**Exercise 1: Write C# code to declare a variable to store the age of a person. Then the output of the program is as an example shown below:**

You are 20 years old.

**Solution:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace Csharp\_exercises

{

class Program

{

static void Main(string[] args)

{

int age = 20;// declaring variable and assign 20 to it.

Console.WriteLine("You are {0} years old.",age);

Console.ReadLine();

}

}

}

**Exercise 2: Write C# code to display the asterisk pattern as shown below:**

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

**Solution:**

using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
  
namespace Csharp\_exercises  
{  
class Program  
{  
static void Main(string[] args)  
{  
  
   Console.WriteLine("\*\*\*\*\*");  
   Console.WriteLine("\*\*\*\*\*");  
   Console.WriteLine("\*\*\*\*\*");  
   Console.WriteLine("\*\*\*\*\*");  
   Console.WriteLine("\*\*\*\*\*");  
   Console.ReadLine();   
        }  
    }  
}

**Exercise 3: Write C# code to declare two integer variables, one float variable, and one string variable and assign 10, 12.5, and "C# programming" to them respectively. Then display their values on the screen.**

**Solution:**

using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
  
namespace Csharp\_exercises  
{  
class Program  
{  
static void Main(string[] args)  
{  
  
int x;  
float y;  
string s;  
x = 10;  
y = 12.5f;  
s = "C# programming";  
Console.WriteLine(x);  
Console.WriteLine(y);  
Console.WriteLine(s);  
Console.ReadLine();   
}  
}  
}

**Exercise 4: Write C# code to prompt a user to input his/her name and then the output will be shown as an example below:**

Hello John!

**Solution:**

using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
  
namespace Csharp\_exercises  
{  
class Program  
{  
static void Main(string[] args)  
   {  
  
  string name;  
  Console.Write("Please enter your name:");  
  name = Console.ReadLine();   
  Console.WriteLine("Hello {0}!", name);  
  Console.ReadLine();   
         }  
   }  
}

## C# operators exercises

﻿

### Exercise 1: Write C# code to produce the output shown below:

x value        y value        expression                 result

10                       5              x=y+3                     x=8

10                       5              x=y-2                      x=3

10                       5              x=y\*5                     x=25

10                       5              x=x/y                      x=2

10                       5              x=x%y                    x=0

**Solution:**

using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
  
namespace Csharp\_exercises  
{  
class Program  
{  
static void Main(string[] args)  
{  
  
  
int x=10;  
  
int y=5;  
  
Console.WriteLine("Result:");  
  
Console.WriteLine("x value\t\ty value\t\tExpressions\tResult");  
  
Console.WriteLine("{0,-8}\t{1,-8}\tx=y+3 \t x={2,-8}",x,y,y + 3);  
Console.WriteLine("{0,-8}\t{1,-8}\tx=y-2 \t x={2,-8}", x, y, y-2);  
Console.WriteLine("{0,-8}\t{1,-8}\tx=y\*5 \t x={2,-8}", x, y, y\*5);  
Console.WriteLine("{0,-8}\t{1,-8}\tx=x/y \t x={2,-8}", x, y, (float)x/y);  
Console.WriteLine("{0,-8}\t{1,-8}\tx=x%y \t x={2,-8}", x, y, x%y);  
  
  
Console.ReadLine();   
   }  
  }  
}

### Exercise 2: Write C# code to display the output as shown below:

Results:

x value                 y value        expressions                results

10                        5                  x+=y                          x=15

10                        5                  x-=y-2                        x=7

10                        5                  x\*=y\*5                       x=250

10                        5                  x/=x/y                         x=5

10                        5                  x%=y                          x=0

**Solution:**

using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
  
namespace Csharp\_exercises  
{  
class Program  
{  
static void Main(string[] args)  
{  
  
  
int x=10;  
  
int y=5;  
  
  
Console.WriteLine("Result:");  
  
Console.WriteLine("x value\t\ty value\t\tExpressions\tResult");  
  
Console.WriteLine("{0,-8}\t{1,-8}\tx+=y \t x={2,-8}",x,y,x+y);  
Console.WriteLine("{0,-8}\t{1,-8}\tx-=y-2 \t x={2,-8}", x, y,x-y+2);  
Console.WriteLine("{0,-8}\t{1,-8}\tx\*=y\*5 \t x={2,-8}", x, y, x\*y\*5);  
Console.WriteLine("{0,-8}\t{1,-8}\tx=x/y \t x={2,-8}", x, y, (float)x/(x/y));  
Console.WriteLine("{0,-8}\t{1,-8}\tx%=y \t x={2,-8}", x, y, x%y);  
  
  
Console.ReadLine();   
}  
}  
}

### Exercise 3: Write C# code to prompt a user to enter an integer value.

The value is stored in a variable called **a**. Then the program will show the output as seen below:

The value of a is 10.

................................

The value of ++a is 11.

The value of a is 11.

The value of a++is 11.

The value of a is 12.

The value of --a is 11.

The value of a is 11.

The value of a-- is 11.

The value of a is 10.

**Solution:**

using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
  
namespace Csharp\_exercises  
{  
class Program  
{  
static void Main(string[] args)  
{  
  
  
int a;  
int b;  
  
Console.Write("Enter a value:");  
  
  
a=int.Parse(Console.ReadLine()) ;  
  
Console.WriteLine("The value of a is {0}.",a);  
  
b=++a;  
  
Console.WriteLine("The value of ++a is {0}.",b);  
  
Console.WriteLine("The value of a is {0}.",a);  
b = a++;  
  
Console.WriteLine("The value of a++ is {0}.", b);  
Console.WriteLine("The value of a is {0}.", a);  
  
b=--a;  
  
Console.WriteLine("The value of --a is {0}.", b);  
Console.WriteLine("The value of a is {0}.", a);  
  
  
b=a--;  
  
Console.WriteLine("The value of a-- is {0}.", b);  
Console.WriteLine("The value of a is {0}.", a);  
  
  
Console.ReadLine();   
    }  
  }  
}

## C# Conditional statements if else exercises

﻿

### Exercise 1: Write a C# program that prompts the user to input three integer values and find the greatest value of the three values.

**Solution:**

using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
  
namespace Csharp\_exercises  
{  
class Program  
{  
static void Main(string[] args)  
{  
  
  
int x,y,z;  
  
Console.Write("Enter value 1:");  
x = int.Parse(Console.ReadLine());  
Console.Write("Enter value 2:");  
y = int.Parse(Console.ReadLine());  
Console.Write("Enter value 3:");  
z = int.Parse(Console.ReadLine());  
  
  
if (x > y)  
   if (x > z) Console.Write("The greatest value is:{0}.",x);  
   else Console.Write("The greatest value is:{0}.", z);  
else if (y > z) Console.Write("The greatest value is:{0}.",y);  
   else Console.Write("The greatest value is:{0}.",z);  
  
  
  
Console.ReadLine();   
    }  
  }  
}

### Exercise 2: Write a C# program that determines a student’s grade.

The program will read three types of scores(quiz score, mid-term score, and final score) and determine the grade based on the following rules:

-if the average score =90% =>grade=A  
-if the average score >= 70% and <90% => grade=B  
-if the average score>=50% and <70% =>grade=C  
-if the average score<50% =>grade=F

**Solution:**

using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
  
namespace Csharp\_exercises  
{  
class Program  
{  
static void Main(string[] args)  
{  
  
  
float quiz\_score;  
float mid\_score;  
float final\_score;  
float avg;  
Console.Write("Enter quiz score:");  
quiz\_score=float.Parse(Console.ReadLine());  
Console.Write("Enter mid-term score:");  
mid\_score = float.Parse(Console.ReadLine());  
Console.Write("Enter final score:");  
final\_score = float.Parse(Console.ReadLine());   
avg = (quiz\_score +mid\_score+final\_score) / 3;  
  
if (avg >= 90) Console.WriteLine("Grade A");  
else if ((avg >= 70) && (avg < 90)) Console.WriteLine("Grade B");  
else if ((avg >= 50) && (avg < 70)) Console.WriteLine("Grade C");  
else if (avg < 50) Console.WriteLine("Grade F");  
else Console.WriteLine("Invalid input");  
  
  
Console.ReadLine();   
     }  
  }  
}

## C# Conditional statements switch case

﻿

### Exercise 1: Write a C# program to detect key presses.

If the user pressed number keys( from 0 to 9), the program will display the number that is pressed, otherwise the program will show "Not allowed".  **Solution:**

using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
  
namespace Csharp\_exercises  
{  
class Program  
{  
static void Main(string[] args)  
{  
char key;  
Console.Write("Press a number key:");  
key = (char)Console.Read();  
switch (key)  
{  
case '0': Console.WriteLine("You pressed 0"); break;  
case '1': Console.WriteLine("You pressed 1"); break;  
case '2': Console.WriteLine("You pressed 2"); break;  
case '3': Console.WriteLine("You pressed 3"); break;  
case '4': Console.WriteLine("You pressed 4"); break;  
case '5': Console.WriteLine("You pressed 5"); break;  
case '6': Console.WriteLine("You pressed 6"); break;  
case '7': Console.WriteLine("You pressed 7"); break;  
case '8': Console.WriteLine("You pressed 8"); break;  
case '9': Console.WriteLine("You pressed 9"); break;  
default: Console.WriteLine("Not allowed!"); break;  
  
            }  
  
  
     }  
  }  
}

### Exercise 2: Write a C# program that allows the user to choose the correct answer of a question.

See the example below:

What is the correct way to declare a variable to store an integer value in C#?

a. int 1x=10;

b. int x=10;

c. float x=10.0f;

d. string x="10";

Choose the answer letter: c

Incorrect choice!

**Solution:**

using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
  
namespace Csharp\_exercises  
{  
class Program  
{  
static void Main(string[] args)  
{

Console.WriteLine("What is the correct way to declare a variable to store an integer value in C#?");  
Console.WriteLine("a. int 1x=10");  
Console.WriteLine("b. int x=10");  
Console.WriteLine("c. float x=10.0f");  
Console.WriteLine("d. string x=\"10\"");  
Console.WriteLine("Choose the answer letter:");  
char ans = (char)Console.read();  
switch (ans)  
{  
   case 'a':Console.WriteLine("Invalid choice!"); break;  
   case 'b':Console.WriteLine("Congratulation!"); break;  
  case 'c':Console.WriteLine("Invalid choice!"); break;  
  case 'd':Console.WriteLine("Invalid choice!"); break;  
  default:Console.WriteLine("Bad choice!");break;  
  
  
       }  
  
  
    }  
  
   }

}

## C# Loops: for loop exercises

﻿

### Exercise 1: Write C# code to produce the output shown below:

\*\*\*\*\*\*\*

\*\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*

\*\*\*

\*\*

\*

**Solution:**

using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
  
namespace Csharp\_exercises  
{  
class Program  
{  
static void Main(string[] args)  
{  
int i,j;  
for(i=0;i<=6;i++){  
  for(j=1;j<=7-i;j++) Console.Write("\*");  
Console.Write("\n");   
}  
  
  
Console.ReadLine();   
  
  
   }  
 }  
}

### Exercise 2: Write C# code to print the following pattern:

1\*\*\*\*\*\*

12\*\*\*\*\*

123\*\*\*\*

1234\*\*\*

12345\*\*

123456\*

1234567

**Solution:**

using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
  
namespace Csharp\_exercises  
{  
class Program  
{  
static void Main(string[] args)  
{  
int i,j,k;  
  
   for (i = 1; i <= 7; i++)  
    {  
       for (j = 1; j <= i; ++j)  
          Console.Write(j);  
  
       for (k = 7 - i; k >= 1; k--)  
          Console.Write("\*");  
  
 Console.Write("\n");  
}  
  
  
Console.ReadLine();   
  
  
  }  
 }  
**}**

## C# Loops: while loop exercises

﻿

### Exercise 1: Write C# program to prompt the user to choose the correct answer from a list of answer choices of a question.

The user can choose to continue answering the question or stop answering it. See the example below:

What is the command keyword to exit a loop in C#?

a. int

b. continue

c. break

d. exit

Enter your choice: b

Incorrect!

Again? press y to continue:

**Solution:**

uusing System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
  
namespace Csharp\_exercises  
{  
class Program  
{  
static void Main(string[] args)  
{  
string choice;  
string con="y";  
Console.WriteLine("What is the command keyword to exit a loop in C#?");  
Console.WriteLine("a.quit");  
Console.WriteLine("b.continue");  
Console.WriteLine("c.break");  
Console.WriteLine("d.exit");  
  
  
while (con=="y")  
{  
  Console.Write("Enter your choice:");  
  choice =Console.ReadLine();  
  
  if (choice == "c")  
  {  
    Console.WriteLine("Congratulation!");  
  }  
  else if (choice == "q" || choice == "e") { Console.WriteLine("Exiting...!");       break; }  
  else Console.WriteLine("Incorrect!");

        Console.Write("Again? press y to continue:");  
        con = Console.ReadLine().ToString();  
  
          }  
  
  
     }  
   }  
}

### Exercise 2: Write C# program to print the table of characters that are equivalent to the Ascii codes from 1 to 122.

The program will print the 10 characters per line.

**Solution:**

using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
  
namespace Csharp\_exercises  
{  
class Program  
{  
static void Main(string[] args)  
{  
  
int i =1;  
while (i <=122)  
{  
  Console.Write((char)i+"\t");  
  if (i % 10 == 0)  
    Console.Write("\n");  
  
  i++;  
}  
  
Console.ReadLine();   
        }  
    }  
}

## C# Loops: do while loop exercises

﻿

### Exercise 1:Write C# program to prompt the user to choose the correct answer from a list of answer choices of a question.

The user can choose to continue answering the question or stop answering it. See the example below:

What is the command keyword to exit a loop in C#?

a. int

b. continue

c. break

d. exit

Enter your choice: b

Incorrect!

Again? press y to continue:

**Solution:**

using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
  
namespace Csharp\_exercises  
{  
class Program  
{  
static void Main(string[] args)  
{  
  
string choice;  
string con = "y";  
Console.WriteLine("What is the command keyword to exit a loop in C#?");  
Console.WriteLine("a.quit");  
Console.WriteLine("b.continue");  
Console.WriteLine("c.break");  
Console.WriteLine("d.exit");  
  
  
do  
{  
    Console.Write("Enter your choice:");  
    choice = Console.ReadLine();  
  
    if (choice == "c")  
    {  
     Console.WriteLine("Congratulation!");  
     }  
   else if (choice == "q" || choice == "e") { Console.WriteLine("Exiting...!");        break; }  
    else Console.WriteLine("Incorrect!");  
  
    Console.Write("Again? press y to continue:");  
    con = Console.ReadLine().ToString();  
   } while (con == "y");  
}  
}  
}

### Exercises 2: Write C# program to print the table of characters that are equivalent to the Ascii codes from 1 to 122.

The program will print the 10 characters per line.

**Solution:**

using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
  
namespace Csharp\_exercises  
{  
class Program  
{  
static void Main(string[] args)  
{  
  
int i =1;  
do  
{  
  Console.Write((char)i+"\t");  
  if (i % 10 == 0)  
    Console.Write("\n");  
  
  i++;  
} while(i<=122);  
  
Console.ReadLine();   
}  
}  
}

## Random number in C#

﻿

### Exercise 1: Write a C# program to allow a user to guess a number( from 1 to 6) that will be randomly generated by computer.

The user is asked to input his/her number. Then the number will be compared with the random number. See the example below:

Enter your number: 2   
You lost.  
My number is: 4

**Solution:**

using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.IO;  
  
namespace ConsoleApplication1  
{  
  
  
class Program  
{  
  
  
  static void Main(string[] args)  
  {  
  
      ranguess();  
  
     Console.ReadLine();  
  
  }  
  
   public static void ranguess()  
  {  
        int yn, rn;   
        Random rd = new Random(); //create random object  
      
        Console.Write("Enter your guess number:");  
        yn = int.Parse(Console.ReadLine());  
        rn =rd.Next(1,7);//generate random number from 1 t 6  
        Console.WriteLine(rn);  
        if (yn == rn)  
        {  
          Console.WriteLine("You won.");  
  
        }  
      else  
        {  
           Console.WriteLine("You lost.");  
           Console.WriteLine("My number is {0}.", rn);  
  
        }  
  
  
  }  
 }  
  
}

### Exercise 2: Modify the C# program above in order to have chances to continue or stop guessing.

**Solution:**

using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.IO;  
  
namespace ConsoleApplication1  
{  
  
  
class Program  
{  
  
  
  static void Main(string[] args)  
  {  
  
      ranguess();  
  
     Console.ReadLine();  
  
  }  
  
   public static void ranguess()  
  {  
     int yn, rn;   
     int con=1;   
     Random rd = new Random(); //create random object  
     while (con == 1)  
      {  
         Console.Write("Enter your guess number:");  
         yn = int.Parse(Console.ReadLine());  
         rn =rd.Next(1,7);//generate random number from 1 t 6  
         Console.WriteLine(rn);  
        if (yn == rn)  
        {  
          Console.WriteLine("You won.");  
  
        }  
      else  
        {  
           Console.WriteLine("You lost.");  
           Console.WriteLine("My number is {0}.", rn);  
  
        }  
  
       Console.Write("Press 1 to continue:");  
       con = int.Parse(Console.ReadLine());  
  
  
    }  
  
  }  
 }  
  
}

## C# Arrays exercise: sort

﻿

### Exercise 1:By using the bubble sort algorithm, write C# code to sort an integer array of 10 elements in ascending.

**Solution:**

using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
  
namespace Csharp\_exercises  
{  
class Program  
{  
static void Main(string[] args)  
{  
int[] arr = new int[10] { 23, 2, 3, 34, 6,1,24,45,78,8}; //unsorted data set  
bubblesort(arr, 10); //sorting process using bubble sort  
int i;  
for (i = 0; i < 10; i++)  
Console.Write(arr[i] + "\t"); //after sorting in ascending order  
  
Console.ReadLine();  
  
}  
///bubble sort  
  
static void bubblesort(int[] dataset, int n)  
{  
int i, j;  
for (i = 0; i < n; i++)  
  for (j = n - 1; j > i; j--)  
   if (dataset[j] < dataset[j - 1])  
    {  
      int temp = dataset[j];  
      dataset[j] = dataset[j - 1];  
      dataset[j - 1] = temp;  
       }  
  
    }  
   }  
}

### Exercise 2: Modify the C# code in exercise 1 in order to sort the array in descending order.

**Solution:**

using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
  
namespace Csharp\_exercises  
{  
class Program  
{  
static void Main(string[] args)  
{  
int[] arr = new int[10] { 23, 2, 3, 34, 6,1,24,45,78,8}; //unsorted data set  
bubblesort(arr, 10); //sorting process using bubble sort  
int i;  
for (i = 0; i < 10; i++)  
Console.Write(arr[i] + "\t"); //after sorting in descending order  
  
Console.ReadLine();  
  
}  
///bubble sort  
  
static void bubblesort(int[] dataset, int n)  
{  
int i, j;  
for (i = 0; i < n; i++)  
  for (j = n - 1; j > i; j--)  
    if (dataset[j] > dataset[j - 1])  
     {  
       int temp = dataset[j];  
       dataset[j] = dataset[j - 1];  
       dataset[j - 1] = temp;  
         }  
  
    }  
  }  
}

## C# Arrays exercise: search (con)

﻿

### Exercise 1: By using the sequential search algorithm, write C# code to search for an element of an integer array of 10 elements.

**Solution:**

using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
  
namespace Csharp\_exercises  
{  
class Program  
{  
static void Main(string[] args)  
{  
  int[] arr = new int[10] { 23, 2, 3, 34, 6,1,24,45,78,8}; //data set  
  int pos,target;  
  Console.Write("Enter value to find:");  
  target = int.Parse(Console.ReadLine());  
  pos = seqsearch(arr, target, 10);  
  if (pos != -1)  
    Console.WriteLine("The target item was found at location:{0}", pos);  
  else  
    Console.WriteLine("The target item was not found in the list.\n");  
  Console.ReadLine();  
  
}  
///sequential search  
static int seqsearch(int[] dataset, int target, int n)  
{  
  int found = 0;  
  int i;  
  int pos = -1;  
  for (i = 0; i < n && found != 1; i++)  
    if (target == dataset[i]) { pos = i; found = 1; }  
  
   return pos;  
  }  
}  
**}**

### Exercise 2: Modify the C# code in exercise 1 in order to search for an element of  the array using binary search algorithm.

**Solution:**

using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
  
namespace Csharp\_exercises  
{  
class Program  
{  
static void Main(string[] args)  
{  
  
int[] arr = new int[10] { 23, 2, 3, 34, 6, 1, 24, 45, 78, 8 }; //unsorted data set  
int pos, target;  
Console.Write("Enter value to find:");  
target = int.Parse(Console.ReadLine());  
pos = binsearch(arr, 23, 10);  
if (pos != -1)  
Console.WriteLine("The target item was found at location:{0}", pos);  
else  
Console.WriteLine("The target item was not found in the list.\n");  
Console.ReadLine();  
  
}  
  
///binary search  
static int binsearch(int[] dataset,int target, int l,int u){  
  insertsort(dataset,dataset.Length);//make sure the list sorted  
  while(u>=l){

    int mid=(l+u)/2;

    if(target==dataset[mid]) return mid;

    else if(target>dataset[mid]) l=mid+1;

    else if(target<dataset[mid]) u=mid-1;

}  
return -1;  
}

static void insertsort(int[] dataset, int n)  
{  
  
int i, j;  
for (i = 1; i < n; i++)  
{  
int pick\_item = dataset[i];  
int inserted = 0;  
for (j = i - 1; j >= 0 && inserted != 1; )  
{  
if (pick\_item < dataset[j])  
{  
dataset[j + 1] = dataset[j];  
j--;  
dataset[j + 1] = pick\_item;  
}  
else inserted = 1;  
}  
}  
  
}  
}  
}

## C# array exercise: series of numbers

﻿

**Exercise:** A two-dimensional array stores values in rows and columns. By using two-dimensional array, write C# program to display a table of numbers as shown below:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |  |
| 6 | 7 |  | 9 | 10 |  |
| 11 | 12 | 13 | 14 | 15 |  |
| 16 | 17 | 18 | 19 | 20 |  |
| 21 | 22 | 23 | 24 | 25 |  |

**Solution:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.IO;

namespace ConsoleApplication1

{

  class Program

    {

        static void Main(string[] args)

        {

            printSeries();

            Console.ReadLine();

        }

        public static void printSeries()

        {

            int[,] tArr = new int[5, 5];

            int i, j;

                for (i = 0; i < 5; i++) //assign values to the two-dimensional array

                    for (j = 0; j < 5; j++)

                    {

                        if (i == 0) tArr[i, j] = j + 1; //fill the first row

                        else if (i > 0 && j == 0)

                            tArr[i, j] = tArr[i - 1, 4] + 1; //fetching the value of the last cell in the previous row

                        else

                         tArr[i, j] = tArr[i, j - 1] + 1; //fill subsequent cells

                    }

     for(i=0;i<5;i++){ //print the array

        for(j=0;j<5;j++)

            Console.Write("{0}\t",tArr[i,j]);

        Console.WriteLine();

       }

  }

 }

}

## C# array exercise: matrix

﻿

In this **C# exercise**, you are about to write C# program to display a matrix as shown below. The diagonal of the matrix fills with 0s. The lower side fills will -1s and the upper side fills with 1s.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 1 | 1 | 1 | 1 |  |
| -1 | 0 | 1 | 1 | 1 |  |
| -1 | -1 | 0 | 1 | 1 |  |
| -1 | -1 | -1 | 0 | 1 |  |
| -1 | -1 | -1 | -1 | 0 |  |

**Solution:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.IO;

namespace ConsoleApplication1

{

    class Program

    {

        static void Main(string[] args)

        {

            printMatrix();

            Console.ReadLine();

        }

        public static void printMatrix()

        {

            int[,] matrix = new int[5, 5];

            int i, j;

            for (i = 0; i < 5; i++) //assign values to the matrix

                for (j = 0; j < 5; j++)

                { //if row=column=> fill the matrix with 0

                    if (i == j) matrix[i, j] = 0;//if row>columns=> fill matrix with -1

                    else if (i > j) matrix[i, j] = -1;//if row<columns=> fill matrix with 1

                    else matrix[i, j] = 1;

                }

            for (i = 0; i < 5; i++)

            { //print the matrix

                for (j = 0; j < 5; j++)

                    Console.Write("{0}\t", matrix[i, j]);

                Console.WriteLine();

            }

        }

  }

}

## C# array exercise: Pascal triangle

﻿

**Exercise:** By using two-dimensional array of C# language, write C# program to display a table that represents a Pascal triangle of any size. In Pascal triangle, the first and the second rows are set to 1. Each element of the triangle (from the third row downward) is the sum of the element directly above it and the element to the left of the element directly above it. See the example Pascal triangle(size=5) below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 |  |  |  |  |
| 1 | 1 |  |  |  |
| 1 | 2 | 1 |  |  |
| 1 | 3 | 3 | 1 |  |
| 1 | 4 | 6 | 4 | 1 |

**Solution:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.IO;

namespace ConsoleApplication1

{

    class Program

    {

        static void Main(string[] args)

        {

            int size;

            Console.Write("Pascal triangle size:");

            size = Int32.Parse(Console.ReadLine());

            printPascalTr(size);

            Console.ReadLine();

        }

        public static void printPascalTr(int size){

         int[,] PascalTr=new int[size,size];

         int row,col;

         //assign zero to every array element

         for(row=0;row<size;row++)

            for(col=0;col<size;col++)  PascalTr[row,col]=0;

         //first and second rows are set to 1s

         PascalTr[0,0]=1;

         PascalTr[1,0]=1;

         PascalTr[1,1]=1;

         for(row=2;row<size;row++){

              PascalTr[row,0]=1;

              for(col=1;col<=row;col++){

                        PascalTr[row,col]=PascalTr[row-1,col-1]+PascalTr[row-1,col];

                                        }

                            }

          //display the Pascal Triangle

          for(row=0;row<size;row++){

            for(col=0;col<=row;col++){

                   Console.Write("{0}\t",PascalTr[row,col]);

                   }

            Console.WriteLine();

            }

         }

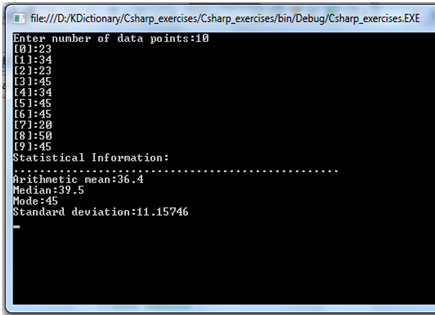
     }

}

## C# array exercise: statistical information

﻿

**Exercise :** Write a C# program to answer about the statistical information such as arithmetic mean, median, mode, and standard deviation of an integer data set. The data points are input by the user from keyboard. This program will display the output similar to the one shown below:



If you are not sure about statistical information such as arithmetic mean, median, mode, and standard deviation, you will need to read this page:

[Descriptive Measures](http://www.worldbestlearningcenter.com/index_files/Statistics-Probability-Chapter3-Descriptive-Measures.htm)

**Solution:**

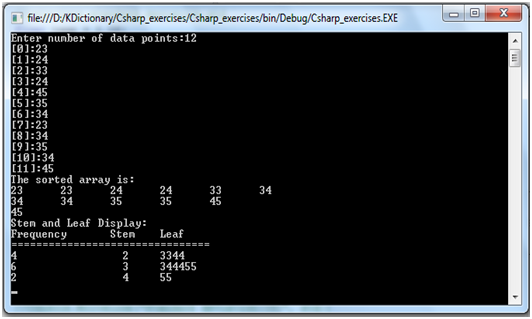
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
  
namespace Csharp\_exercises  
{  
class Program  
{  
static void Main(string[] args)  
{  
  int n;  
  float mean,median,std;  
  Console.Write("Enter number of data points:");  
  n = int.Parse(Console.ReadLine());  
  if (n < 3)  
   {  
    Console.WriteLine("The number of data points should be greater than 2.");  
  
    }  
  else  
  {  
  
    //declare an array of n size to store integral data points  
    int[] dataset = new int[n];  
   //allow user inputs  
   int i = 0;  
   for (i = 0; i < n; i++)  
    {  
      Console.Write("[{0}]:", i);  
      dataset[i] = int.Parse(Console.ReadLine());  
    }  
  
   //sort the data set  
   bubblesort(dataset, n);

  //calculate the mean  
   int sum = 0;  
   int j = 0;  
   while (j < n)  
  {  
     sum = sum + dataset[j];  
     j++;  
   }  
  
   mean = (float)sum / n;  
  
  //calculate median  
  //If n is odd, median=dataset[n/2]  
  //If n is even, median=(dataset[n/2]+dataset[1+n/2])/2  
  //The index of array starts from 0, so you need to subtract 1 from the           //indices   used in calculating the median  
  if (n % 2 != 0) median = dataset[n / 2];  
  else median = (dataset[(n / 2) - 1] + dataset[n / 2]) / (float)2;  
  
   //calculate the mode  
   int[,] mode = new int[n, 2];  
  //initialize 2D array storing numbers of occurences, and values  
  for (i = 0; i < 2; i++)  
  for (j = 0; j < n; j++) mode[j, i] = 0;  
  mode[0, 0] = 1;  
  
  for (i = 0; i < n; i++)  
    for (j = 0; j < n - 1; j++)  
      if (dataset[i] == dataset[j + 1]) { ++mode[i, 0]; mode[i, 1] = dataset[i]; }  
  
   int max;  
   int k = 0;  
   max = mode[0, 0];  
   for (j = 0; j < n; j++)  
     if (max < mode[j, 0]) { max = mode[j, 0]; k = j; }  
  
  
   //calculate standard deviation, std  
   float temp = 0.0f;  
  
   for (j = 0; j < n; j++)  
  {  
     temp = temp + (float)Math.Pow(dataset[j] - mean, 2);  
   }  
  
   std = (float)Math.Sqrt(temp / (n - 1));  
  
   //Show results  
  
   Console.WriteLine("Statistical Information:");  
   Console.WriteLine("..................................................");  
   Console.WriteLine("Arithmetic mean:{0}", mean);  
   Console.WriteLine("Median:{0}", median);  
   if (mode[k, 1] != 0)  
     Console.WriteLine("Mode:{0}", mode[k, 1]);  
   else Console.WriteLine("Mode: no mode");  
   Console.WriteLine("Standard deviation:{0}", std);  
  
  
 }  
   Console.ReadLine();   
  
}  
  
///bubble sort  
  
static void bubblesort(int[] dataset, int n)  
{  
int i, j;  
for (i = 0; i < n; i++)  
  for (j = n - 1; j > i; j--)  
    if (dataset[j] < dataset[j - 1])  
      {  
         int temp = dataset[j];  
         dataset[j] = dataset[j - 1];  
         dataset[j - 1] = temp;  
  }  
  
}  
}  
}

## C# array exercise: stem leaf

﻿

**Exercise :** Write a C# program to display an integer data set in the form of **stem and leaf**. The data points are input by the user from keyboard. This program will display the output similar to the one shown below:



If you are not sure how to display a data set in the form of stem and leaf, you will need to read this page:

[Stem and Leaf display](http://www.worldbestlearningcenter.com/index_files/Statistics-Probability-Chapter2-Present-numerical-data-stem-leaf.htm)

**Solution:**

using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
  
namespace Csharp\_exercises  
{  
class Program  
{  
static void Main(string[] args)  
{  
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
  
namespace Projecttest  
{  
class Program  
{  
static void Main(string[] args)  
{  
  
  int n;  
  Console.Write("Enter number of data points:");  
  n = int.Parse(Console.ReadLine());  
  if (n < 3)  
   {  
   Console.WriteLine("The number of data points should be greater than 2.");  
  
    }  
  else  
  {  
  
//declare an array of n size to store integral data points  
  int[] dataset = new int[n];  
//allow user inputs  
   int i = 0;  
   for (i = 0; i < n; i++)  
  {  
    Console.Write("[{0}]:", i);  
    dataset[i] = int.Parse(Console.ReadLine());  
   }  
  
//sort the data set  
   bubblesort(dataset, n);  
//show data in stem and leaf display  
//Display sorted array  
  Console.WriteLine("The sorted array is:");  
   for (i = 0; i < n; i++)  
    {  
      Console.Write("{0}\t", dataset[i]);  
      if (i % 5 == 0 && i != 0) Console.Write("\n");  
     }  
  
//store stem and leaf in a 2D array  
    int[,] stem\_leaf = new int[n, 2];  
    for (i = 0; i < n; i++)  
    {  
      stem\_leaf[i, 0] = dataset[i] / 10;  
      stem\_leaf[i, 1] = dataset[i] % 10;  
    }  
  
//initialize 2D array storing numbers of occurences, and values  
   int[,] mode = new int[n, 2];  
  
   for (i = 0; i < n; i++)  
     for (int j = 0; j < 2; j++) mode[i, j] = 0;  
  
       mode[0, 0] = 1;  
//find mode  
  
   int count = 1;  
   for (i = count - 1; i < n; i++)  
   {  
     for (int j = count - 1; j < n - 1; j++)  
      {  
        if (stem\_leaf[i, 0] == stem\_leaf[j + 1, 0]) { count++; mode[i, 0]++; mode[i, 1] = stem\_leaf[i, 0]; }  
       else if (i == 0) mode[i, 1] = stem\_leaf[i, 0];  
       }  
   }  
    Console.WriteLine();  
    Console.WriteLine("Stem and Leaf Display:");  
    Console.WriteLine("Frequency\tStem\tLeaf");  
    Console.WriteLine("================================");  
    int c = 0, leaf = 0;  
    for (i = 0; i < n; i++)  
     {  
       if (mode[i, 1] != 0)  
        {  
          leaf += mode[i, 0];  
  
          Console.Write("{0,-18}", mode[i, 0]);  
          Console.Write("{0,-6}", mode[i, 1]);  
          for (int j = c; j < leaf; j++)  
            {  
              Console.Write("{0}", stem\_leaf[j, 1]);  
             }  
  
         c = leaf;  
        Console.WriteLine();  
      }  
   }  
 }  
Console.ReadLine();  
}  
  
static void bubblesort(int[] dataset, int n)  
{  
  int i, j;  
  for (i = 0; i < n; i++)  
  for (j = n - 1; j > i; j--)  
  if (dataset[j] < dataset[j - 1])  
   {  
     int temp = dataset[j];  
     dataset[j] = dataset[j - 1];  
     dataset[j - 1] = temp;  
            }  
  
      }  
  
  }  
}  
  
  
}  
  
}  
  
}

## C# structure: student records application

﻿

**Exercise:**

Write a C# program to keep records and perform statistical analysis for a class of 20 students. The information of each student contains ID, Name, Sex, quizzes Scores (2 quizzes per semester), mid-term score, final score, and total score.  
The program will prompt the user to choose the operation of records from a menu as shown below:

========================================================

                                                   MENU

========================================================

1. Add student records  
  
2. Delete student records  
  
3. Update student records  
  
4. View all student records  
  
5. Calculate an average of a selected student’s scores  
  
6. Show student who gets the max total score  
  
7. Show student who gets the min total score  
  
8. Find student by ID

9. Sort records by total scores  
  
Enter your choice:1  
  
  
Note: All students records are stored in an array of structures

**Solution:**

To make this solution simple and easy to follow, we divide this solution in to different steps:

**Step1:** Declaring a structure called student to store the records

struct student  
  
{  
public string stnumber;  
  
public string stname;  
  
public string sex;  
  
public float quizz1;  
  
public float quizz2;  
  
public float assigment;  
  
public float midterm;  
  
public float final;  
  
public float total;  
  
  
};

## C# structure: student records application

﻿

### Menu of choices

**Step2:** Defining the displaymenu() method to display the menu. The simple menu provides nine choices from 1 to 9 to work with the records.

static void displaymenu(){  
  
Console.WriteLine("=====================================================");  
  
Console.WriteLine(" MENU ");  
  
Console.WriteLine("=====================================================");  
  
Console.WriteLine(" 1.Add student records");  
Console.WriteLine(" 2.Delete student records");  
Console.WriteLine(" 3.Update student records");  
Console.WriteLine(" 4.View all student records");  
Console.WriteLine(" 5.Calculate an average of a selected student's scores");  
Console.WriteLine(" 6.Show student who get the max total score");   
Console.WriteLine(" 7.Show student who get the min total score");  
Console.WriteLine(" 8.Find a student by ID");  
Console.WriteLine(" 9.Sort students by TOTAL");  
  
  
  
}

## C# structure: student records application

﻿

### Append record to list

**Step3:** defining the add(student[] st, ref int itemcount) method to add a new record to the the array of student objects. This method takes two arguments. The first argument is the array of student objects(st) and the second argument is the number of items in the array. The two arguments are passed by references. For an array, we don't need to use the ref keyword when we want to pass it by reference. However, we need to use the ref keyword when we want to pass an argument of primitive type such as int, float, dobule,etc. When the new item is added the value itemcount variable increases by 1 that means the number of records in the list increases.

//method add/append a new record  
static void add(student[] st,ref int itemcount){  
  
Again:  
Console.WriteLine();   
Console.Write("Enter student's ID:");  
st[itemcount].stnumber=Console.ReadLine().ToString() ;

//making sure the record to be added doesn't already exist  
if(search(st,st[itemcount].stnumber,itemcount)!=-1){  
  
Console.WriteLine("This ID already exists.");  
goto Again;  
  
}  
  
  
Console.Write("Enter student's Name:");   
  
st[itemcount].stname=Console.ReadLine ().ToString();  
  
  
Console.Write("Enter student's Sex(F or M):");  
st[itemcount].sex=Console.ReadLine().ToString();  
  
  
Console.Write("Enter student's quizz1 score:");  
st[itemcount].quizz1=float.Parse(Console.ReadLine());  
  
  
Console.Write("Enter student's quizz2 score:");  
st[itemcount].quizz2=float.Parse(Console.ReadLine());  
  
  
Console.Write("Enter student's assigment score:");  
st[itemcount].assigment=float.Parse(Console.ReadLine());  
  
  
Console.Write("Enter student's mid term score:");  
st[itemcount].midterm=float.Parse(Console.ReadLine());  
  
Console.Write("Enter student's final score:");  
st[itemcount].final=float.Parse(Console.ReadLine());  
  
st[itemcount].total=st[itemcount].quizz1+st[itemcount].quizz2+st[itemcount].assigment+st[itemcount].midterm+st[itemcount].final;  
  
  
  
++itemcount; //increase the number of items by one  
  
  
  
}

## C# structure: student records application

﻿

### Show all records in list

**Step4:** Defining the viewall(student[] st, int itemcount) method to display the list of all records in the set. To display all records, we need a while loop to traverse through the array of student objects.

static void viewall(student[] st,int itemcount)  
{  
  
int i = 0;  
  
Console.WriteLine("{0,-5}{1,-20}{2,-5}{3,-5}{4,-5}{5,-5}{6,-5}{7,-5}{8}(column index)", "0", "1", "2", "3", "4", "5", "6", "7", "8");  
Console.WriteLine("{0,-5}{1,-20}{2,-5}{3,-5}{4,-5}{5,-5}{6,-5}{7,-5}{8,-5}", "ID", "NAME", "SEX", "Q1", "Q2", "As", "Mi", "Fi", "TOTAL");  
  
Console.WriteLine("=====================================================");  
  
while (i < itemcount)  
{  
  
if (st[i].stnumber !=null )  
{  
  
Console.Write("{0,-5}{1,-20}{2,-5}", st[i].stnumber, st[i].stname, st[i].sex);  
  
Console.Write("{0,-5}{1,-5}{2,-5}",st[i].quizz1,st[i].quizz2,st[i].assigment);  
  
Console.Write("{0,-5}{1,-5}{2,-5}",st[i].midterm,st[i].final,st[i].total);  
  
Console.Write("\n");  
}  
  
i = i + 1;  
  
  
  
}  
  
**}**

## C# structure: student records application

﻿

### Find record index

**Step5:** Defining the search(student[] st, int itemcount) method to search for the index of a target record. This method is useful as we need it to find the location of the target record in the array of student objects. It can help us to make sure the record does exit before we allow the record for deletion or updating. If the target element is found, the method returns the index of this element. It return -1, if the target element is not found in the array.

static int search(student[] st, string id,int itemcount){  
int found =-1;  
for (int i = 0; i < itemcount && found==-1; i++)  
{  
  
  if (st[i].stnumber == id) found=i;  
  
  else found=-1 ;  
}  
  
return found;  
  
}

## C# structure: student records application

﻿

### Delete record

**Step6:** Defining the delete(student[] st, ref int itemcount) method to delete a target record from the array of student objects. The user will be prompted to enter the id of student record that his/her want to delete. Then this id will be checked to make sure it does exist in the list. If the target record or element really exists, the deletion process can be made. The deletion process starts by checking whether the target record is the last record, beginning or middle record. If the target record is the last record in the list, we simply delete the record by supplying it to the clean(student[] st, int index) method. The last record is the record that has it index equal to itemcount subtracted by 1. If the target record stays at the beginning or in the middle of the list, we need to use a loop to allow the previous element to take over the next element. This process continue until it reaches the end of the list(itemcount-1). Then the clean() method is called to clean the last element of the list that should not exit. After the element is cleaned, the itemcount variable decreases by 1. This means that the number of elements in the list decreases.

static void delete(student[] st, ref int itemcount)  
{  
string id;  
int index;  
if (itemcount > 0)  
{  
Console.Write("Enter student's ID:");  
id = Console.ReadLine();  
index = search(st, id.ToString(),itemcount);   
  
if ((index!=-1) && (itemcount != 0))  
{  
if (index == (itemcount-1)) //delete the last record  
{  
  
clean(st, index);  
--itemcount;  
  
Console.WriteLine("The record was deleted.");  
}  
else //delete the first or middle record  
{  
for (int i = index; i < itemcount-1; i++)  
{  
st[i] = st[i + 1];  
clean(st, itemcount);  
--itemcount ;  
}  
  
}  
  
}  
else Console.WriteLine("The record doesn't exist. Check the ID and try again.");  
  
  
}  
else Console.WriteLine("No record to delete");  
}

static void clean(student[] st,int index)  
{  
st[index].stnumber = null;  
st[index].stname = null;  
st[index].sex = null;  
st[index].quizz1 = 0;  
st[index].quizz2 = 0;  
st[index].assigment = 0;  
st[index].midterm = 0;  
st[index].final = 0;  
st[index].total = 0;  
  
}

## C# structure: student records application

﻿

### Update record

**Step7:** Defining the update\_rec(struct student st[], int itemcount) method to update a specified record. The update process starts by asking the user to input the id of the record to be changed. The id value is check to make sure it really exists. If it exits the change to the target record can be made after asking the user to input the new value of the field that need change.

static void update(student[] st, int itemcount)  
{  
string id;  
int column\_index;  
Console.Write("Enter student's ID:");  
id=Console.ReadLine();  
Console.Write("Which field you want to update(1-7)?:");  
column\_index=int.Parse(Console.ReadLine());  
  
int index = search(st, id.ToString(),itemcount);  
  
if ((index != -1) && (itemcount != 0))  
{  
if (column\_index == 1)  
{  
Console.Write("Enter student's Name:");  
  
st[index].stname = Console.ReadLine().ToString();  
}  
  
else if (column\_index == 2)  
{  
Console.Write("Enter student's Sex(F or M):");  
st[index].sex = Console.ReadLine().ToString();  
}  
else if (column\_index == 3)  
{  
Console.Write("Enter student's quizz1 score:");  
st[index].quizz1 = float.Parse(Console.ReadLine());  
}  
else if (column\_index == 4)  
{  
Console.Write("Enter student's quizz2 score:");  
st[index].quizz2 = float.Parse(Console.ReadLine());  
}  
else if (column\_index == 5)  
{  
Console.Write("Enter student's assigment score:");  
st[index].assigment = float.Parse(Console.ReadLine());  
}  
else if (column\_index == 6)  
{  
Console.Write("Enter student's mid term score:");  
st[index].midterm = float.Parse(Console.ReadLine());  
}  
else if (column\_index == 7)  
{  
Console.Write("Enter student's final score:");  
st[index].final = float.Parse(Console.ReadLine());  
}  
else Console.WriteLine("Invalid column index");  
st[index].total = st[index].quizz1 + st[index].quizz2 + st[index].assigment + st[index].midterm + st[index].final;  
  
  
}  
else Console.WriteLine("The record deosn't exits.Check the ID and try again.");  
  
}

## C# structure: student records application

﻿

### Average score

**Step8:** Defining the average(student[] st, int itemcount) method to calculate the average score of a selected student. The method alo starts by asking the user to input the id of the target student. This id is checked to make sure it really exist. The average score can be calculated by dividing the sum of quizz1 score, quizz2 score, assignment score, mid-term score, and final score by 5.

static void average(student[] st, int itemcount)  
{  
string id;  
float avg=0;  
Console.Write("Enter students'ID:");  
id = Console.ReadLine();  
int index = search(st, id.ToString(),itemcount);  
if (index != -1 && itemcount>0)  
{  
st[index].total = st[index].quizz1 + st[index].quizz2 + st[index].assigment + st[index].midterm + st[index].final;  
avg = st[index].total /5;  
}  
  
Console.WriteLine("The average score is {0}.", avg);  
}

## C# structure: student records application

﻿

### Min and Max scores

**Step9:** Defining the showmax(student[] st, int itemcount) and showmin(student[] st, int itemcount) methods show about the student who gets the maximum score and the student who gets the minimum score. To find the highest total core or lowest total core, we need to compare every total score of each element.

static void showmax(student[] st, int itemcount)  
{  
float max = st[0].total;  
int index=0;  
Console.WriteLine(itemcount);  
if (itemcount >= 2)  
{  
  
for (int j = 0; j < itemcount-1; ++j)  
if (max < st[j+1].total) {  
max = st[j+1].total;  
index = j+1;  
  
}  
  
  
}  
  
else if (itemcount == 1)  
{  
index = 0;  
max = st[0].total;  
}  
  
  
else Console.WriteLine("Not record found!");  
  
if (index != -1) Console.WriteLine("The student with ID:{0} gets the highest score {1}.", st[index].stnumber, max);  
  
  
}  
\

static void showmin(student[] st, int itemcount)  
{  
  
float min = st[0].total;  
int index = 0;  
if (itemcount >= 2)  
{  
for (int j = 0; j < itemcount-1; ++j)  
if (min > st[j+1].total)  
{  
min = st[j+1].total;  
index = j+1;  
  
}  
  
  
  
}  
  
else if (itemcount == 1)  
{  
index = 0;  
min = st[0].total;  
}  
else Console.WriteLine("No record found!");  
  
if (index != -1) Console.WriteLine("The student with ID:{0} gets the lowest score {1}.", st[index].stnumber, min);  
  
  
}  
//method to find record  
static void find(student[] st, int itemcount)  
{  
string id;  
Console.Write("Enter student's ID:");  
id=Console.ReadLine();  
  
int index=search(st,id.ToString(),itemcount);  
if (index != -1)  
{  
Console.Write("{0,-5}{1,-20}{2,-5}", st[index].stnumber, st[index].stname, st[index].sex);  
  
Console.Write("{0,-5}{1,-5}{2,-5}", st[index].quizz1, st[index].quizz2, st[index].assigment);  
  
Console.Write("{0,-5}{1,-5}{2,-5}", st[index].midterm, st[index].final, st[index].total);  
Console.WriteLine();   
  
}  
else Console.WriteLine("The record deosn't exits.");  
  
}

## C# structure: student records application

﻿

### Find record in the list

**Step10:** Defining the find(student[] st, int itemcount) method to find the record in the list. This method asks the user to enter the id of the student record. Then this id is checked to make sure it really exists. If the record is found, the information of the target student will be displayed. If the record is not found the message "The record doesn't exist." will be displayed.

static void find(student[] st, int itemcount)  
{  
  string id;  
  Console.Write("Enter student's ID:");  
  id=Console.ReadLine();  
  
  int index=search(st,id.ToString(),itemcount);  
  if (index != -1)  
  {  
    Console.Write("{0,-5}{1,-20}{2,-5}", st[index].stnumber, st[index].stname, st[index].sex);  
    Console.Write("{0,-5}{1,-5}{2,-5}", st[index].quizz1, st[index].quizz2, st[index].assigment);  
    Console.Write("{0,-5}{1,-5}{2,-5}", st[index].midterm, st[index].final, st[index].total);  
    Console.WriteLine();   
  
}  
  else Console.WriteLine("The record doesn't exits.");  
  
}

## C# structure: student records application

﻿

### Sort records

**Step11:** Defining the sort(student[] st, int itemcount) function to sort the records in ascending order by total scores. For the sort technique we use bubble sort algorithm.

static void bubblesort(student[] dataset, int n)  
{  
int i, j;  
for (i = 0; i < n; i++)  
for (j = n - 1; j > i; j--)  
if (dataset[j].total < dataset[j - 1].total )  
{  
  student temp = dataset[j];  
  dataset[j] = dataset[j - 1];  
  dataset[j - 1] = temp;  
  }  
 **}**

If youwant to sort the records in descending order, you need to modify the C**#** code above to:

static void bubblesort(student[] dataset, int n)  
{  
int i, j;  
for (i = 0; i < n; i++)  
for (j = n - 1; j > i; j--)  
**if (dataset[j].total > dataset[j - 1].total )**  
{  
student temp = dataset[j];  
dataset[j] = dataset[j - 1];  
dataset[j - 1] = temp;  
}  
 **}**

## C# code of students records application

﻿

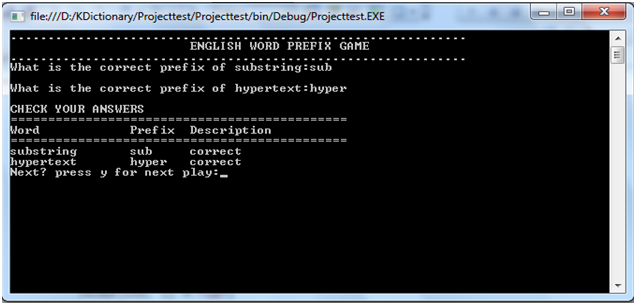
**Step12:** Now combine the C# code together and test your program.

using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
  
namespace Projecttest  
{  
class Program  
{  
//Declare student structure  
struct student  
  
{  
public string stnumber;  
  
public string stname;  
  
public string sex;  
  
public float quizz1;  
  
public float quizz2;  
  
public float assigment;  
  
public float midterm;  
  
public float final;  
  
public float total;  
  
  
};  
  
//main method  
static void Main(string[] args)  
{  
  
try  
{  
student[] st = new student[20]; //create an array to store only 20 students'records for testing.  
int itemcount = 0;  
//show menu  
displaymenu();  
int yourchoice;  
string confirm;  
  
do  
{  
  
Console.Write("Enter your choice(1-8):");  
  
yourchoice = int.Parse(Console.ReadLine());  
  
  
switch (yourchoice)  
{  
  
case 1:add(st, ref itemcount); break;  
case 2:delete(st, ref itemcount); break;  
case 3:update(st, itemcount); break;  
case 4:viewall(st, itemcount); break;  
case 5:average(st, itemcount ); break;  
case 6:showmax(st, itemcount); break;  
case 7:showmin(st, itemcount); break;  
case 8:find(st, itemcount); break;  
case 9: bubblesort (st, itemcount); break;   
  
default: Console.WriteLine("invalid"); break;  
  
}  
  
  
  
Console.Write("Press y or Y to continue:");  
  
confirm = Console.ReadLine().ToString();  
  
} while (confirm == "y" || confirm == "Y");  
  
  
  
}  
catch (FormatException f) { Console.WriteLine("Invalid input"); }  
  
  
///////////////  
  
Console.ReadLine();  
  
}  
  
  
//Menu contruction  
  
static void displaymenu(){  
  
Console.WriteLine("=====================================================");  
  
Console.WriteLine(" MENU ");  
  
Console.WriteLine("=====================================================");  
  
Console.WriteLine(" 1.Add student records");  
Console.WriteLine(" 2.Delete student records");  
Console.WriteLine(" 3.Update student records");  
Console.WriteLine(" 4.View all student records");  
Console.WriteLine(" 5.Calculate an average of a selected student's scores");  
Console.WriteLine(" 6.Show student who get the max total score");   
Console.WriteLine(" 7.Show student who get the min total score");  
Console.WriteLine(" 8.Find a student by ID");  
Console.WriteLine(" 9.Sort students by TOTAL");  
  
  
  
}  
//method add/append a new record  
static void add(student[] st,ref int itemcount){  
  
Again:  
Console.WriteLine();   
Console.Write("Enter student's ID:");  
st[itemcount].stnumber=Console.ReadLine().ToString() ;  
  
if(search(st,st[itemcount].stnumber,itemcount)!=-1){  
  
Console.WriteLine("This ID already exists.");  
goto Again;  
  
}  
  
  
Console.Write("Enter student's Name:");   
  
st[itemcount].stname=Console.ReadLine ().ToString();  
  
  
Console.Write("Enter student's Sex(F or M):");  
st[itemcount].sex=Console.ReadLine().ToString();  
  
  
Console.Write("Enter student's quizz1 score:");  
st[itemcount].quizz1=float.Parse(Console.ReadLine());  
  
  
Console.Write("Enter student's quizz2 score:");  
st[itemcount].quizz2=float.Parse(Console.ReadLine());  
  
  
Console.Write("Enter student's assigment score:");  
st[itemcount].assigment=float.Parse(Console.ReadLine());  
  
  
Console.Write("Enter student's mid term score:");  
st[itemcount].midterm=float.Parse(Console.ReadLine());  
  
Console.Write("Enter student's final score:");  
st[itemcount].final=float.Parse(Console.ReadLine());  
  
st[itemcount].total=st[itemcount].quizz1+st[itemcount].quizz2+st[itemcount].assigment+st[itemcount].midterm+st[itemcount].final;  
  
  
  
++itemcount;  
  
  
  
}  
//method to search for the location of the target record  
static int search(student[] st, string id,int itemcount){  
int found =-1;  
for (int i = 0; i < itemcount && found==-1; i++)  
{  
  
if (st[i].stnumber == id) found=i;  
  
else found=-1 ;  
}  
  
return found;  
  
}  
//method display all students' records  
static void viewall(student[] st,int itemcount)  
{  
  
int i = 0;  
  
Console.WriteLine("{0,-5}{1,-20}{2,-5}{3,-5}{4,-5}{5,-5}{6,-5}{7,-5}{8}(column index)", "0", "1", "2", "3", "4", "5", "6", "7", "8");  
Console.WriteLine("{0,-5}{1,-20}{2,-5}{3,-5}{4,-5}{5,-5}{6,-5}{7,-5}{8,-5}", "ID", "NAME", "SEX", "Q1", "Q2", "As", "Mi", "Fi", "TOTAL");  
  
Console.WriteLine("=====================================================");  
  
while (i < itemcount)  
{  
  
if (st[i].stnumber !=null )  
{  
  
Console.Write("{0,-5}{1,-20}{2,-5}", st[i].stnumber, st[i].stname, st[i].sex);  
  
Console.Write("{0,-5}{1,-5}{2,-5}",st[i].quizz1,st[i].quizz2,st[i].assigment);  
  
Console.Write("{0,-5}{1,-5}{2,-5}",st[i].midterm,st[i].final,st[i].total);  
  
Console.Write("\n");  
}  
  
i = i + 1;  
  
  
  
}  
  
}  
  
  
  
// method to elete record  
  
static void delete(student[] st, ref int itemcount)  
{  
string id;  
int index;  
if (itemcount > 0)  
{  
Console.Write("Enter student's ID:");  
id = Console.ReadLine();  
index = search(st, id.ToString(),itemcount);   
  
if ((index!=-1) && (itemcount != 0))  
{  
if (index == (itemcount-1)) //delete the last record  
{  
  
clean(st, index);  
--itemcount;  
  
Console.WriteLine("The record was deleted.");  
}  
else //delete the first or middle record  
{  
for (int i = index; i < itemcount-1; i++)  
{  
st[i] = st[i + 1];  
clean(st, itemcount);  
--itemcount ;  
}  
  
}  
  
}  
else Console.WriteLine("The record doesn't exist.Check the ID and try again.");  
  
  
}  
else Console.WriteLine("No record to delete");  
}  
  
//method ot update record  
  
static void update(student[] st, int itemcount)  
{  
string id;  
int column\_index;  
Console.Write("Enter student's ID:");  
id=Console.ReadLine();  
Console.Write("Which field you want to update(1-7)?:");  
column\_index=int.Parse(Console.ReadLine());  
  
int index = search(st, id.ToString(),itemcount);  
  
if ((index != -1) && (itemcount != 0))  
{  
if (column\_index == 1)  
{  
Console.Write("Enter student's Name:");  
  
st[index].stname = Console.ReadLine().ToString();  
}  
  
else if (column\_index == 2)  
{  
Console.Write("Enter student's Sex(F or M):");  
st[index].sex = Console.ReadLine().ToString();  
}  
else if (column\_index == 3)  
{  
Console.Write("Enter student's quizz1 score:");  
st[index].quizz1 = float.Parse(Console.ReadLine());  
}  
else if (column\_index == 4)  
{  
Console.Write("Enter student's quizz2 score:");  
st[index].quizz2 = float.Parse(Console.ReadLine());  
}  
else if (column\_index == 5)  
{  
Console.Write("Enter student's assigment score:");  
st[index].assigment = float.Parse(Console.ReadLine());  
}  
else if (column\_index == 6)  
{  
Console.Write("Enter student's mid term score:");  
st[index].midterm = float.Parse(Console.ReadLine());  
}  
else if (column\_index == 7)  
{  
Console.Write("Enter student's final score:");  
st[index].final = float.Parse(Console.ReadLine());  
}  
else Console.WriteLine("Invalid column index");  
st[index].total = st[index].quizz1 + st[index].quizz2 + st[index].assigment + st[index].midterm + st[index].final;  
  
  
}  
else Console.WriteLine("The record deosn't exits.Check the ID and try again.");  
  
}  
  
  
//method to calculate average score  
static void average(student[] st, int itemcount)  
{  
string id;  
float avg=0;  
Console.Write("Enter students'ID:");  
id = Console.ReadLine();  
int index = search(st, id.ToString(),itemcount);  
if (index != -1 && itemcount>0)  
{  
st[index].total = st[index].quizz1 + st[index].quizz2 + st[index].assigment + st[index].midterm + st[index].final;  
avg = st[index].total /5;  
}  
  
Console.WriteLine("The average score is {0}.", avg);  
}  
//method to show max total score   
static void showmax(student[] st, int itemcount)  
{  
float max = st[0].total;  
int index=0;  
Console.WriteLine(itemcount);  
if (itemcount >= 2)  
{  
  
for (int j = 0; j < itemcount-1; ++j)  
if (max < st[j+1].total) {  
max = st[j+1].total;  
index = j+1;  
  
}  
  
  
}  
  
else if (itemcount == 1)  
{  
index = 0;  
max = st[0].total;  
}  
  
  
else Console.WriteLine("Not record found!");  
  
if (index != -1) Console.WriteLine("The student with ID:{0} gets the highest score {1}.", st[index].stnumber, max);  
  
  
}  
//method to show min total score  
static void showmin(student[] st, int itemcount)  
{  
  
float min = st[0].total;  
int index = 0;  
if (itemcount >= 2)  
{  
for (int j = 0; j < itemcount-1; ++j)  
if (min > st[j+1].total)  
{  
min = st[j+1].total;  
index = j+1;  
  
}  
  
  
  
}  
  
else if (itemcount == 1)  
{  
index = 0;  
min = st[0].total;  
}  
else Console.WriteLine("No record found!");  
  
if (index != -1) Console.WriteLine("The student with ID:{0} gets the lowest score {1}.", st[index].stnumber, min);  
  
  
}  
//method to find record  
static void find(student[] st, int itemcount)  
{  
string id;  
Console.Write("Enter student's ID:");  
id=Console.ReadLine();  
  
int index=search(st,id.ToString(),itemcount);  
if (index != -1)  
{  
Console.Write("{0,-5}{1,-20}{2,-5}", st[index].stnumber, st[index].stname, st[index].sex);  
  
Console.Write("{0,-5}{1,-5}{2,-5}", st[index].quizz1, st[index].quizz2, st[index].assigment);  
  
Console.Write("{0,-5}{1,-5}{2,-5}", st[index].midterm, st[index].final, st[index].total);  
Console.WriteLine();   
  
}  
else Console.WriteLine("The record doesn't exits.");  
  
}  
  
  
//method to sort records by total score  
static void bubblesort(student[] dataset, int n)  
{  
int i, j;  
for (i = 0; i < n; i++)  
for (j = n - 1; j > i; j--)  
if (dataset[j].total < dataset[j - 1].total )  
{  
student temp = dataset[j];  
dataset[j] = dataset[j - 1];  
dataset[j - 1] = temp;  
}  
  
}  
//method to clean deleted record  
static void clean(student[] st,int index)  
{  
st[index].stnumber = null;  
st[index].stname = null;  
st[index].sex = null;  
st[index].quizz1 = 0;  
st[index].quizz2 = 0;  
st[index].assigment = 0;  
st[index].midterm = 0;  
st[index].final = 0;  
st[index].total = 0;  
  
}  
  
  
}  
}

## C# String: English Word Prefix game

﻿

**Exercise:** Write a C# program create a simple English Word Prefix game. The program starts the first two questions by asking the user to guess the answer of each question. The answers are collected and report is produced. The report should show a list of words, prefixes, and descriptions. See the sample screen shot below:



The program also allows the user to play the next words.

**Solution:**

using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
  
namespace Projecttest  
{  
class Program  
{  
  
//main method  
static void Main(string[] args)  
{  
//Declaring multi-dimensional array to store words, prefixes, and descriptions  
//Each default answer is incorrect  
//we store 4 works, 4 prefixes, and 4 default answers  
string[,] terms=new string[4,3];   
int row = 0;  
terms[row, 0] = "substring";  
terms[row, 1] = "sub";  
terms[row, 2] = "incorrent";  
row++;  
  
terms[row, 0] = "hypertext";  
terms[row, 1] = "hyper";  
terms[row, 2] = "incorrent";  
  
row++;  
  
terms[row, 0] = "antivirus";  
terms[row, 1] = "anti";  
terms[row, 2] = "incorrent";  
  
row++;  
  
terms[row, 0] = "immutable";  
terms[row, 1] = "im";  
terms[row, 2] = "incorrent";  
//play the first two words   
int numrows = 0;  
playnext(terms, numrows );  
//allow the user to play the next two words  
Console.Write("Next? press y for next play:");  
string next = Console.ReadLine();  
if (next.CompareTo("y") == 0)  
{  
  Console.Clear();//clear screen  
  numrows+=2;  
  playnext(terms, numrows);  
}  
  
  
Console.ReadLine();  
  
  
}  
  
static void playnext(string[,] terms, int rows)  
{  
 Console.WriteLine(".............................................................")  
Console.WriteLine("\t\t\tENGLISH WORD PREFIX GAME");  
Console.WriteLine(".............................................................");  
//collect answer and make comparison then update the array  
for (int i = rows; i <rows+2 ; i++)  
{  
  Console.Write("What is the correct prefix of {0}:", terms[i, 0]);  
  string ans = Console.ReadLine();  
  if (terms[i, 1].ToLower().CompareTo(ans.ToString().ToLower()) == 0)  
    terms[i, 2] = "correct";  
  Console.WriteLine();  
  
  
}

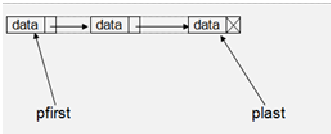
//Print report  
Console.WriteLine("CHECK YOUR ANSWERS");  
Console.WriteLine("=============================================");  
Console.WriteLine("Word\t\tPrefix\tDescription");  
Console.WriteLine("=============================================");  
  
for (int i = rows; i < rows+2; i++)  
 {  
   for (int j = 0; j < 3; j++)  
  Console.Write("{0}\t", terms[i, j]);  
  Console.WriteLine();  
}  
}  
  
}  
  
}

## C# OOP: Singly Linked List

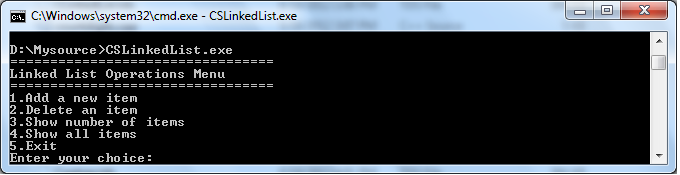
﻿

**Exercise:** By using C# objected-oriented concept, create a singly linked list.

The singly linked list is a chain of its item. Each item contain two parts--data and a link. The data part is to store information and the link part of an item is used to point or to store the address of the next item.



You are going to build a singly linked list that has two links--one(pfirst) links to the first item of the list and another one(plast) point to the last item of the list. The program also provides a menu of choices that a user can use do some operations on the linked list data structure.



**Solution:** To keep the solution simple, we divide it in to different steps as you see their links on the right menu.

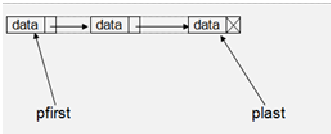
## C# OOP: Singly Linked List

﻿

### Element of singly linked list

**Step 1:** Define element of the singly linked list

The linked list element has two parts--data and a link. Therefore, we define the the element of the linked list by using a class that has two members--data and next link.



//C# code to define list element

//List element class  
class ListNode<T> //T is the generic type.  
{  
public ListNode(T elem) { val = elem;next = null; }  
public T val; //element data  
public ListNode<T> next;//next link  
}

## C# OOP: Singly Linked List

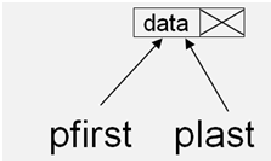
﻿

### Add element to list singly linked list

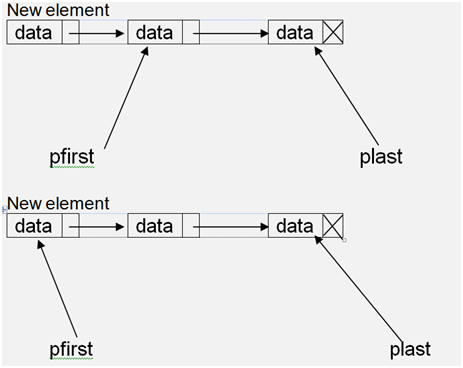
**Step 2:** Add an element to the linked list

To add an element to the linked list, you need to consider 4 things:

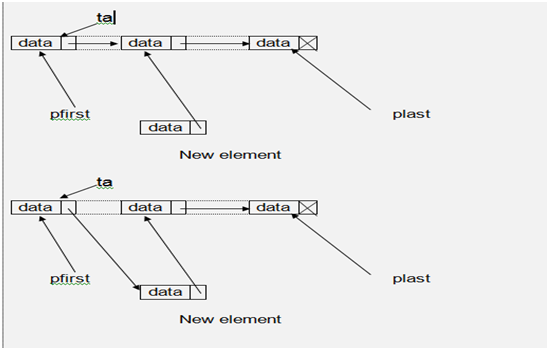
1. When the list is empty, to add a new element to the list, you only let the pfirst and plast links point to the new item. We mark the null value of link of an element by crossed lines.



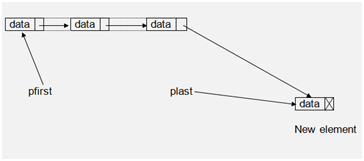
2.If the new element is to be added to the beginning of  the list, you will need to let the link of the new item points to the pfirst and then update the pfirst to point to the new item.



 3. If the new element is to be added to the middle position of the list, you need to let a link point to the position immediately before the position that the new element will be placed in.



4. If the new element is to be added to the last of the list, you need to let the link of the plast point to the new element then update the plast to point the new element.



//C# code to insert a new item to the linkedlist  
  
public void insert(T val, int pos)  
{  
ListNode<T> newnode = new ListNode<T>(val);  
//empty list  
if (pfirst == null && plast == null)  
{  
 newnode.next = null;   
 pfirst = newnode;  
 plast = newnode;  
 Console.WriteLine("Inserted:{0}", newnode.val);  
}  
//Insert at the beginning of the list  
else if (pos == 1)  
{  
 newnode.next = pfirst;  
 pfirst = newnode;  
 Console.WriteLine("Inserted:{0}", newnode.val);  
}  
//Insert in the middle of the list  
else if (pos > 1 && pos <= countitem())  
{  
ListNode<T> ta;  
ta = pfirst;  
for (int t = 1; t < pos - 1; t = t + 1) { ta = ta.next; }  
 newnode.next = ta.next;  
 ta.next = newnode;   
 Console.WriteLine("Inserted:{0}", newnode.val);  
}  
else if (pos == countitem() + 1)  
{  
 newnode.next = null; //The next link of the item is null.  
 plast.next = newnode;  
 plast = newnode;  
 Console.WriteLine("Inserted:{0}", newnode.val);  
  
}  
else Console.WriteLine("Invalid position!");  
  
  
}

## C# OOP: Singly Linked List

﻿

### Count elements of the singly linked list

**Step 3:** Count elements of the singly linkedlist

To count all elements of the linkedlist, we will need a loop to traverse through the linkedlist. We will let a variable (i) of ListElem type to point to the pfirst then move to its next element and increase the number of item(t) one at a time by using a while loop until the end of the linkedlist(indicated by null value) is reached.

//C# code count items in the linked list  
public int countitem()  
{  
ListNode<T> i;  
int t = 0;  
for (i = pfirst; i != null; i = i.next)  
{  
   t = t + 1;  
}  
return t;  
}

## C# OOP: Singly Linked List

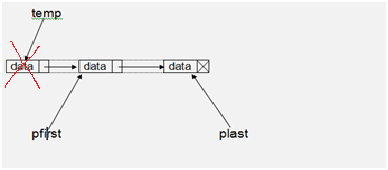
﻿

### Delete item of singly linked list

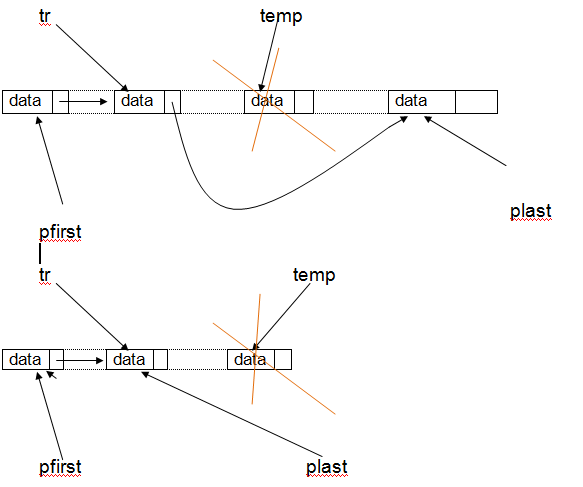
**Step 4:** Delete an element of the singly linked list

To delete an element of the linked list, you need to consider the followings:

1. If the element to be deleted is the first element of the list and the list contains only one element, you only need to assign null to the pfirst and plast. If the element to be deleted is the first element of the list and the list contain more than one element, you need a temporary variable to point to the pfirst then move the pfirst to point to its next element and set the temporary varialbe to null.



2. If the element to be deleted is in the middle of the list, you need a traversing variable(temp) to point to the element before the element to be deleted and a temporary variable(del) to point to the element to be deleted. Then let the link of the traversing pointer to point to the link of the temporary pointer. To handle situation where the element to be deleted is the last element of the list, you need to test whether the link of the temporary variable is null. If it is really null, you need to update the plast pointer to point to the traversing variable. Finally set the temporary pointer to null.



//C# code to delete an item from the linked list  
public void delete(int pos)  
{  
if (countitem() > 0)  
{ //make sure the list is not empty.  
ListNode<T> temp,del;  
  
if (pos == 1)  
{//delete the first item  
if(countitem()==1){ //The list contains only one item  
pfirst=null;  
plast=null;  
  
}  
else{ //The list contains more than one item  
temp=pfirst;  
pfirst=pfirst.next;   
temp=null;  
}  
Console.WriteLine("Deleted");  
  
}  
  
else if (pos > 1 && pos <=countitem())  
{//delete middle item  
temp=pfirst;  
int i;  
for(i=1;i<pos-1;i=i+1){temp=temp.next;} //move to the item staying before the target item to be deleted  
del=temp.next; //target item to be deleted  
temp.next=del.next;  
if(del.next==null)plast=temp; //delete last item  
del=null;  
Console.WriteLine ("Deleted");  
  
}  
  
else Console.WriteLine("Invalid position!");  
  
}  
  
else Console.WriteLine("No item found");  
  
    }  
}

## C# OOP: Singly Linked List

﻿

### Print items of singly linked list

**Step 5:** Print all elements of the linked list

To print all elements of the list is simple, you need to traverse through the list and output the data of each element.

//Print all elements  
public void showall()  
{  
  ListNode<T> t;  
  if (countitem() > 0)  
  {  
    Console.WriteLine("All items in the list:");  
    for (t = pfirst; t != null; t = t.next)  
    {  
  
         Console.WriteLine(t.val);  
    }  
  }  
else Console.WriteLine("No item found!");  
}

## C# OOP: Singly Linked List

﻿

### Menu to operate singly linked list

**Step 6:** Display a menu of choices

To show the menu that allow the user to choose an operation on the linked list, you need the showmenu() and select() functions as shown below:

 public static void showmenu(){  
  
Console.WriteLine("=================================");  
Console.WriteLine("Linked List Operations Menu");  
Console.WriteLine("=================================");  
Console.WriteLine("1.Add a new item");  
Console.WriteLine("2.Delete an item");  
Console.WriteLine("3.Show number of items");  
Console.WriteLine("4.Show all items");  
Console.WriteLine("5.Exit");  
  
}

public static void select(){  
  
LinkedList<int> mylist=new LinkedList<int>();  
int val,ch, pos;  
char yes = 'y';  
  
while (yes == 'y')  
{  
Console.Write("Enter your choice:");  
ch =int.Parse(Console.ReadLine().ToString ());  
  
switch (ch)  
{  
  
case 1:  
Console.Write("Value:");  
val = int.Parse(Console.ReadLine());  
Console.Write("Position:"); pos = int.Parse(Console.ReadLine());   
mylist.insert(val, pos);  
break;  
  
case 2:  
Console.Write("Position:"); pos = int.Parse(Console.ReadLine());  
mylist.delete(pos);  
break;  
  
case 3:  
Console.WriteLine("Number of items:" + mylist.countitem());  
break;  
  
case 4:  
Console.WriteLine("All items:");  
mylist.showall();  
break;  
  
case 5: Environment.Exit(0); break;  
  
default: Console.WriteLine("Invalid choice!"); break;  
  
}  
  
  
Console.Write("Continue? Press y to continue:");  
yes =char.Parse (Console.ReadLine());  
  
}  
  
  
  
}

## C# OOP: Singly Linked List

﻿

### C# code of singly linked list

**Step 7: Put the C# code together**

This is the complete C# code of a linked list data structure:

Top of Form

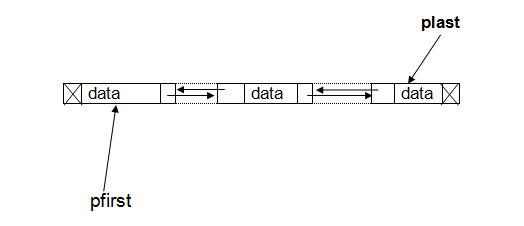


## C# OOP: doubly linked list

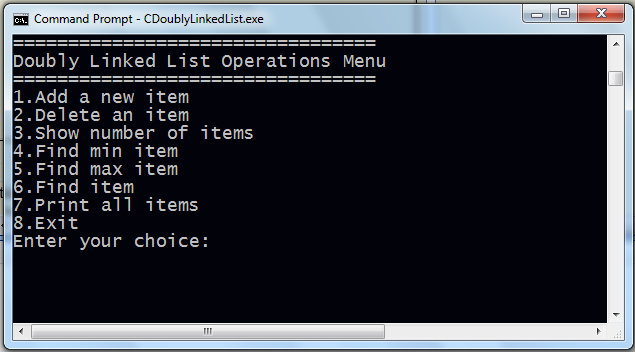
﻿

**Exercise:** create a doubly linked list in C#(Csharp) programming language by using Object-Oriented Programming(OOP) concept.

The doubly linked list is much similar to the singly linked list, except that in the doubly linked list, we add a previous link to the element of the list. By doing this the list can be traversed forward and backward.



In this solution to the exercise, we are going to build a doubly linked list that has two helpful links--one(pfirst) points to the first item of the list and another one(plast) point to the last item of the list. The program also provides a menu of choices that a user can use do some operations on the linked list.



To keep the solution simple, we divide it in to different steps as you see their links on the right menu.

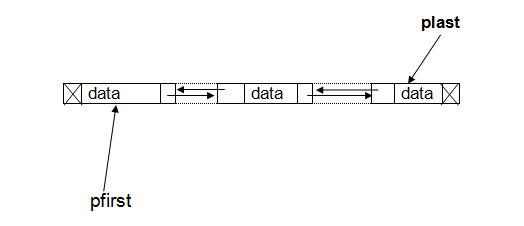
## C# OOP: doubly linked list

﻿

### Element of doubly linked list

**Step 1:** Define element and operations of the doubly linked list

The doubly linked list element has tree parts--previous pointer or link, data, and next pointer or next link. Therefore, we define the the element of the doubly linked list by using a structure that has three members--data and two pointers.



//C#(Csharp) code to define list element

class ListNode<T> //T is the generic type.  
{  
  public ListNode(T elem) { val = elem; next = null; }  
  public T val; //element data  
  public ListNode<T> prev;//previous link  
  public ListNode<T> next;//next link  
}

The operations of the doubly linked list are outlined as the methods in abstract class Cls shown below:

 abstract class Cls<T>  
{  
  public void insert(T val, int pos) { }  
  public void delete(int pos) { }  
  public void printlist() { }  
  public ListNode<T> findmin() { return null; }  
  public ListNode<T> findmax() { return null; }  
  public ListNode<T> find() { return null; }  
}

## C++ OOP: doubly linked list

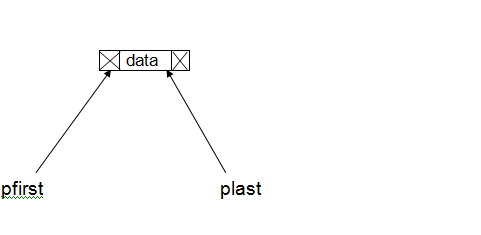
﻿

### Add new element to doubly linked list

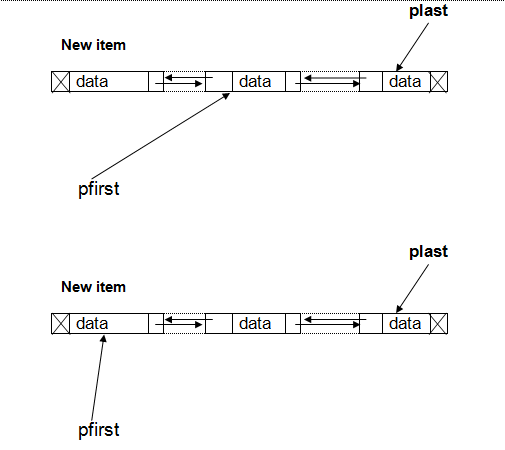
**Step 2:** Add an element to the doubly linked list

To add an element to the doubly linked list, you need to consider 4 things:

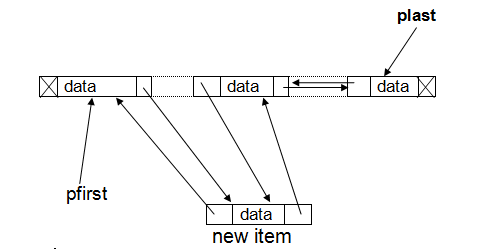
1. When the list is empty, to add a new element to the list, you only let the pfirst and plast links point to the new item.



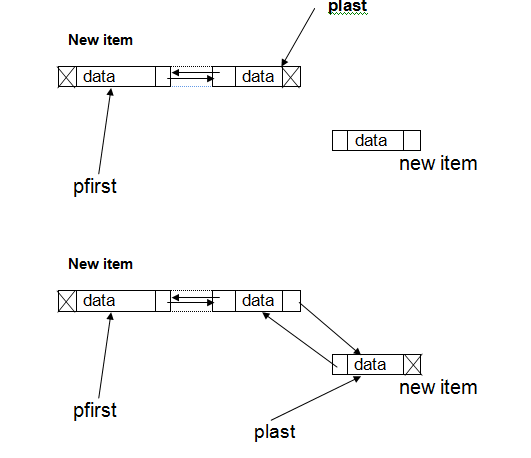
2.If the new element is to be added to the beginning of  the list, you will need to let the next link of the new item points to the pfirst, set its previous link to null, and then update the pfirst to point to the new item.



 3. If the new element is to be added to the middle position of the list, you need to let a link move to the position that the new element will be placed in. Then exchange the links as shown below:



4. If the new element is to be added to the last of the list, you need to let the next link of the plast point to the new element, and the previous link of the new element point to the plast. Then update the plast to point the new element.



//C# code to add a new element to the doubly linkedlist

public void insert(T val, int pos)

            {

                ListNode<T> newnode = new ListNode<T>(val);

                int inserted = 1;

                //empty list

                if (pfirst == null && plast == null)

                {

                    newnode.prev = null;

                    newnode.next = null;

                    pfirst = newnode;

                    plast = newnode;

                }

                //Insert at the beginning of the list

                else if (pos == 1)

                {

                    newnode.next = pfirst;

                    newnode.prev = null;

                    pfirst = newnode;

                }

                //Insert in the middle of the list

                else if (pos > 1 && pos <= countitem())

                {

                    ListNode<T> ta;

                    ta = pfirst;

                    for (int t = 1; t < pos; t = t + 1) { ta = ta.next; } //move to the insertion point

                    newnode.next = ta;

                    newnode.prev = ta.prev;

                    ta.prev.next = newnode;

                    ta.prev = newnode;

                }

                else if (pos == countitem() + 1)

                {

                    newnode.next = null;

                    plast.next = newnode;

                    newnode.prev = plast;

                    plast = newnode;

                }

                else { inserted = 0; Console.WriteLine("Invalid position!"); }

                if (inserted != 0) Console.WriteLine("Inserted") ;

            }

}

## C++ OOP: doubly linked list

﻿

### Count elements of doubly linked list

**Step 3:** Count elements of the doubly linked list

To count all elements of the doubly linkedlist, we will need a loop to traverse through the doubly linked list. We will let a pointer (i) of ListElem type to point to the pfirst then move the pointer to its next element and increase the number of item(t) one at a time by using a while loop until the end of the doubly linked list is reached.

//Count the number of items in the doubly linked list

public int countitem()

            {

                ListNode<T> i;

                int t = 0;

                for (i = pfirst; i != null; i = i.next)

                {

                    t = t + 1;

                }

                return t;

            }

## C# OOP: doubly linked list

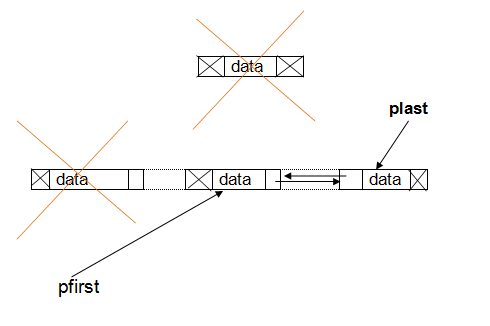
﻿

### Delete item of doubly linked list

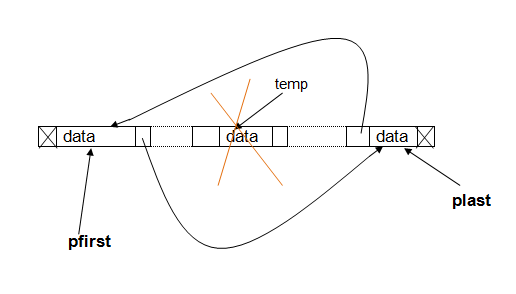
**Step 4:** Delete an element of the doubly linked list

To delete an element of the doubly linked list, you need to consider the followings:

1. If the element to be deleted is the first element of the list and the list contains only one element, you only need to assign null to the pfirst and plast. If the element to be deleted is the first element of the list and the list contain more than one element, you need a temporary link to point to the pfirst then move the pfirst to point to its next element and set the temporary pointer and the previous link of the pfirst to null.



2. If the element to be deleted is in the middle of the list, you need a traversing link(temp) to point to the element to be deleted. Then exchange the links as you see in the picture below. To handle situation where the element to be deleted is the last element of the list, you need to test whether the target item is equal to the plast. If it is really equal, you need to update the plast to point to the item staying before the target item to be deleted. Finally set the temporary pointer to null.



//C#(Csharp) code to  delete an item of the doubly linked list

public void delete(int pos)

            {

                int deleted = 1;

                int i;

                if (pfirst != null)

                { //make sure the list is not empty.

                    ListNode<T> tr, temp;

                    if (pos == 1)

                    {//delete the first item

                        if (countitem() == 1)

                        { //The list contains only one item

                            pfirst = null;

                            plast = null;

                        }

                        else

                        { //The list contains more than one item

                            tr = pfirst;

                            pfirst = pfirst.next;

                            pfirst.prev = null;

                            tr = null;

                        }

                    }

                    else if (pos > 1 && pos <= countitem())

                    {//delete middle item

                        temp = pfirst;

                        for (i = 1; i < pos; i = i + 1) { temp = temp.next; }

                        if (temp.next == null) { plast = temp.prev; plast.next = null; }

                        else

                        {

                            temp.prev.next = temp.next;

                            temp.next.prev = temp.prev;

                        }

                        temp = null;

                    }

                    else { deleted = 0; Console.WriteLine("Invalid position!"); }

                    if (deleted != 0) Console.WriteLine("Deleted");

                }

                else Console.WriteLine("No item found");

            }

## C++ OOP: doubly linked list

﻿

### Max and Min elements of doubly linked list

**Step 5:** Showing the minimum and maximum items

To find the minimum item of the doubly linked list, you need to compare each item of the list by allowing a min variable to point to the first item of the list then starting to compare its data with its next item. If the data of its next item is less than the data of the min, simply allow the min to catch the next item.

//Find min item

public ListNode<T> findmin()

       {

             ListNode<T> t, min;

             t = pfirst;

             min = t;

             while (t != null)

              {

                 if (t.val.ToString().CompareTo(min.val.ToString()) < 0) min = t;

                 t = t.next;

                }

                return min;

            }

Finding the maximum item of the doubly linked list can be done similarly as finding the minimum item. You need to compare each item of the list by allowing a max variable to point to the first item of the list then starting to compare its data with its next item. If the data of its next item is greater than the data of the max, simply allow the max to catch the next item.

//Find max item

public ListNode<T> findmax()

            {

                ListNode<T> t, max;

                t = pfirst;

                max = t;

                while (t != null)

              {

                if (t.val.ToString().CompareTo(max.val.ToString()) > 0) max = t;

                t = t.next;

                }

                return max;

            }

        }

## C# OOP: doubly linked list

﻿

### Search item of doubly linked list

**Step 6:** Searching for an item in the doubly linked list

Another important operation of the doubly linked list is searching for an item. To search for a specific item in the doubly linked list is a sequential process. The comparison starts from the beginning of the list until the target item is found or until the end of the list is reached. Therefore, the firs matched item is returned.

//Searching for an item in the doubly linked list

public ListNode<T> find(T tar)

            {

                ListNode<T> t;

                t = pfirst;

                bool f = false;

                while (t != null)

                {

                    if (t.val.ToString().CompareTo(tar.ToString()) == 0) { f = true; break; }

                    t = t.next;

                }

                if (f != false) return t;

                else return null;

            }

## C++ OOP: doubly linked list

﻿

### Print doubly linked list

**Step 7:** Print all elements of the doubly linked list

To print all elements of the list is simple, you need to traverse through the list and output the data of each element.

//Print all items of the doubly Linked List

public void printlist()

            {

                ListNode<T> t;

                if (pfirst!=null)

                {

                    Console.WriteLine("All items in the list:");

                    for (t = pfirst; t != null; t = t.next)

                    {

                        Console.WriteLine(t.val);

                    }

                }

                else Console.WriteLine("No item found!");

            }

## C++ OOP: doubly linked list

﻿

### Doubly linked list menu

**Step 8:** Display a menu of choices

To show the menu that allow the user to choose an operation on the doubly linked list, you need the showmenu() and select() functions as shown below:

    public static void select()

        {

            DoublyLinkedList<int> mylist = new DoublyLinkedList<int>();

            ListNode<int> temp;

            int val, ch, pos;

            char yes = 'y';

            //display menu

            showmenu();

            while (yes == 'y')

            {

                Console.Write("Enter your choice:");

                ch = int.Parse(Console.ReadLine().ToString());

                switch (ch)

                {

                    case 1:

                        Console.Write("Value:");

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

                        Console.Write("Position:");

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

                        mylist.insert(val, pos);

                        break;

                    case 2:

                        Console.Write("Position:");

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

                        mylist.delete(pos);

                        break;

                    case 3:

                        Console.WriteLine("Number of items:" + mylist.countitem());

                        break;

                    case 4:

                        if (mylist.findmax() != null && mylist.findmin() != null)

                            Console.WriteLine("Min item:{0}\nMax item:{1}", mylist.findmin().val, mylist.findmax().val);

                        break;

                    case 5: Console.Write("Find what?");

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

                        temp = mylist.find(val);

                        if (temp != null) Console.WriteLine("Found {0}", temp.val);

                        else Console.WriteLine("Not found"); break;

                    case 6:

                        Console.WriteLine("All items:");

                        mylist.printlist();

                        break;

                    case 7: Environment.Exit(0); break;

                    default: Console.WriteLine("Invalid choice!"); break;

                }

                Console.Write("Continue? Press y to continue:");

                yes = char.Parse(Console.ReadLine());

            }

        }

        public static void showmenu()

        {

            Console.WriteLine("======================================");

            Console.WriteLine("Doubly LinkedList Operations Menu");

            Console.WriteLine("======================================");

            Console.WriteLine("1.Add a new item");

            Console.WriteLine("2.Delete an item");

            Console.WriteLine("3.Show number of items");

            Console.WriteLine("4.Show min and max items");

            Console.WriteLine("5.Find an item");

            Console.WriteLine("6.Show all items");

            Console.WriteLine("7.Exit");

        }

## C++ OOP: doubly linked list

﻿

### Code of doubly linked list

**Step 9: Put the C#(Csharp) code together**

This is the complete C#(Csharp) code of a doubly linked list data structure:

Top of Form



Bottom of Form

Bottom of Form