()

{

Console.WriteLine("non parameterized delegates ");

}

public static void addition (int a , int b)

{

int result = a + b;

Console.WriteLine("your addtion is {0}", result);

}

public static void mul(int a)

{

int result = a \* a;

Console.WriteLine("your addtion is {0}", result);

}

static void Main(string[] args)

{

calculation obj = new calculation(Program.addition);

obj.Invoke(20, 30);

show\_d obj2 = new show\_d(Program.show);

obj2();

mul\_ obj3 = new mul\_(mul);

obj3(20);

}

}

}

Anonymous function

We discussed that delegates are used to reference any methods that has the same signature as that of the delegates

namespace ConsoleApp1

{

public delegate void calculation(int num);

class Program

{

public static void mymethod(calculation cal, int a)

{

a += 10;

cal.Invoke(a);

}

static void Main(string[] args)

{

Program.mymethod(

delegate (int b)

{

b += 10;

Console.WriteLine(b);

},5

);

}

}

}

Simple function

public delegate int calculation(int a);

class Program

{

static void Main(string[] args)

{

calculation obj = delegate (int a)

{

a += 5;

Console.WriteLine(a);

};

obj(5);

}

}

namespace ConsoleApp1

{

public delegate int calculation(int a);

class Program

{

static void Main(string[] args)

{

calculation obj = delegate (int a)

{

a += 5;

return a;

};

Console.WriteLine(obj(5));

}

}

}

Lambda expression

It is also work like anonymous method

Lambda exp have two types

Statement lambda

Expression lambda

Simple anonymous function

namespace ConsoleApp1

{

public delegate void mydelegagate(int a);

class Program

{

static void Main(string[] args)

{

mydelegagate obj = delegate(int b){

b += 6;

Console.WriteLine(b);

};

obj(5);

}

}

}

With lambda expression

Statement lambda

namespace ConsoleApp1

{

public delegate void mydelegagate(int a);

class Program

{

static void Main(string[] args)

{

mydelegagate obj = (b) => //lambda expresion

{

b += 6;

Console.WriteLine(b);

};

obj(5);

}

}

}

namespace ConsoleApp1

{

public delegate int mydelegagate(int a);

class Program {

static void Main(string[] args)

{

mydelegagate obj = (b) => //lambda expresion

{

b += 6;

return b;

};

Console.WriteLine(obj(5));

}

}

}

Expression lambda

namespace ConsoleApp1

{

public delegate int mydelegagate(int a);

class Program {

static void Main(string[] args)

{

mydelegagate obj = (b) => b \* b;

Console.WriteLine(obj(5));

}

}

}

Abstraction

class Abstraction {

public int eid;

public string name;

public int grosspay;

public int tax = 10;

public int netsalary;

public Abstraction(int Emp\_id,string Ename, int Egross)

{

this.eid = Emp\_id;

this.name = Ename;

this.grosspay = Egross;

}

void cal()

{

if (grosspay >= 30000)

{

netsalary = grosspay - (grosspay / 10);

Console.WriteLine("your net salary is {0}", netsalary);

}

else

{

Console.WriteLine("your net salary is {0}", grosspay);

}

}

public void show()

Console.WriteLine(this.eid);

Console.WriteLine(this.name);

cal();

}

static void Main(string[] args)

{

Abstraction obj = new Abstraction(333,"mangal",40000);

obj.show();

}

}

Abstract class and method

We cannot create object of abstract class

class Abstraction {

abstract class person

{

public string fname;

public string lname;

public int age;

public long number;

public abstract void show();

}

class student:person

{

public int rollno;

public int fees;

public override void show()

{

string name = this.fname + " " + this.lname;

Console.WriteLine("name: {0}",name);

Console.WriteLine(this.age);

Console.WriteLine(this.rollno);

Console.WriteLine(this.number);

Console.WriteLine(this.fees);

}

}

class teacher: person

{

public string qualification;

public int salary;

public override void show()

{

string name = this.fname + " " + this.lname;

Console.WriteLine("name: {0}", name);

Console.WriteLine(this.age);

Console.WriteLine(this.qualification);

Console.WriteLine(this.number);

Console.WriteLine(this.salary);

}

}

static void Main(string[] args)

{

student st = new student();

st.fname = "mangal";

st.lname = "singh";

st.age = 20;

st.number = 23423424343;

st.rollno = 33;

st.fees = 400;

st.show();

teacher tc = new teacher();

tc.fname = "rakesh";

tc.lname = "yadav";

tc.age = 55;

tc.qualification = "BCA";

tc.number = 23423424343;

tc.salary = 70000;

tc.show();

}

}

Abstract properties