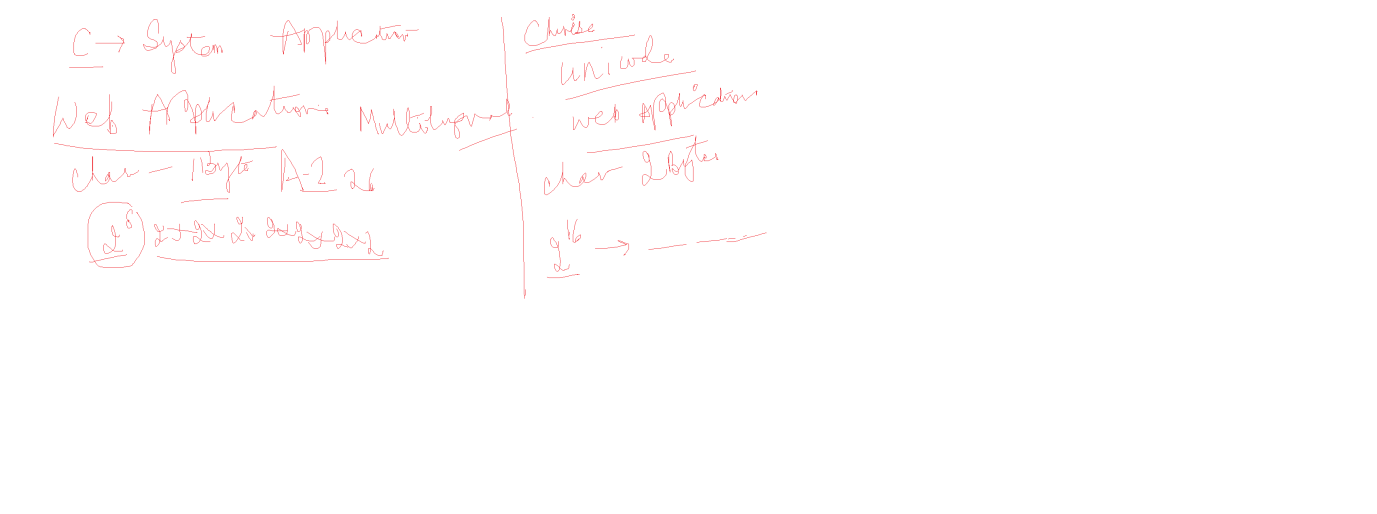
Data Types





* Int 1 2 3 9008

In C , we use ASCII code

Int > 2 bytes

Char > 1 Byte

Web Applications : MultiLingual

In C language

1 char > 1 Byte , 8 bits

2^8 =

C#, Java > UNICODE

Int > 4 bytes

Char > 2 bytes

1 char > 2 Byte , 16 bits

2^16 =

Convert.ToInt16(Console.ReadLine())

Convert.ToInt32(Console.ReadLine())

Convert.ToByte(Console.ReadLine()) > 1 byte , 8 bits (0 – 255)

Convert.ToSByte(Console.ReadLine()) > 1 byte , 8 bits (-128 – 127)

Convert.ToUInt16(Console.ReadLine())

Signed > +909

-9

Unsigned

98

ToInt16

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| + | 8 | 9 | 8 | 9 | 8 | 9 | 7 | 9 | 7 |

2^15

Unsigned > 2 ^ 16

* Char
* Float
* Double
* Boolean
* Date

|  |
| --- |
|  |
| Type | Description | Range | Suffix |
| Byte | 8-bit unsigned integer | 0 to 255 |  |
| Sbyte | 8-bit signed integer | -128 to 127 |  |
| Short | 16-bit signed integer | -32,768 to 32,767 |  |
| Ushort | 16-bit unsigned integer | 0 to 65,535 |  |
| Int | 32-bit signed integer | -2,147,483,648 to 2,147,483,647 |  |
| Uint | 32-bit unsigned integer | 0 to 4,294,967,295 | u |
| long | 64-bit signed integer | -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 | l |
| ulong | 64-bit unsigned integer | 0 to 18,446,744,073,709,551,615 | ul |
| float | 32-bit Single-precision floating point type | -3.402823e38 to 3.402823e38 | f |
| double | 64-bit double-precision floating point type | -1.79769313486232e308 to 1.79769313486232e308 | d |
| decimal | 128-bit decimal type for financial and monetary calculations | (+ or -)1.0 x 10e-28 to 7.9 x 10e28 | m |
| char | 16-bit single Unicode character | Any valid character, e.g. a,\*, \x0058 (hex), or\u0058 (Unicode) |  |
| bool | 8-bit logical true/false value | True or False |  |
| object | Base type of all other types. |  |  |
| string | A sequence of Unicode characters |  |  |
| DateTime | Represents date and time | 0:00:00am 1/1/01 to 11:59:59pm 12/31/9999 |  |

<https://www.tutorialsteacher.com/csharp/csharp-data-types>

using System; // namespace

class Program // One class is mandatory

{

static void Main() // Its entry and exit point of your program

{

int n;

Console.WriteLine("Enter No");

//n = Convert.ToInt16(Console.ReadLine());

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

Console.WriteLine(n);

}

}

Functions > Modular Programming

Breaking a big program into subprograms

Why ?

Easy to understand a small program

Easy to debug a same program

**Reusability** > Once created , can be used multiple times

Tested

Optimized

Cost effective

Saves efforts

Save cost

How do we make function

<return type> function\_name(parameters)

{

Statements;

}

using System; // namespace

class Program // One class is mandatory

{

static void Main()

{

int num1, num2;

Console.WriteLine("Enter Number 1");

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

//num1 = Convert.ToByte( Console.ReadLine());

Console.WriteLine("Enter Number 2");

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

Console.WriteLine("Sum of {0} and {1} is {2}", num1, num2 ,

(num1 + num2));

Console.WriteLine("Difference of {0} and {1} is {2}", num1, num2,

(num1 - num2));

Console.WriteLine("Product of {0} and {1} is {2}", num1, num2,

(num1 \* num2));

Console.WriteLine("Quotient of {0} and {1} is {2}", num1, num2,

(num1 / num2));

}

}

Above program, we divided into methods

using System; // namespace

class Program // One class is mandatory

{

static int num1, num2;

static void Add()

{

Console.WriteLine("Sum of {0} and {1} is {2}", num1, num2,

(num1 + num2));

}

static void Subtract()

{

Console.WriteLine("Difference of {0} and {1} is {2}", num1, num2,

(num1 - num2));

}

static void Product()

{

Console.WriteLine("Product of {0} and {1} is {2}", num1, num2,

(num1 \* num2));

}

static void Divide()

{

Console.WriteLine("Quotient of {0} and {1} is {2}", num1, num2,

(num1 / num2));

}

static void GetValues()

{

//{

// Convert.ToInt16();

// Int16.Parse();

// Convert.ToByte()

// Byte.Parse()

Console.WriteLine("Enter Number 1");

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

//num1 = Convert.ToByte( Console.ReadLine());

Console.WriteLine("Enter Number 2");

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

}

static void Main()

{

GetValues();

Add();

Subtract();

Product();

Divide();

}

}

How do we make function

How do we insert Breakpoint

How do we debug by pressing F5 and F11

How do we see value of variables in auto & local windows during debug time

Passing parameters to methods

using System; // namespace

class Program // One class is mandatory

{

static void Add(int num1 , int num2)

{

Console.WriteLine("Sum of {0} and {1} is {2}", num1, num2,

(num1 + num2));

}

static void Subtract(int num1, int num2)

{

Console.WriteLine("Difference of {0} and {1} is {2}", num1, num2,

(num1 - num2));

}

static void Product(int num1, int num2)

{

Console.WriteLine("Product of {0} and {1} is {2}", num1, num2,

(num1 \* num2));

}

static void Divide(int num1, int num2)

{

Console.WriteLine("Quotient of {0} and {1} is {2}", num1, num2,

(num1 / num2));

}

static void Main()

{

int num1, num2;

Console.WriteLine("Enter Number 1");

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

//num1 = Convert.ToByte( Console.ReadLine());

Console.WriteLine("Enter Number 2");

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

Add(num1 , num2);

Subtract(num1 , num2 );

Product(num1 , num2);

Divide(num1 , num2 );

}

}

Function returns an int value

using System; // namespace

class Program // One class is mandatory

{

//static string name;

//static int age;

//static string country;

//static void GetDetails()

//{

// Console.WriteLine("Enter Name");

// name = Console.ReadLine();

// Console.WriteLine("Enter Age");

// age = Convert.ToByte(Console.ReadLine());

// Console.WriteLine("Enter Country");

// country = Console.ReadLine();

//}

//static void Display()

//{

// Console.WriteLine("Welcome {0}. Your age is {1} and you are from {2}",

// name, age, country);

//}

static int Add(int num1, int num2)

{

return num1 + num2;

}

static int Subtract(int num1, int num2)

{

return num1 - num2;

}

static int Product(int num1, int num2)

{

return num1 \* num2;

}

static int Divide(int num1, int num2)

{

return num1 / num2;

}

static void Main()

{

//GetDetails();

//Display();

int num1, num2;

Console.WriteLine("Enter Number 1");

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

//num1 = Convert.ToByte( Console.ReadLine());

Console.WriteLine("Enter Number 2");

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

int result = Add(num1 , num2);

Console.WriteLine("Sum of {0} and {1} is {2}", num1, num2,

result);

result = Subtract(num1 , num2 );

Console.WriteLine("Difference of {0} and {1} is {2}", num1, num2,

(num1 - num2));

result = Product(num1 , num2);

Console.WriteLine("Product of {0} and {1} is {2}", num1, num2,

(num1 \* num2));

result = Divide(num1 , num2 );

Console.WriteLine("Quotient of {0} and {1} is {2}", num1, num2,

(num1 / num2));

}

}

Break and continue in loops

Break is used

* In switch (we use it with every case)
* Loops : to exit out of the loop

Continue is used in loops , to skip the statements of the current pass and it takes you to the next iteration

Check a no. for even or odd

using System; // namespace

class Program // One class is mandatory

{

static void Main()

{

int num;

num = Convert.ToInt16(Console.ReadLine());

Console.WriteLine("Enter Number");

if(num % 2 ==0)

Console.WriteLine("Its even");

else

Console.WriteLine("its odd");

}

}

Check a no. for prime

using System; // namespace

class Program // One class is mandatory

{

static void Main()

{

int num;

bool isPrime = true;

Console.WriteLine("Enter Number");

num = Convert.ToInt16(Console.ReadLine());

for(int i = 2; i<num/2;i++)

{

if(num % i ==0)

{

isPrime = false;

break;

}

}

if(isPrime== true)

Console.WriteLine("Its a Prime no");

else

Console.WriteLine("Its not a Prime no");

}

}