Functions in C#
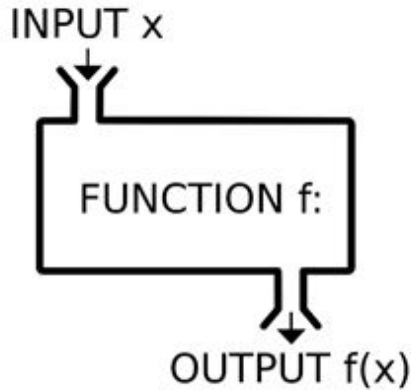
Working Example: Vision

Write a Function named addition that takes two parameters as input and then returns their sum.

INPUT x

FUNCTION f:

OUTPUT f(x)

Solution with Functions

```csharp
static int add(int n1, int n2)
{
    return n1 + n2;
}
```

```csharp
static void Main(string[] args)
{
    int num1;
    int num2;
    Console.Write("Enter 1st Number: ");
    num1 = int.Parse(Console.ReadLine());
    Console.Write("Enter 2nd Number: ");
    num2 = int.Parse(Console.ReadLine());
    int result = add(num1, num2);
    Console.WriteLine("Sum is {0}", result);
    Console.Read();
}
```

Solution with Functions

```csharp
static int add(int n1, int n2)
{
    return n1 + n2;
}
```

```csharp
static void Main(string[] args)
{
    int num1;
    int num2;
    Console.Write("Enter 1st Number: ");
    num1 = int.Parse(Console.ReadLine());
    Console.Write("Enter 2nd Number: ");
    num2 = int.Parse(Console.ReadLine());
    int result = add(num1, num2);
    Console.WriteLine("Sum is {0}", result);
    Console.Read();
}
```

Everything is same except we have to write static at the start of function definition

Solution with Functions

```
static int add(int n1, int n2)
{
    return n1 + n2;
}
```

We can write this function even after the main without explicitly giving the function prototype.

```
static void Main(string[] args)
{
    int num1;
    int num2;
    Console.Write("Enter 1st Number: ");
    num1 = int.Parse(Console.ReadLine());
    Console.Write("Enter 2nd Number: ");
    num2 = int.Parse(Console.ReadLine());
    int result = add(num1, num2);
    Console.WriteLine("Sum is {0}", result);
    Console.Read();
}
```

File Handling in C#

Working Example: Vision

Write a program that reads the data line by line from the file if the file exists.

Solution

```csharp
static void Main(string[] args)
{
    string path = "G:\\OOP 2022\\BootingCSharp\\textfile.txt";
    if (File.Exists(path))
    {
        StreamReader fileVariable = new StreamReader(path);
        string record;
        while ((record = fileVariable.ReadLine()) != null)
        {
            Console.WriteLine(record);
        }
        fileVariable.Close();
    }
    else
    {
        Console.WriteLine("Not Exists");
    }
}
```

Solution

```csharp
static void Main(string[] args)
{
    string path = "G:\\OOP 2022\\BootingCSharp\\textfile.txt";
    if (File.Exists(path))
    {
        StreamReader fileVariable = new StreamReader(path);
        string record;
        while ((record = fileVariable.ReadLine()) != null)
        {
            Console.WriteLine(record);
        }
        fileVariable.Close();
    }
    else
    {
        Console.WriteLine("Not Exists");
    }
}
```

This returns true if the file exists otherwise false

Solution

```csharp
static void Main(string[] args)
{
    string path = "G:\\OOP 2022\\BootingCSharp\\textfile.txt";
    if (File.Exists(path))
    {
        StreamReader fileVariable = new StreamReader(path);
        string record;
        while ((record = fileVariable.ReadLine()) != null)
        {
            Console.WriteLine(record);
        }
        fileVariable.Close();
    }
    else
    {
        Console.WriteLine("Not Exists");
    }
}
```

This Creates the **StreamReader** variable to read the data from the file

Solution

```csharp
static void Main(string[] args)
{
    string path = "G:\\OOP 2022\\BootingCSharp\\textfile.txt";
    if (File.Exists(path))
    {
        StreamReader fileVariable = new StreamReader(path);
        string record;
        while ((record = fileVariable.ReadLine()) != null)
        {
            Console.WriteLine(record);
        }
        fileVariable.Close();
    }
    else
    {
        Console.WriteLine("Not Exists");
    }
}
```

This Keeps on reading the data into record variable until there is no more data in the file

Working Example: Vision

Write a program that Appends the data into the file.

Solution

```csharp
static void Main(string[] args)
{
    string path = "G:\\OOP 2022\\BootingCSharp\\textfile.txt";
    StreamWriter filevariable = new StreamWriter(path, true);
    filevariable.WriteLine("hello");
    filevariable.Flush();
    filevariable.Close();
}
```

Solution

```csharp
static void Main(string[] args)
{
    string path = "G:\\OOP 2022\\BootingCSharp\\textfile.txt";
    StreamWriter filevariable = new StreamWriter(path, true);
    filevariable.WriteLine("hello");
    filevariable.Flush();
    filevariable.Close();
}
```

This Creates the StreamWriter variable to write the data into the file

Solution

```csharp
static void Main(string[] args)
{
    string path = "G:\\OOP 2022\\BootingCSharp\\textfile.txt";
    StreamWriter filevariable = new StreamWriter(path, true);
    filevariable.WriteLine("hello");
    filevariable.Flush();
    filevariable.Close();
}
```

This parameters shows that the data will be appended into the file

Solution

```
static void Main(string[] args)
{
    string path = "G:\\OOP 2022\\BootingCSharp\\textfile.txt";
    StreamWriter filevariable = new StreamWriter(path, true);
    filevariable.WriteLine("hello");
    filevariable.Flush();          ⬅
    filevariable.Close();
}
```

Clears all buffers.

Working Example: Vision

Make a SignIn and SignUp Application 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 Menu

```csharp
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 2: Read the data from the file

```csharp
static void readData(string path, string[] names, string [] password)
{
    int x = 0;
    if (File.Exists(path))
    {
        StreamReader fileVariable = new StreamReader(path);
        string record;
        while ((record = fileVariable.ReadLine()) != null)
        {
            names[x] = parseData(record, 1);
            password[x] = parseData(record, 2);
            x++;
            if (x >= 5)
            {
                break;
            }
        }
        fileVariable.Close();
    }
    else
    {
        Console.WriteLine("Not Exists");
    }
}
```

```csharp
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 3: Make the SignIn Function

```csharp
static void signIn(string n, string p, string[] names, string [] password)
        {
            bool flag = false;
            for (int x = 0; x < 5; x++)
            {
                if (n == names[x] && p == password[x])
                {
                    Console.WriteLine("Valid User");
                    flag = true;
                }
            }
            if (flag == false)
            {
                Console.WriteLine("Invalid User");
            }
            Console.ReadKey();
        }
```

Step 4: Make the SignUp Function

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

Step 5: main

```csharp
static void Main(string[] args)
{
    string path = "G:\\OOP 2022\\BootingCSharp\\textfile.txt";
    string[] names = new string[5];
    string[] password = new string[5];
    int option;
    do
    {
        readData(path, names, password);
        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, names, password);
        }
        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();
}
```

Learning Objective

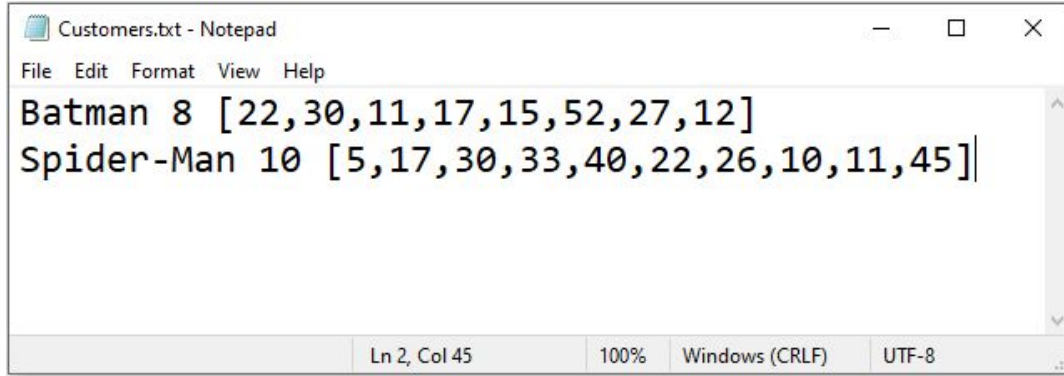Write **C#** programs that stores data permanently in files.

Self Assessment

1. **KamyabLife** is launching a network of autonomous pizza delivery drones and wants you to create a flexible rewards system (**Pizza Points**) that can be tweaked in the future. The rules are simple:

if a customer has made at least **N orders** of at least **Y price**, they get a **FREE pizza!**

The information of the customers is stored in a file **Customers.txt** in the following format. First the name of the customer is given then after the space the number of orders are given then after the space within the brackets all the orders prices are given.

Self Assessment

```
Customers.txt - Notepad
File  Edit  Format  View  Help
Batman 8 [22,30,11,17,15,52,27,12]
Spider-Man 10 [5,17,30,33,40,22,26,10,11,45]

                        Ln 2, Col 45      100%   Windows (CRLF)   UTF-8
```

Your task is to create a function that takes a minimum number of orders and a minimum order price then display the names of the customers that are eligible for a free pizza.

Self Assessment

Test Cases

Input	Output
pizza_points(5, 20)	"Spider-Man"
pizza_points(3, 10)	"Batman" "Spider-Man"
pizza_points(5, 100)	""