|  |
| --- |
| **Day 6 Assignment by M.Pallavi**  **31-01-2022** |

|  |
| --- |
| Program 1: Write a c# program to declare ArrayList and assign some values and find sum. |
| code: |
| using System;  using System.Collections;  namespace arraysum  {  internal class Program  {  static void Main(string[] args)  {  int sum = 0;  ArrayList data = new ArrayList();  data.Add(1);  data.Add(2);  data.Add(3);  data.Add(4);  data.Add(5);  foreach (var d in data)  {  sum = sum + (int)d;  }  Console.WriteLine(sum);    }  }  } |
| Output: |
|  |

|  |
| --- |
| **2. Research and find how the values of Array List are stored in the memory**. |
| Array List changes memory allocation as it grows.  When we specify the capacity while initializing the Array List, it allocates enough memory to store objects up to that capacity. The logical size remains 0. When it is time to expand the capacity, a new, larger array is created, and the values are copied to it. The capacity of an Array List is the number of elements the Array List can hold.  As elements are added to an Array List, the capacity is automatically increased as required through reallocation. The capacity can be decreased by calling Trim ToSize or by setting the Capacity property explicitly.  Elements in an Array List collection can be accessed using an integer index. Indexes in this collection are zero-based.  The Array List collection accepts null as a valid value, an allows duplicate elements. |

|  |
| --- |
| 3.What are the dis-advantages of ArrayList (Collections ArrayList). |
| Array List and all other collection classes like stack, queue and hash table which are present in System.  Collection namespace operate on object and hence are loosely typed.  The loosely typed nature of these collections makes them vulnerable to runtime errors.  Loosely typed collections can also cause performance overhead, because boxing and unboxing happens.  It is unnecessary boxing and unboxing happens behind the scenes every time we add and remove value types to the collection classes present in System. Collections namespace.  This can severely affect the performance, especially if your collections are large.  To solve this problem, generics introduced in dot net.  In Array List ,whenever we want to perform any arithmetic operation on object value, we need unbox the value If In case wrong data type is given to values in arraylist,it shows run time errors not compile time errors. |

|  |
| --- |
| 4.Create a simple program to declare List<int> and assign some values  and find sum |
| code: |
| using System;  using System.Collections.Generic;  namespace Listprogram  {  internal class Program  {  static void Main(string[] args)  {  int sum = 0;  List<int> data=new List<int>();    data.Add(1);  data.Add(2);  data.Add(3);  data.Add(4);  data.Add(5);  foreach (var d in data)  {  sum = sum + d;  }  Console.WriteLine(sum);      }  }  } |
| Output: |
|  |

**5. In a tabular format write the differences between Collections and generics.**

|  |  |  |
| --- | --- | --- |
|  | Collections | Generics |
| Namespace | System.Collections | System.Collections.Generics |
| Type | object | Int |
| type casting | yes | No |
| Example | ArrayList arlist = new ArrayList(); | ArrayList<String> list=new ArrayList<String>(); |

|  |
| --- |
| **6. Research and find how the values of List<T> are stored in the memory.** |
| Values are stored in either a short-term memory pool (implemented as the stack or registers) or a long-term memory pool (implemented as a garbage-collected heap). Storage is allocated depending on the known lifetime of the value. If the value is known to be short-lived then its storage is allocated on the short-term pool. If the value isnot*known* to be short-lived then it must be allocated on the long-term pool.  The 1, 2, 3 owned by the list could live forever;. Therefore the memory to store the 1, 2, 3 is allocated on the long-term pool.  "value types are always allocated on the stack".  List<int> can scale its size to any size at runtime unlike int[].It simply allocates an array of T larger than it needs. If it discovers that it guessed too small, it allocates a new, larger array and copies the old array contents to the new one.  A List<T> is just a convenient wrapper around a bunch of array copies!  If the values 1,2,3 were stored in stack, and a new item 4 is added to the list, then it wouldn't be continuous to the first three .The storage is actually an array allocated on the heap .The list allocates an array that istoo big. When you add a new item, it sticks it into unused space in the too-big array. When the array runs out of room, it allocates a new array. |

|  |
| --- |
| **7.Write example programs for implicit and explicit type casting**. |
| code: |
| using System;  namespace day6mor  {  internal class Program  {  static void Main(string[] args)  {  double d = 5673.74;  int i;  short a = 10;  int b;  b = a;  Console.WriteLine(b);  i = (int)d;  Console.WriteLine(i);  Console.ReadKey();  }  }  } |
| Output: |
|  |

**8.In a tabular format write all data types in C# and write the respective alias name.**

|  |  |
| --- | --- |
| **Alias** | **Type Name** |
| short | System.Int16 |
| sbyte | System.Sbyte |
| Int | System.Int32 |
| long | System.Int64 |
| byte | System.byte |
| ushort | System.UInt16 |
| uint | System.UInt32 |
| ulong | System.UInt64 |
|  |  |

|  |
| --- |
| Program 9:WACP to declare List<String> and add 5 values and print the values using  a. for loop  b. foreach loop  c. Lambda Expression |
| code: |
| using System;  using System.Collections;  using System.Collections.Generic;  namespace arraysum  {  internal class Program  {  static void Main(string[] args)  {  List<string> data = new List<string>();  data.Add("ram");  data.Add("lakshman");  data.Add("sita");  data.Add("rani");  data.Add("raju");  for (int i = 0; i < data.Count; i++)  {  Console.WriteLine("{0}",data[i]);  }  foreach (var d in data)  {  Console.WriteLine(d);  }  data.ForEach(p => Console.WriteLine(p));  }  }  }  Output: |

|  |
| --- |
| **Program 10: WACP to declare List<int> and read 5 values from user and find sum using**  **a. for loop**  **b. foreach loop**  **c. Lamdba Expression** |
| code: |
| using System;  using System.Collections;  using System.Collections.Generic;  namespace arraysum  {  internal class Program  {  static void Main(string[] args)  {  List<int> data = new List<int>();  int temp;  int sum1 = 0, sum2 = 0, sum3 = 0;  for (int i = 1; i <= 5; i++)  {  Console.WriteLine("enter any number");  temp = Convert.ToInt32(Console.ReadLine());  data.Add(temp);  }  for (int i = 0; i < data.Count; i++)  {  sum1 = sum1 + data[i];  }  Console.WriteLine(sum1);    foreach (var d in data)  {  sum2 = sum2 + d;  }  Console.WriteLine(sum2);  data.ForEach(d => sum3=sum3+d);  Console.WriteLine(sum3);  }  }  }  Output: |