# File Handling

اَللّٰهُمَّ اُرْزُقْنِى عِلْمًا نَافِعًا وَاسِعًا عَمِيْقًا

اَللّٰهُمَّ اُرْزُقْنِى رِزْقًا وَاسِعًا حَلَالًا طَيِّبًا مُبَارَكًا مِنْ عِنْدِكَ

# Revision

We can conclude that every program performs **3 basic operations**.

1. Takes data as **input**
2. **Performs computations** based on the input
3. **Displays/Stores** the result

# Revision

Write a Program that takes **Distance** (in kilometers) travelled by a car in **Time** (hours) and calculates its **Speed** (kilometer/hour).

```cpp
#include <iostream>
using namespace std;
main()
{
    int distance;
    int time;
    int speed;
    cout << "Enter distance: ";
    cin >> distance;
    cout << "Enter time: ";
    cin >> time;
    speed = distance / time;
    cout << "Speed is " << speed;
}
```

# Revision

Write a Program that takes **Distance** (in kilometers) travelled by a car in **Time** (hours) and calculates its **Speed** (kilometer/hour).

**Input**

```cpp
#include <iostream>
using namespace std;
main()
{
    int distance;
    int time;
    int speed;
    cout << "Enter distance: ";
    cin >> distance;
    cout << "Enter time: ";
    cin >> time;
    speed = distance / time;
    cout << "Speed is " << speed;
}
```

# Revision

Write a Program that takes **Distance** (in kilometers) travelled by a car in **Time** (hours) and calculates its **Speed** (kilometer/hour).

**Computation** ⟶

```cpp
#include <iostream>
using namespace std;
main()
{
    int distance;
    int time;
    int speed;
    cout << "Enter distance: ";
    cin >> distance;
    cout << "Enter time: ";
    cin >> time;
    speed = distance / time;
    cout << "Speed is " << speed;
}
```

# Revision

Write a Program that takes **Distance** (in kilometers) travelled by a car in **Time** (hours) and calculates its **Speed** (kilometer/hour).

**Output**

```cpp
#include <iostream>
using namespace std;
main()
{
    int distance;
    int time;
    int speed;
    cout << "Enter distance: ";
    cin >> distance;
    cout << "Enter time: ";
    cin >> time;
    speed = distance / time;
    cout << "Speed is " << speed;
}
```
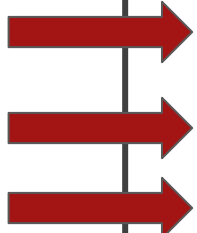
# Stream:

In programming languages, **inputs** and **outputs** are nothing but the **sequence of bytes** called the Stream.

**Input**

**Output**

```cpp
#include <iostream>
using namespace std;
main()
{
    int distance;
    int time;
    int speed;
    cout << "Enter distance: ";
    cin >> distance;
    cout << "Enter time: ";
    cin >> time;
    speed = distance / time;
    cout << "Speed is " << speed;
}
```
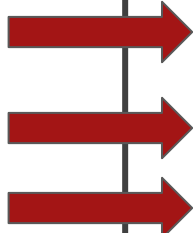
# Stream:

Stream refers to the **flow of Data**
from any device to RAM
Or
from RAM to the device

**Input**

**Output**

```cpp
#include <iostream>
using namespace std;
main()
{
    int distance;
    int time;
    int speed;
    cout << "Enter distance: ";
    cin >> distance;
    cout << "Enter time: ";
    cin >> time;
    speed = distance / time;
    cout << "Speed is " << speed;
}
```
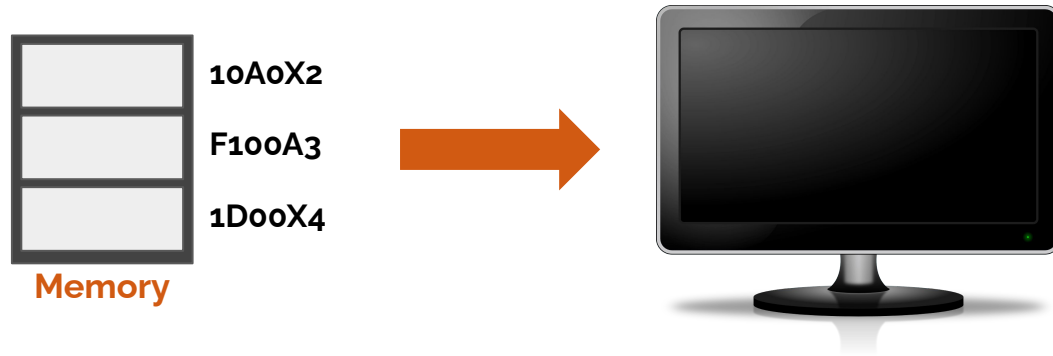
# Stream: Input Stream

The flow of data from input device to memory is called **InStream**



10A0X2

F100A3

1D00X4

**Memory**

# Stream: Output Stream

The flow of data from memory to output device is called OutStream.



**Memory**

10A0X2
F100A3
1D00X4

# Stream: Input/Output Stream

If we want to write on console or read from console, we have already used **IOSTREAM** library.

```
#include <iostream>
```

# Revision: Working Example

Write a program that takes **names of students** from the user and then **displays them on the console**.

Let's see the solution in terms of functions.

```cpp
#include <iostream>
using namespace std;
        // Function Prototype
void inputNames();
void displayNames();
        // Global Variables
string names[100];
int namesIndex = 0;

main()
{
    inputNames();
    displayNames();
}


        // Function Definition
void displayNames()
{
    cout << "Names of the Students are:" << endl;
    for (int i = 0; i < namesIndex; i = i + 1)
    {
        cout << names[i] << endl;
    }
}
```

```cpp
void inputNames()
{
    string option;
    while (true)
    {
        cout << "Enter the Name or Enter 'No' to Exit: ";
        cin >> option;
        if (option == "No" || option == "no")
        {
            break;
        }
        else
        {
            names[namesIndex] = option;
            namesIndex = namesIndex + 1;
        }
    }
}
```

# Revision: Working Example

Do you see any limitation in this program?

# Input/Output Stream: Limitation

- Inputting data in a program from the keyboard is comfortable as long as the amount of input is very small.
- Sending output to the screen works well if the amount of data is small (no larger than the size of the screen) and you do not want to distribute the output in a printed format to others.
- As soon as the program terminates, the data in the memory (RAM) is deleted.

# Input/Output Stream: Limitation

So, what is the solution?

# File Stream

- If we need to **store data permanently**, we have to store it into the **files**.
- Now, if we need to write or read from the file, we need to do it through **FileStream**.

```
#include <fstream>
```

# File Stream

- We have to **open the file** in the correct mode, before **reading** or **writing** in the file.

| Sr. No. | Open Mode | Description |
|---------|-----------|-------------|
| 1 | ios::in | Open for reading |
| 2 | ios::out | Open for writing |
| 3 | ios::app | Append mode |

# File Stream: Writing in the File

- Let's say we want to write "Welcome to UET" in a text file.

```
string line = "Welcome to UET";
```

# File Stream: Writing in the File

- Writing in a file or Storing output in the file is a five-step process

# Writing in the File: 5 Step Process

**Step 1:** Include the header file fstream in the program.

```cpp
#include <fstream>
using namespace std;
main()
{
    string line = "Welcome to UET";

}
```

# Writing in the File: 5 Step Process

**Step 2:** Declare file stream variable to open the file.

```
#include <fstream>
using namespace std;
main()
{
    string line = "Welcome to UET";
    fstream newFile;

}
```

# Writing in the File: 5 Step Process

**Step 3:** Associate the file stream variables with the text file and define the opening mode.

```cpp
#include <fstream>
using namespace std;
main()
{
    string line = "Welcome to UET";
    fstream newFile;
    newFile.open("TextFile.txt", ios::out);

}
```

# Writing in the File: 5 Step Process

**Step 3:** Associate the file stream variables with the text file and define the opening mode.

```cpp
#include <fstream>
using namespace std;
main()
{
    string line = "Welcome to UET";
    fstream newFile;
    newFile.open("TextFile.txt", ios::out);
}
```

**ios** Input Output Stream

**::** Scope resolution operator which specifies the scope

# Writing in the File: 5 Step Process

**Step 4:** Use the file stream variables insertion operator << (to store in the file).

```cpp
#include <fstream>
using namespace std;
main()
{
    string line = "Welcome to UET";
    fstream newFile;
    newFile.open("TextFile.txt", ios::out);
    newFile << line;
}
```

# Writing in the File: 5 Step Process

**Step 5:** Close the file.

```cpp
#include <fstream>
using namespace std;
main()
{
    string line = "Welcome to UET";
    fstream newFile;
    newFile.open("TextFile.txt", ios::out);
    newFile << line;
    newFile.close();
}
```

# Output: Writing in the File

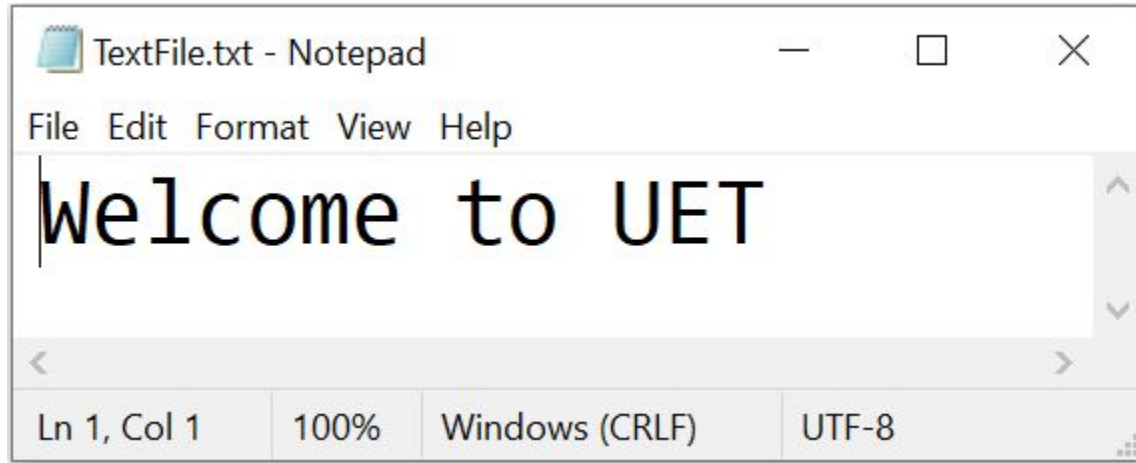- Now we will **not** see anything on the **Console**.

```
C:\C++>c++ program.cpp -o program.exe

C:\C++>program.exe

C:\C++>
```

# Output: Writing in the File

- Now we will **not** see anything on the **Console**. But the output is permanently stored in a text file.

# File in the Same Directory

Important thing to note the .txt file will be created in the same directory as that of our .exe file.

| | | | |
|---|---|---|---|
| program.cpp | 12/9/2021 8:38 PM | CPP File | 3 KB |
| program.exe | 12/9/2021 8:38 PM | Application | 50 KB |
| TextFile.txt | 1/7/2022 11:35 AM | Text Document | 1 KB |

# Activity

Write a program to input **five names** in an array and then create a file named as **names.txt** and store the values from the array in the text file.

# Activity Skeleton Code:

```cpp
#include <iostream>
#include <fstream>
using namespace std;
void inputNames();
void storeNames();
string userNames[5];
main () {
    inputNames();
    storeNames();
}
```

# Activity Solution:

```cpp
#include <iostream>
#include <fstream>
using namespace std;
void inputNames();
void storeNames();
string userNames[5];
main () {
    inputNames();
    storeNames();
}
```

```cpp
void inputNames()
{
 for(int idx=0;idx<5;idx++)
 {
   cout<<"Enter Name:";
   cin>>userNames[idx];
 }
}
```

# Activity Solution:

```cpp
#include <iostream>
#include <fstream>
using namespace std;
void inputNames();
void storeNames();
string userNames[5];
main () {
    inputNames();
    storeNames();
}
```

```cpp
void inputNames()
{
 for(int idx=0;idx<5;idx++)
 {
    cout<<"Enter Name:";
    cin>>userNames[idx];
  }
}
```

```cpp
void storeNames()
{
fstream file;
file.open("data.txt",ios::out);
for(int idx=0;idx<5;idx++)
 {
    file << userNames[idx];
    file << "\n";
  }
    file.close();
}
```

# Learning Objective

**Understand** the limitations of taking input and displaying on the **console**.

# Learning Objective

Write **C++ Program** that creates a **text file** and **store data** into the **permanent storage**.

# Conclusion

- The **limitations** of using input and output streams are:
  - **Inputting data** in a program from the keyboard is comfortable as long as the **amount of input is very small**.
  - **Sending output** to the screen works well if the amount of **data is small** and one does not want to distribute the output in a printed format to others.
  - As soon as the program terminates, the data in the **memory (RAM) is deleted**.

# Conclusion

- If we need to store data permanently, we have to store it into the **files.**
- Now, if need to write or read from the file, we need to do it through **FileStream.**

# Conclusion

- Storing output in the file is a **five-step** process:

1. Include the header file fstream in the program.
2. Declare file stream variables.
3. Associate the file stream variables with the text file and define the opening mode.
4. Use the file stream variables insertion operator << (to store output in the file i.e., write in the file)
5. Close the file.

# Self Assessment

1. Develop a Signup Application using File System.
As a user, when I SignUp to the system the
**username** and **password** stores into the file.