Contents

[APIs 1](#_Toc10907303)

[Gender List Object API 1](#_Toc10907304)

[Stacks, Queues and Dictionaries 2](#_Toc10907305)

[Stack 2](#_Toc10907306)

[Dictionary 3](#_Toc10907307)

[Queue 5](#_Toc10907308)

[Queue with timer 6](#_Toc10907309)

[Searching Algorithms 8](#_Toc10907310)

[Sort Algorithms 10](#_Toc10907311)

[Random Sort Algorithm Class 11](#_Toc10907312)

[Basic Programming Stuff 15](#_Toc10907313)

[Object List 18](#_Toc10907314)

# APIs

## Gender List Object API

static void Main(string[] args)

{

string json = new WebClient().DownloadString("https://uinames.com/api/?ext&amount=10");

List<Person> data = new JavaScriptSerializer().Deserialize<List<Person>>(json);

List<Person> male = new List<Person>();

List<Person> female = new List<Person>();

foreach (Person x in data)

{

if (x.gender == "male")

{

male.Add(x);

}

else

{

female.Add(x);

}

}

Console.WriteLine("Male Records");

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*");

foreach (Person x in male)

{

Console.WriteLine("Title: {0}\nName: {1} {2}\nGender: {3}\nRegion: {4}\nPhone: {5}\nBirthdate\nDMY: {6}\nMDY: {7}\nRAW: {8}\nCredit Card Details\nCard Number: {9}\nExpiration {10}\nPIN: {11}\nCVC: {12}\nPhotoURL: {13}\nAge: {14}\n", x.title, x.name, x.surname, x.gender, x.region, x.phone, x.birthday.dmy, x.birthday.mdy, x.birthday.raw, x.credit\_card.number, x.credit\_card.expiration, x.credit\_card.pin, x.credit\_card.security, x.photo, x.age);

}

Console.WriteLine("Female Records");

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*");

foreach (Person x in female)

{

Console.WriteLine("Title: {0}\nName: {1} {2}\nGender: {3}\nRegion: {4}\nPhone: {5}\nBirthdate\nDMY: {6}\nMDY: {7}\nRAW: {8}\nCredit Card Details\nCard Number: {9}\nExpiration {10}\nPIN: {11}\nCVC: {12}\nPhotoURL: {13}\nAge: {14}\n", x.title, x.name, x.surname, x.gender, x.region, x.phone, x.birthday.dmy, x.birthday.mdy, x.birthday.raw, x.credit\_card.number, x.credit\_card.expiration, x.credit\_card.pin, x.credit\_card.security, x.photo, x.age);

}

Console.ReadLine();

}

}

class Person

{

public string name { get; set; }

public string surname { get; set; }

public string gender { get; set; }

public int age { get; set; }

public string region { get; set; }

public string title { get; set; }

public string phone { get; set; }

public Birthday birthday { get; set; }

public CreditCard credit\_card { get; set; }

public string email { get; set; }

public string password { get; set; }

public string photo { get; set; }

}

public class Birthday

{

public string dmy { get; set; }

public string mdy { get; set; }

public string raw { get; set; }

}

public class CreditCard

{

public string expiration { get; set; }

public string number { get; set; }

public string pin { get; set; }

public string security { get; set; }

}

# Stacks, Queues and Dictionaries

## Stack

public class Program

{

public static string palindrome = ""; //declares a variable accessible across the entire class

public static Stack palindrome1 = new Stack(); //instantiates a new stack called palindrome1

public static string palindromeout = ""; //declares a variable accessible across the entire class

public static void Main()

{

Console.WriteLine("Please enter the word you think is a palindrome");

palindrome = Console.ReadLine(); //user enters their word and it is stored as variable "palindrome"

Console.WriteLine();

string palinlower = palindrome.ToLower(); //new variable declared that stores the palindrome variable converted to lowercase

string palinlowernosp = palinlower.Replace(" ", String.Empty); //new variable declared that stores the palinlower variable with spaces removed

Console.WriteLine("Your original word, all lowercase with any spaces removed is: \n" + palinlowernosp);

Console.WriteLine();

BuildWordStack(palinlowernosp); //calls the BuildWordStack method and passes the palinlowernosp variable into it

Console.WriteLine("Your original word, all lower case with any spaces removed and reversed is: \n" + ReverseWord());

Console.WriteLine();

PalindromeCheck(palinlowernosp, ReverseWord()); //calls the PalindromeCheck method and passes the palinlowernosp variable and the ReverseWord method into it

Console.ReadLine();

}

public static void BuildWordStack(string \_palindrome)

{

char[] palinArray = \_palindrome.ToCharArray(); //create character array from the \_palindrome parameter

foreach (char ch in palinArray) //steps through each array position

{

palindrome1.Push(ch); //at each step through the array the character is push into the stack palindrome1

}

}

public static string ReverseWord()

{

int palinlength = palindrome1.Count; //the length of the stack is set to a variable before the loop begins so it is not altered as the stack objects are removed

for (int i = 0; i < palinlength; i++) //will loop through the stack until its length value is met

{

palindromeout = palindromeout + palindrome1.Pop(); //each iteration of the loop concantenates the string will the current stack object

}

return palindromeout; //returns the concantenated string which is in reverse to the original string because a stack uses last in, first out principles

}

public static void PalindromeCheck(string \_forward, string \_reverse)

{

if (\_forward == \_reverse) //this simply checks that the two parameters match, which are the original string and reversed string

{

Console.WriteLine("This is a palindrome"); //this message is displayed if the strings match

}

else

{

Console.WriteLine("This is not a palindrome"); //this message is displayed of they do not

}

}

}

## Dictionary

public class Program

{

public static void Main()

{

Console.WriteLine("Please enter a sentence to turn into an acronym");

string inputSentence = Console.ReadLine();

Acronym acronym1 = new Acronym(inputSentence); //instatiates object and passes the inputsentence variable into the constructor

acronym1.BuildAcronym(); //calls the method BuildAcronym method in the Acronym class

acronym1.DisplayAcronym(); //calls the DisplayAcronym method inside the Acronym class

Console.ReadLine();

}

}

class Acronym

{

private string fullSentence; //declares a private variable called fullSentence

private string[] words { get; set; } //declares a private string array called words and uses auto getter and setter for access

private Dictionary<char, string> acronymDictionary = new Dictionary<char, string>(); //creates an empty private Dictionary

public Acronym(string sentence) //Constructor for the Acronym class

{

fullSentence = sentence;

words = sentence.Split(' '); //splits the sentence into an array called words, divided by a space character

}

public void BuildAcronym() //method for building the acronym.

{

for (int i = 0; i < words.Length; i++) //creates a loop with the same number of iterations as words in the array

{

char keyUpper = Char.ToUpper(words[i][0]); //the words array coordinates references the current loop iteration for the word and 0 for the first letter of the word because the word string is an array by default.

if (acronymDictionary.ContainsKey(keyUpper) == false) //checks to see if a key with the same character in uppercase already exists in the dictionary

{

acronymDictionary.Add(keyUpper, words[i]); //if the above check returns false then the key is added as is

}

else

{

char keyLower = Char.ToLower(words[i][0]); //if the key already exists in uppercase then the character case is changed to lowercase

acronymDictionary.Add(keyLower, words[i]); //the character in lowercase is added to the record key

}

}

}

public void DisplayAcronym() //this method displays the acronym from the Dictionary

{

Console.WriteLine();

Console.Write("The acronym of the sentence you entered is: ");

foreach (KeyValuePair<char, string> x in acronymDictionary) //the foreach loop iterates through the dictionary records

{

Console.Write("{0}", x.Key.ToString()); //displays the key of the record in the dictionary

}

}

}

}

## Queue

public class Program

{

static string item = "Exit"; //declares and sets a default value of Exit to the item valiable

public static void Main()

{

do //this dowhile loop checks to see if the word exit was entered and if it was then the loop ends.

//If another string is entered then this string gets passed to the Enqueue method in the MyQueue class

{

Console.WriteLine("Please enter an item to add to the list or type exit to end entry");

item = Console.ReadLine();

if (item != "exit" && item != "Exit")

{

MyQueue.Enqueue(item);

}

else

{

Console.WriteLine("Entry Complete");

}

}

while (item != "exit" && item != "Exit");

Console.WriteLine("Press enter to display the queue");

Console.ReadLine();

int counter = MyQueue.queue.Count; //sets a counter to the number or items in the queue ArrayList

for (int i = 0; i < counter; i++) //iterates through the loop for the number of items in the queue ArrayList

{

Console.WriteLine(MyQueue.Dequeue()); //Displays the first item in the queue and by calling the Dequeue method in the MyQueue class, removes the first item for this iteration.

}

Console.WriteLine();

Console.WriteLine("Queue has been displayed. Press enter to end");

Console.ReadLine();

}

}

class MyQueue

{

public static ArrayList queue = new ArrayList(); //instatiates a new ArrayList called queue

public static void Enqueue(string \_item)

{

queue.Add(\_item); //adds and item to the ArrayList defined by the parameter passed into the method

}

public static string Dequeue()

{

string item = (string) queue[0]; //converts the first queue item to a string

queue.Remove(item); //removes the first item from the ListArray queue

return item; //returns the variable item to be used by the Console.WriteLine in the Main method.

}

}

## Queue with timer

public class Program

{

public static int[] timearr = new int[3]; //timer array created to hold the time values for the 3 selected customers

public static List<Person> people = new List<Person>(); //object list created to hold the customer objects with their name and wait time

public static void Main()

{

int person1; //these three variables below are used for the menu selection

int person2;

int person3;

people.Add(new Person("Tradesman Joe", 5)); //this block below adds objects to the list and instantiates them

people.Add(new Person("Dr Windy Pops", 2));

people.Add(new Person("Granpa Bob", 8));

people.Add(new Person("Billy the kid", 3));

people.Add(new Person("Chris on crutches", 6));

Console.WriteLine("Westpac queue system");

int counter = 0; //creates a variable called counter

foreach (Person x in people) //this foreach loop creates the menu list from the object list Person

{

Console.WriteLine("{0}) {1} has a processing time of {2} seconds",counter + 1, people[counter].Name, people[counter].Time);

counter++;

}

Console.WriteLine();

Console.WriteLine("Please select 3 people from the list above you want to add to the queue using thier list number");

do //this dowhile loop checks to make sure the value entered does not go above 5

{

Console.WriteLine("PLease enter the number for the first person to add");

person1 = int.Parse(Console.ReadLine());

}

while (person1 > 5);

string p1name = people[person1 - 1].Name; //this creates a name string from the people objects value Name

timearr[0] = people[person1 - 1].Time; //This adds the Time value from the object into the timearr array

MyQueue.Enqueue(p1name); //Calls the Enqueue method in the MyQueue class and passes it the name string created above

do //same as previous code block

{

Console.WriteLine("PLease enter the number for the second person to add");

person2 = int.Parse(Console.ReadLine());

}

while (person2 > 5);

string p2name = people[person2 - 1].Name;

timearr[1] = people[person2 - 1].Time;

MyQueue.Enqueue(p2name);

do //same as previous code block

{

Console.WriteLine("PLease enter the number for the third person to add");

person3 = int.Parse(Console.ReadLine());

}

while (person3 > 5);

string p3name = people[person3 - 1].Name;

timearr[2] = people[person3 - 1].Time;

MyQueue.Enqueue(p3name);

Console.WriteLine();

Console.WriteLine("Press enter to begin the queue sequence");

Console.ReadLine();

int counter2 = MyQueue.queue.Count; //creates a new counter integer set to the number of items in the queue

for (int i = 0; i < counter2; i++) //creates a for loop for the number of items as set above

{

Console.Clear();

Console.WriteLine("Bank Teller");

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*");

Console.WriteLine();

OuputQueue(); //calls the ouput queue method to display the current queue composition

MyQueue.Dequeue(); //calls the Dequeue method in the MyQueue class to remove the first record in the queue

Console.WriteLine();

Console.WriteLine("Next customer in {0} seconds", timearr[i]); //shows the number of seconds in the timearr array taken from the object value Time

System.Threading.Thread.Sleep(timearr[i] \* 1000); //sets a delay timer using the current timearr value for this iteration of the loop

}

Console.WriteLine();

Console.WriteLine("Queue complete. Press enter to end");

Console.ReadLine();

}

public static void OuputQueue()

{

foreach (string q in MyQueue.queue) //foreach loop iterates through the items in the queue in the MyQueue class

{

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

}

}

}

public class MyQueue

{

public static ArrayList queue = new ArrayList(); //creates and instantiates a new ArrayList called queue

public static void Enqueue(string \_person1) //this method recieves one parameter and is used to add items to the ArrayList queue

{

queue.Add(\_person1); //this adds an item to the queue ListArray based on the parameter passed to it

}

public static void Dequeue() //this method is used to remove the first item from the ArrayList queue

{

string item = (string)queue[0]; //converts the first queue item to a string

queue.Remove(item); //removes the first item from the ListArray queue

}

}

public class Person //created so that the persons Name and Wait Time could be stored

{

string name;

int time;

public Person(string \_name, int \_time) //this constructor sets the name and time values

{

name = \_name;

time = \_time;

}

public string Name //this method returns the persons Name

{

get { return name; }

}

public int Time //this method returns the persons Wait Time

{

get { return time; }

}

}

# Searching Algorithms

class Program

{

static int[] nums = new int[101];

static void Main(string[] args)

{

ArrayPop();

Array.Sort(nums);

//Linear();

Binary();

}

static void ArrayPop()

{

Random rand = new Random();

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

{

nums[i] = rand.Next(100, 999);

Console.WriteLine(nums[i]);

}

}

static void Linear()

{

Console.Write("I can check if a number is in the array\nPlease enter a number (100 - 999): ");

if (Searching<int>(nums, int.Parse(Console.ReadLine())))

Console.WriteLine("\nYour number was found in the collection");

else

Console.WriteLine("\nYour number wasn't found in the collection");

Console.WriteLine("Press Enter To End");

Console.ReadLine();

}

static bool Searching<T>(T[] arr, T val)

{

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

{

if (arr[i].Equals(val))

return true;

}

return false;

}

static void Binary()

{

Console.Write("\n\nPlease enter a number to search for: ");

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

/\*Search for the number and give a response\*/

if (Find(nums, input))

Console.WriteLine("\nWe have found the number " + input + " in the collection...");

else

Console.WriteLine("\nWe could not find the number " + input + " in the collection...");

Console.WriteLine("Press Enter To End");

Console.ReadLine();

}

static bool Find(int[] arr, int val)

{

/\*Create and set the ncessary variables.\*/

int upper, mid, lower;

upper = arr.Length;

lower = 0;

/\*Continue looping until the lower point is moved above the upper point, then break.\*/

while (lower <= upper)

{

/\*Set the mid point\*/

mid = lower + (upper - lower) / 2;

/\*If the mid point value is the one we are looking for return true\*/

if (arr[mid] == val)

return true;

/\*If the value is less than the mid point move the top down so we can seach in the bottom half\*/

else if (val < arr[mid])

upper = mid - 1;

/\*If the value is more than the mid point move the bottom up so we can seach in the top half\*/

else

lower = mid + 1;

}

/\*If the value isn't found return false\*/

return false;

}

}

# Sort Algorithms

class Program

{

static int[] arr = new int[1000];

static void Main(string[] args)

{

ArrayPop();

int[] arrins = arr;

int[] arrbub = arr;

Stopwatch stopWatch = new Stopwatch();

stopWatch.Start();

InsertionSort(arrins);

stopWatch.Stop();

Stopwatch stopWatch2 = new Stopwatch();

stopWatch2.Start();

BubbleSort(arrbub);

stopWatch2.Stop();

TimeSpan ts = stopWatch.Elapsed;

TimeSpan ts2 = stopWatch2.Elapsed;

DisplayBubble(arrbub);

Console.WriteLine();

Console.WriteLine("Time To Sort Using Insertion: " + ts);

Console.WriteLine("Time To Sort Using Bubble: " + ts2);

Console.ReadKey();

}

static void InsertionSort(int[] \_arr)

{

int temp, j;

for (int i = 1; i < \_arr.Length; i++)

{

j = i;

temp = \_arr[i];

while (j > 0 && \_arr[j - 1] >= temp)

{

\_arr[j] = \_arr[j - 1];

j -= 1;

}

\_arr[j] = temp;

}

Console.WriteLine();

Console.WriteLine("Sorted Data Set Using Insertion Sort");

Console.WriteLine();

foreach (int x in arr)

{

Console.Write(x);

Console.WriteLine();

}

}

static int[] BubbleSort(int[] \_arr)

{

/\*External loop makes sure we check all of array\*/

for (int i = 0; i < \_arr.Length - 1; i++)

{

/\*Internal loop goes through the elements and checks each one\*/

for (int j = 0; j < \_arr.Length - 1; j++)

{

/\*If the current element is larger than the next element, swap them\*/

if (\_arr[j] > \_arr[j + 1])

{

int temp = \_arr[j + 1];

\_arr[j + 1] = \_arr[j];

\_arr[j] = temp;

}

}

}

return \_arr;

}

static void DisplayBubble<T>(T[] \_arr)

{

foreach (T x in \_arr)

{

Console.Write(x + " ");

}

}

static void ArrayPop()

{

Random rand = new Random();

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

{

arr[i] = rand.Next(10, 99999);

//Console.WriteLine(arr[i]);

}

}

}

# Random Sort Algorithm Class

class Algorithm

{

public static int[] Numbers = new int[50]; //creates a new array with length 50

public static int[] ArrayGenerator()

{

Random rand = new Random(); //creates a local variable called rand which is defined as a random number

for (int i = 0; i < Numbers.Length; i++) //starts a loop at 0, defining the number of iterations as the length of the array called Numbers

{

Numbers[i] = rand.Next(1, 99); //each iteration adds a new random number to the array with a value between 1 and 99

}

return Numbers; //returns the Numbers array to the main program

}

public static void findMaximum(int[] \_array, int \_n)

{

int temp, j; //declares two vairables called temp and j

for (int i = 1; i < \_array.Length; i++) //starts a loop at 1, defining the number of iterations as the lengths of the array that has been passed into the method

{

j = i; //j is set to the value of i

temp = \_array[i]; //the variable temp is set to the value of the current array index according to the loop iteration

while (j > 0 && \_array[j - 1] >= temp) //this will loop as long as j is greater than 0 and the array index of j less 1 is greater than or equal to temp

{

\_array[j] = \_array[j - 1]; //if the current iteriation of the array index is equal to the index preceding it

j -= 1; //then j with be deincremented by 1

}

\_array[j] = temp; //array index j is set to temp

}

int[] result = new int[\_n]; //a new array is created to hold the maximum values requested by the user

int position = 49; //a variable called position is created to define the array index

for (int z = 0; z < \_n; z++) //a loop is initianted at 0 and will iterate for the number of times as the values requested

{

result[z] = \_array[position]; //the array called result writes the highest index position from the paramerter array to index[z] of the results array

position = position - 1; //the index of the parameter array is decreased by one to get the next highest position

}

Algorithm.Display(result); //The results array is passed into the Display method to be displayed for the user

}

public static void NumOccuranceSearch(int[] \_array, int \_value, int \_occurrence)

{

int counter = 0; //declares an integer variable called counter and sets it to 0

int result = 0; //declares an integer variable called result and sets it to 0

string output = ""; //declares a string variable called output and sets it to and empty string

for (int i = 0; i < \_array.Length; i++) //intitiates a loop begining at 0 and iterating for the length of the array that is passed into the method

{

if (\_array[i] == \_value) //this checks to see if the current iteration of the array index value is equal to the requested value from the user

{

counter++; //the counter is incremented by 1

if (counter == \_occurrence) //if the value from the previous if statement is equal to the requested value then a further check is carried out to see if the occurence value taken from user is equal to the value of the counter

{

result = i; //if the value of the counter is equal to the requested occurence value then the result has been found and the result is set to the i value

}

else

{

output = "Your number does not have more than one occurrence"; //if occurrence number never equals the counter number then there is only one instance of the number

}

}

}

if (counter == 0)

{

output = "You have chosen a number that is not in the array"; //if the counter is equal to zero then the number they have chosen is not in the array

}

else if (\_occurrence > counter) //if they request an occurrence value that is greater than the instances of the number then they will be told so

{

output = "Your number does not occur that many times in the array";

}

else if (counter > 1)

{

output = "The item " + \_value + " has occurrence " + \_occurrence + " at index: " + result; //if the requested values are within required parameters and all condition met then a result is displayed

}

Console.WriteLine(output);

}

public static string LastOccuranceSearch(int[] \_array, int \_value)

{

int counter = 0; //sets counter to 0

int result = 0; //sets result to 0

string output = ""; //sets output to empyty sting

for (int i = 0; i < \_array.Length; i++) //initiates a loop starting at 0 for the length of the array that was passed into the method

{

if (\_array[i] == \_value) //this checks to see if the array index from the current loop iteration matches the entered value

{

counter++; //if the above condition matches then the counter is increased by 1

result = i; //if the above condition matches the counter then the result variable is set to i. This can be overwritten if a new instance of the value is found

}

}

if (counter == 0)

{

output = "You have chosen a number that is not in the array"; //this will display if the entered number if not in the array

}

else

{

output = "The item " + \_value + " has its last occurrence at index: " + result; //if all conditions are met then the result is displayed

}

return output;

}

public static int[] bubbleSort(int[] \_array) //a method called bubbleSort that takes in one array and returns one array

{

for (int i = 0; i < \_array.Length - 1; i++) //sets up loop starting at 0 and iterates to 1 index value less than the length of the array

{

for (int j = 0; j < \_array.Length - 1; j++) //sets up loop starting at 0 and iterates to 1 index value less than the length of the array

{

if (\_array[j] > \_array[j + 1]) //if the array index j is greater than the next highest index then the code below will execute

{

int temp = \_array[j + 1]; //a new variable called temp is created with the value of the next highest index value of the current loop iteration index (j+1)

\_array[j + 1] = \_array[j]; //the j+1 index is now replaced with the current loop iteration index (j)

\_array[j] = temp; //the current loop iteration index is replaced with the value stored in temp

//Effectively the above code block swaps the two values of j and j+1 indexes in the array

}

}

}

return \_array; //once the loop has completed the sorted array is returned

}

public static int[] improvedBubbleSort(int[] arr) //a method called bubbleSort that takes in one array and returns one array

{

bool swap = true; //a new boolean variable is declared called swap with has a value set to true

for (int i = 0; i < arr.Length - 1; i++) //sets up loop starting at 0 and iterates to 1 index value less than the length of the array

{

swap = false; //the variable swap is set to false

for (int j = 0; j < arr.Length - 1 - i; j++) //sets up loop starting at 0 and iterates to 1 index value less than the length of the array. Each iteration decreases the i value of the outer loop by 1 to eliminate sorting the last sorted value in the array

{

if (arr[j] > arr[j + 1]) //if the array index j is greater than the next highest index then the code below will execute

{

int temp = arr[j + 1];

arr[j + 1] = arr[j]; //a new variable called temp is created with the value of the next highest index value of the current loop iteration index (j+1)

arr[j] = temp; //the current loop iteration index is replaced with the value stored in temp

swap = true; //sets the swap variable to true

}

}

if (!swap) //this will break the loop if no swaps occur on the latest iteration of the loop

break;

}

return arr; //sorted array is returned

}

public static void Display(int[] \_array) //this method takes an array as a parameter

{

foreach (int x in \_array) //this for each loop iterates though each array index

{

Console.Write(x + " "); //this writes each index value with a space after it for each pass of the loop

}

}

}

# Basic Programming Stuff

|  |
| --- |
| class Program |
|  | { |
|  | static void Main(string[] args) |
|  | { |
|  | Ex9(); |
|  | } |
|  |  |
|  | static void Ex1() |
|  | { |
|  | Console.WriteLine("Hello"); |
|  | Console.ReadLine(); |
|  | } |
|  |  |
|  | static void Ex2() |
|  | { |
|  | int num1; |
|  | int num2; |
|  | int num3; |
|  |  |
|  | Console.WriteLine("Please enter the first number to add"); |
|  | num1 = int.Parse(Console.ReadLine()); |
|  | Console.WriteLine("Please enter the second number to add"); |
|  | num2 = int.Parse(Console.ReadLine()); |
|  | num3 = num1 + num2; |
|  | Console.WriteLine("The sum of " + num1 + " + " + num2 + " = " + num3); |
|  | Console.ReadLine(); |
|  | } |
|  |  |
|  | static void Ex3() |
|  | { |
|  | int num1; |
|  | int num2; |
|  | int num3; |
|  |  |
|  | Console.WriteLine("Please enter the first number to divide"); |
|  | num1 = int.Parse(Console.ReadLine()); |
|  | Console.WriteLine("Please enter the second number to divide"); |
|  | num2 = int.Parse(Console.ReadLine()); |
|  | num3 = num1 / num2; |
|  | Console.WriteLine("The answer for " + num1 + " / " + num2 + " = " + num3); |
|  | Console.ReadLine(); |
|  |  |
|  | } |
|  |  |
|  | static void Ex4() |
|  | { |
|  | int num1; |
|  | int num2; |
|  | int num3; |
|  |  |
|  | Console.WriteLine("Please enter the first number to multiply"); |
|  | num1 = int.Parse(Console.ReadLine()); |
|  | Console.WriteLine("Please enter the second number to multipy"); |
|  | num2 = int.Parse(Console.ReadLine()); |
|  | num3 = num1 \* num2; |
|  | Console.WriteLine("The product of " + num1 + " x " + num2 + " = " + num3); |
|  | Console.ReadLine(); |
|  | } |
|  |  |
|  | static void Ex5() |
|  | { |
|  | int num1; |
|  | int i; |
|  | Console.WriteLine("Please enter a number and its multiples up to 10 will be displayed"); |
|  | num1 = int.Parse(Console.ReadLine()); |
|  | for (i = 1; i <= 10; i++) |
|  | { |
|  | Console.WriteLine(num1 \* i); |
|  | } |
|  | Console.ReadLine(); |
|  | } |
|  |  |
|  | static void Ex6() |
|  | { |
|  | double radius; |
|  | double pi = 3.14159265359; |
|  | double area; |
|  | double perimeter; |
|  |  |
|  | Console.WriteLine("Please enter the radius of the circle to calcuate area and perimeter"); |
|  | radius = double.Parse(Console.ReadLine()); |
|  | area = pi \* (radius \* radius); |
|  | perimeter = (2 \* pi) \* radius; |
|  | Console.WriteLine("The area of the circle is " + area + " and the perimeter is " + perimeter); |
|  | Console.ReadLine(); |
|  | } |
|  |  |
|  | static void Ex7() |
|  | { |
|  | double variable1; |
|  | double variable2; |
|  | double tempvariable; |
|  |  |
|  | Console.WriteLine("Please enter the first number to be stored as variable 1"); |
|  | variable1 = double.Parse(Console.ReadLine()); |
|  | Console.WriteLine("Please enter the first number to be stored as variable 2"); |
|  | variable2 = double.Parse(Console.ReadLine()); |
|  | Console.WriteLine("Press enter to swap the variable values you have entered"); |
|  | Console.ReadLine(); |
|  | tempvariable = variable1; |
|  | variable1 = variable2; |
|  | variable2 = tempvariable; |
|  | Console.WriteLine("Variable 1 is now " + variable1 + " and variable 2 is now " + variable2); |
|  | Console.ReadLine(); |
|  | } |
|  |  |
|  | static void Ex8() |
|  | { |
|  | int[] array1 = new int[1000]; |
|  | for (int i = 0; i <= 999; i++) |
|  | { |
|  | array1[i] = new Random().Next(1000, 9999); |
|  | Console.WriteLine(array1[i]); |
|  | } |
|  | Console.ReadLine(); |
|  | } |
|  |  |
|  | static void Ex9() |
|  | { |
|  | int min = 100; |
|  | int max = 999; |
|  | int size = 100; |
|  | Random random = new Random(); |
|  | int[] array1 = new int[size]; |
|  | for (int i = 0; i < array1.Length; i++) |
|  | { |
|  | int num = random.Next(min, max); |
|  | while(array1.Contains(num)) |
|  | { |
|  | num = random.Next(min, max); |
|  | } |
|  | array1[i] = num; |
|  |  |
|  | } |
|  | Array.Sort(array1); |
|  | foreach (int x in array1) |
|  | { |
|  | Console.WriteLine("{0} ", x); |
|  | } |
|  | Console.ReadLine(); |
|  | } |
|  | } |

# Object List

|  |
| --- |
| class Program |
|  | { |
|  | static void Main(string[] args) |
|  | { |
|  | var people = new List<Person>(); |
|  | for (int i = 0; i < 3; i++) |
|  | { |
|  | Console.WriteLine("Enter your first name: "); |
|  | string input1 = Console.ReadLine(); |
|  | Console.WriteLine("Enter your last name: "); |
|  | string input2 = Console.ReadLine(); |
|  | people.Add(new Person(input1, input2)); |
|  | } |
|  |  |
|  | foreach(Person x in people) |
|  | { |
|  | Console.WriteLine($"Hello, {x.Fname} {x.Lname}"); |
|  | } |
|  | Console.ReadLine(); |
|  | } |
|  | } |
|  | class Person |
|  | { |
|  | string fName; |
|  | string lName; |
|  | public string Fname |
|  | { |
|  | get => fName; set => fName = value; |
|  | } |
|  | public string Lname |
|  | { |
|  | get => lName; set => lName = value; |
|  | } |
|  | public Person(string \_Fname, string \_Lname) |
|  | { |
|  | Fname = \_Fname; |
|  | Lname = \_Lname; |
|  | } |
|  | } |