|  |
| --- |
| Day 6 Morning Assignment    By  Anusha Bellala |

|  |
| --- |
| 1.Create a simple program to declare ArrayList and assign some values and find sum. |
| using System;  using System.Collections;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace ListConsole  {  internal class Program  {  static void Main(string[] args)  {  ArrayList data = new ArrayList();  int sum = 0;  data.Add(5);  data.Add(10);  data.Add(20);  data.Add(50);  data.Add(70);  foreach(var d in data)  {  sum = sum + (int)d;  }  Console.WriteLine("sum="+sum);  Console.ReadLine();  }  }  } |
| Output: |

|  |
| --- |
| 2.Research and find how the values of ArrayList are stored in the memory. |
| The elements of an ArrayList are stored in achunk of contiguous memory. When that memory becomes full, a larger chunk of contiguous memory has to be allocated (usually twice the size) and the existing elements are copied into this new chunk. We call this chunk the capacity of the ArrayList object. |

|  |
| --- |
| 3. What are the dis-advantages of ArrayList (Collections ArrayList). |
| From the example above, we not only inserted the string "abc" but also the number 123 in the list. So inserting different types of data in the ArrayList is allowed. Because ArrayList will treat all data inserted into it as an object type. In this way, when we use the data in the ArrayList to deal with problems, it is likely to report a type mismatch error, which means that the ArrayList is not type safe  Even if we ensure that we are very careful when inserting data, we have inserted the same type of data, but when using it, we also need to convert them to the corresponding original type for processing. This involves the operation of packing and unpacking , which will bring a lot of performance loss.  **boxing**: is to pack the data of the value type into the instance of the reference type,  such as assigning the value 123 of type int to the object object o.  int i=123; object o=(object)i;  **Unboxing:** is to extract value type from the reference data, such as assigning the value of the object object o to the variable I of type int.  object o=123; int i=(int)o; |

|  |
| --- |
| 4. Create a simple program to declare List<int> and assign some values and find sum |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Generics  {  internal class Program  {  static void Main(string[] args)  {  List<int> data=new List<int>();  int sum = 0;  data.Add(5);  data.Add(10);  data.Add(20);  data.Add(50);  data.Add(70);  foreach(var d in data)  {  sum = sum + d;  }  Console.WriteLine("sum="+sum);  Console.ReadLine();  }  }  } |
| Output: |

|  |
| --- |
| 5. In a tabular format write the differences between Collections and generics.  1. namespace  2. Each element is of what type  3. do you need type casting here  4. Example - ArrayList, List<T> |

|  |  |  |
| --- | --- | --- |
|  | Collections | Generics |
| namespace | System.Collections; | System.Collections.Generic; |
| element | Object type | Whatever we declare on <T> |
| Typecasting | yes | no |
|  | Example:  using System;  using System.Collections;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace ListConsole  {  internal class Program  {  static void Main(string[] args)  {  ArrayList data = new ArrayList();  int sum = 0;  data.Add(5);  data.Add(10);  data.Add(20);  data.Add(50);  data.Add(70);  foreach(var d in data)  {  sum = sum + (int)d;  }  Console.WriteLine("sum="+sum);  Console.ReadLine();  }  }  } | Example:  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Generics  {  internal class Program  {  static void Main(string[] args)  {  List<int> data=new List<int>();  int sum = 0;  data.Add(5);  data.Add(10);  data.Add(20);  data.Add(50);  data.Add(70);  foreach(var d in data)  {  sum = sum + d;  }  Console.WriteLine("sum="+sum);  Console.ReadLine();  }  }  } |

|  |
| --- |
| 6. Research and find how the values of List<T> are stored in the memory. |
| Lists, then, are stored in distinct chunks of memory which are linked together with pointers, which enables efficient use of memory generally and doesn't require resizing. ... Arrays, by contrast, are stored in sequential slabs of contiguous memory of fixed size, which enables efficient indexing and random access. |

|  |
| --- |
| 7. WACP to declare List<String> and add 5 values and print the values using  a. for loop  b. foreach loop  c. Lambda Expression |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace ListName  {  internal class Program  {  static void Main(string[] args)  {  List<string> data = new List<string>();  data.Add("Anu");  data.Add("Madhuri");  data.Add("Sindu");  data.Add("Prem");  data.Add("vyshu");  //print values using for loop  for(int i=0;i<data.Count;i++)  {  Console.WriteLine(data[i]);  }  //print values using foreach loop  foreach(var d in data)  {  Console.WriteLine(d);  }  //print values using lambda  data.ForEach(p=> Console.WriteLine(p));  Console.ReadLine();  }  }  } |
| Output: |

|  |
| --- |
| 8. WACP to declare List<int> and read 5 values from user and find sum using  a. for loop  b. foreach loop  c. Lamdba Expression |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace genericsanotherexample  {  internal class Program  {  static void Main(string[] args)  {  List<int> data = new List<int>();  int temp;  int sum1 = 0, sum2 = 0, sum3 = 0;  //Read 5 values from user  for (int i = 1; i <= 5; i++)  {  Console.WriteLine("Enter any value:");  temp = Convert.ToInt32(Console.ReadLine());  data.Add(temp);  }  //Find sum using for loop  for (int i = 0; i <= data.Count; i++)  {  sum1 = sum1 + data[i];  }  //Find sum using foreach loop  foreach (var d in data)  {  sum2 = sum2 + d;  }  //Find sum using lambda expression  data.ForEach(d => sum3 = sum3 + d);  Console.WriteLine(sum1);  Console.WriteLine(sum2);  Console.WriteLine(sum3);  Console.ReadLine();  }  }  } |
|  |

|  |
| --- |
| 9. In a tabular format write all data types in C# and write the respective alias name. |

|  |  |
| --- | --- |
| **Datatype** | **Alias name** |
| byte | Byte |
| ushort | UInt16 |
| uint | UInt32 |
| ulong | UInt64 |
| sbyte | SByte |
| short | Int16 |
| int | Int32 |
| long | Int64 |
| float | Single |
| double | Double |
| decimal | Decimal |
| bool | Boolean |
| char | Char |
| string | String |

|  |
| --- |
| 10. Write example programs for implicit and explicit type casting. |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace TypeCasting  {  internal class Program  {  static void Main(string[] args)  {  short a = 1;  int b = a;  long c = b;  float d = c;  double e = d;  int f = 23;  long g = f;  double h = g;    long i = 30;  float j = i;  double k = j;  float l = 10.0f;  double m = l;  Console.WriteLine(a);  Console.WriteLine(b);  Console.WriteLine(c);  Console.WriteLine(d);  Console.WriteLine(e);  Console.WriteLine(f);  Console.WriteLine(g);  Console.WriteLine(h);  Console.WriteLine(i);  Console.WriteLine(j);  Console.WriteLine(k);  Console.WriteLine(l);  Console.WriteLine(m);  Console.ReadLine();  }  }  } |
| Output: |