Arrays VS Lists in C#

اَللّٰهُمَّ اُرْزُقْنِى عِلْمًا نَافِعًا وَاسِعًا عَمِيْقًا

اَللّٰهُمَّ اُرْزُقْنِى رِزْقًا وَاسِعًا حَلَالًا طَيِّبًا
مُبَارَكًا مِنْ عِنْدِكَ

Working Example

Ahmad has become a professional programmer and he has written a low coupled program. But now his teacher wants him to rewrite the same program using classes in C# instead of parallel arrays in C++.
Make functions low coupled and highly independent as much as possible.

Example

```
#include <iostream>
using namespace std;

int main()
{
    string students[3];
    float urdu[3];
    float eng[3];
    float math[3];
    float result[3];
    for (int x = 0; x < 3; x++)
    {
       takeInput(students, urdu, eng, math, x);
    }
    total(urdu,eng,math,result,3);
    output(students,result,3);
}
```

```
void takeInput(string s[], float ur[], float eng[], float
ma[], int i)
{
    cout << "Enter Name: ";
    cin >> s[i];
    cout << "Enter Urdu Marks: ";
    cin >> ur[i];
    cout << "Enter English Marks: ";
    cin >> eng[i];
    cout << "Enter Maths Marks: ";
    cin >> ma[i];
}
```

```
void total(float ur[], float eng[], float ma[], float
t[], int len)
{
    for(int x = 0; x < len; x++)
    {
        t[x] = ur[x] + eng[x] + ma[x];
    }
}
```

```
void output(string s[], float t[], int len)
{
    for(int x = 0; x < len; x++)
    {
        cout << s[x] << ": " << t[x] << endl;
    }
}
```

Solution in C#: Step 1

Make a class.

```
class student
{
    public string sname;
    public float urduMarks;
    public float engMarks;
    public float mathsMarks;
    public float total;
}
```

Solution in C#: Step 2

Make the function to take input.

```
static student takeInput()
{
      student s = new student();
      Console.Write("Enter Name: ");
      s.sname = Console.ReadLine();
      Console.Write("Enter Urdu Marks: ");
      s.urduMarks = float.Parse(Console.ReadLine());
      Console.Write("Enter English Marks: ");
      s.engMarks = float.Parse(Console.ReadLine());
      Console.Write("Enter Maths Marks: ");
      s.mathsMarks = float.Parse(Console.ReadLine());
      return s;
}
```

Solution in C#: Step 3

Make the function to calculate the Sum.

```
static void sum(student s)
{
    s.total = s.engMarks + s.urduMarks + s.mathsMarks;
}
```

Solution in C#: Step 4

Make the function to display Output.

```
static void output(student s)
{
        Console.Write(s.sname);
        Console.Write("\t");
        Console.WriteLine(s.total);
}
```

Solution in C#: Step 5

Make the main Function.

```
static void Main(string[] args)
{
      student [] stu = new student[3];
      for (int x = 0; x < 3; x++)
      {
            stu[x] = takeInput();
            sum(stu[x]);
      }
      for (int x = 0; x < 3; x++)
      {
            output(stu[x]);
      }
      Console.Read();
}
```

Lists in C#

We can make our previous code more dynamic by storing the objects into lists instead of arrays.

Lists in C#

We can make our previous code more dynamic by storing the objects into lists instead of arrays.

It's better to use lists in C# because lists are far more easily sorted, searched through, and manipulated in C# than arrays using pre-defined functions.

Lists VS Arrays in C#

Syntax to declare an Array of class objects is:

```
student [] stu = new student[3];
```

Syntax to declare a List of class objects is:

```
List<student> stu = new List<student>();
```

Lists VS Arrays in C#

Syntax to declare an Array of class objects is:

```
student [] stu = new student[3];
```

Syntax to declare a List of class objects is:

```
List<student> stu = new List<student>();
```

Here we do not have to specify the size of the elements in the list

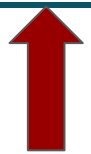
Lists VS Arrays in C#

Syntax to initialize an Array of class objects is:

```
stu[0] = takeInput();
```

Syntax to initialize a List of class objects is:

```
stu.Add(takeInput());
```



We can keep on adding the elements as follows

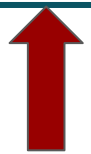
Lists VS Arrays in C#

Syntax to access an Array of class objects is:

```
stu[0];
```

Syntax to access a List of class objects is:

```
stu[0];
```



Both are Same

Lists VS Arrays in C#

Array

```
static void Main(string[] args)
{
      student [] stu = new student[3];
      for (int x = 0; x < 3; x++)
      {
            stu[x] = takeInput();
            sum(stu[x]);
      }
      for (int x = 0; x < 3; x++)
      {
            output(stu[x]);
      }
      Console.Read();
}
```

List

```
static void Main(string[] args)
{
      List<student> stu = new List<student>();
      for (int x = 0; x < 3; x++)
      {
            stu.Add(takeInput());
      }
      for (int x = 0; x < stu.Count; x++)
      {
            sum(stu[x]);
            output(stu[x]);
      }
      Console.Read();
}
```

Lists VS Arrays in C#

Array

```
static void Main(string[] args)
{
      student [] stu = new student[3];
      for (int x = 0; x < 3; x++)
      {
            stu[x] = takeInput();
            sum(stu[x]);
      }
      for (int x = 0; x < 3; x++)
      {
            output(stu[x]);
      }
      Console.Read();
}
```

List

```
static void Main(string[] args)
{
      List<student> stu = new List<student>();
      for (int x = 0; x < 3; x++)
      {
            stu.Add(takeInput());
      }
      for (int x = 0; x < stu.Count; x++)
      {
            sum(stu[x]);
            output(stu[x]);
      }
      Console.Read();
}
```

We can easily check how many elements are present in the list

Lists VS Arrays in C#

We can even insert element at a specific location in the list.

```
stu.Insert(1, takeInput());
```

This will insert a new student object at the 1st index

Lists VS Arrays in C#

We can even delete element at a specific location in the list.

```
stu.RemoveAt(1);
```

This will remove the object present on the 1st index

Working Example: SignIn SignUp Application

Make a SignIn and SignUp Application using Class that will check if the user is stored in the file then it will allow it to LogIn. If the user SignUp then the record is stored in the file in comma separated format.

Step 1: Make the class

Make a class.

```
class credential
{
    public string name;
    public string password;
}
```

Step 2: Make the Menu

```
static int menu()
        {
            int option;
            Console.WriteLine("1. SignIn");
            Console.WriteLine("2. SignUp");
            Console.WriteLine("Enter Option");
            option = int.Parse(Console.ReadLine());
            return option;
        }
```

Step 3: Read the data from the file

```
static void readData(string path, List<credential> users)
{
     if (File.Exists(path))
     {
          StreamReader fileVariable = new StreamReader(path);
          string record;
          while ((record = fileVariable.ReadLine()) != null)
          {
                credential info = new credential();
                info.name = parseData(record, 1);
                info.password = parseData(record, 2);
                users.Add(info);
          }
          fileVariable.Close();
     }
     else
     {
          Console.WriteLine("Not Exists");
     }
}
```

```
static string parseData(string record, int field)
{
     int comma = 1;
     string item = "";
     for(int x = 0; x < record.Length; x++)
     {
          if (record[x] == ',')
          {
               comma++;
          }
          else if (comma == field)
          {
               item = item + record[x];
          }
     }
     return item;
}
```

Step 4: Make the SignIn Function

```
static void signIn(string n, string p, List<credential> users)
{
      bool flag = false;
      for (int x = 0; x < users.Count; x++)
      {
          if (n == users[x].name && p == users[x].password)
          {
               Console.WriteLine("Valid User");
               flag = true;
               break;
          }
      }
      if (flag == false)
      {
          Console.WriteLine("Invalid User");
      }
      Console.ReadKey();
}
```

Step 5: Make the SignUp Function

```
static void signUp(string path, string n, string p)
        {
            StreamWriter file = new StreamWriter(path, true);
            file.WriteLine(n + "," + p);
            file.Flush();
            file.Close();
        }
```

Step 6: Main

```
static void Main(string[] args){
     List<credential> users = new List<credential>();
     string path = "G:\\OOP 2022\\BootingCSharp\\textfile.txt";
     int option;
     do{
        readData(path, users);
        Console.Clear();
        option = menu();
        Console.Clear();
        if (option == 1){
            Console.WriteLine("Enter Name: ");
            string n = Console.ReadLine();
            Console.WriteLine("Enter Password: ");
            string p = Console.ReadLine();
            signIn(n, p, users);
        }
        else if (option == 2){
            Console.WriteLine("Enter New Name: ");
            string n = Console.ReadLine();
            Console.WriteLine("Enter New Password: ");
            string p = Console.ReadLine();
            signUp(path, n, p);
         }
     }
     while (option < 3);
     Console.Read();}
```

Conclusion

Arrays	List
The size of the array should be known before we initialize its elements.	List size can be scaled up after declaration.
we can access its elements by indexes.	Like an array, we can access its elements by indexes.
Insertion and deletion are quite difficult in an array because they are stored in consecutive memory locations.	It has useful functions like Add(), Remove() RemoveAt() and sort() etc.
Arrays occupy less memory as compared to lists	Lists occupy more memory than arrays.

Learning Objective

Write a C# Program that stores objects in lists easily.

Self Assessment: Class, Objects and Lists

1. Write a program that shows three menu options
   - Add Student.
   - Show Students.
   - Top Students.

Add Student allows user to add a student's information that includes RollNo, Name, GPA, isHostelide, Department.
Show Student displays all the added students on the screen.
Top Student lists the information of the top 3 students.

Self Assessment: Class, Objects and Lists

2. Write a program that show three menu options
   - Add Products.
   - Show Products.
   - Total Store Worth.

Add Product allow user to add a product information that that includes ID, Name, price, Category, BrandName, Country.
Show Product display all the added product on the screen.
Total Store Worth calculates the sum of price of all the products.