TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING

HIMALAYA COLLEGE OF ENGINEERING
CHYASAL, LALITPUR

Lab Report No: - File Handling
Title: -09

Submitted by: -
Name: -Aditya Man Shrestha
Roll NO: -HCE081BEI005

Submitted To: -
Department Of
Checked by: -

Date of submission: -

Objectives:

- To understand the concept of File handling in C++.

Tools and Libraries Used:

- Programming Language: C++
- IDE: Code::Blocks
- Libraries: include <iostream>, include <string>

Theory:

In C++, stream computation refers to the process of performing input and output (I/O) operations using streams. A stream is an abstraction representing a continuous flow of data between a program and external sources or destinations, such as the keyboard, console, or files. Streams provide a uniform, independent device, and object-oriented interface to handle data transfer. This abstraction hides the complexities of device-specific operations, allowing programmers to read from or write to different devices seamlessly.

Conceptually, a stream is a sequence of bytes flowing into or out of a program. Input streams transfer data into the program (for example, cin reads input from the keyboard). Output streams transfer data out of the program (for example, cout writes output to the console). Besides console I/O, streams can also be used to read from or write to files or memory buffers.

• **Standard I/O Streams:**

a. cin → standard input (keyboard)

b. cout → standard output (console)

c. cerr → standard error (unbuffered)

d. clog → standard error (buffered)

• **File I/O Streams:**

a. ifstream → input file stream (for reading from files)

b. ofstream → output file stream (for writing files)

c. fstream → input/output file stream (for both reading and writing)

Input/Output Using cin and cout.

Example:

```
int age;
cout << "Enter your age: ";
cin >> age;
cout << "You are " << age << " years old.";
```

File Handling in C++

File streams are part of the <fstream> header in C++. They allow programs to store and retrieve data from disk files, which is essential for data persistence.

• Opening a File:

Files can be opened using constructors or the open() function.

```
ofstream fileOut("data.txt");
ifstream fileIn("data.txt");
```

• Writing to a File:

```
ofstream file("example.txt");
<< "Hello, file!";
file.close();
```

• Reading from a File:

```
ifstream file("example.txt");
string line;
while(getline(file, line)) {
cout << line << endl;
}
file.close();
```

• Checking File Status:

Always verify that the file has been opened successfully using .is_open() method or by checking the stream object.

File Modes

Different modes are used to open a file:

• ios::in – open for reading

• ios::out – open for writing

• ios::app – append mode

• ios::trunc – truncate file if exists

• ios::binary – binary mode

Example:

fstream file("file.txt", ios::in | ios::out | ios::app);

Advantages of Using File Streams

• Enables persistent data storage.

• Provides a mechanism for reading and writing data in a structured way.

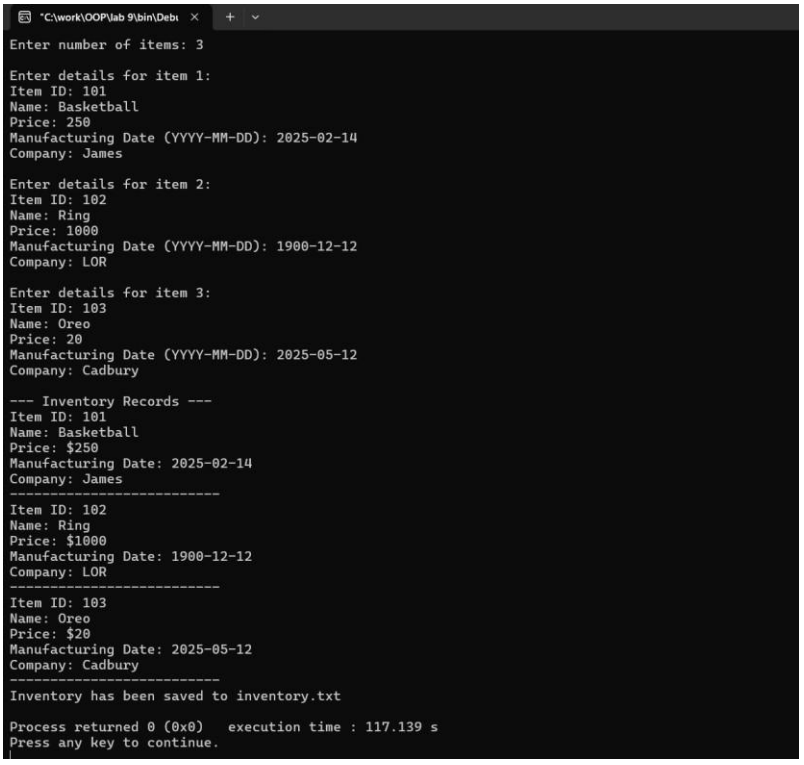
Types of Streams in C++

Lab Assignment

Qn1.

```
#include <iostream>
#include <vector>
#include <string>
#include <fstream>
using namespace std;
struct Item {
    int item_ID;
    string name;
    double price;
    string mfd_date;
    string company;
};
int main() {
    int n;
    cout << "Enter number of items: ";
    cin >> n;
    cin.ignore();
    vector<Item> inventory;
    for (int i = 0; i < n; ++i) {
        Item temp;
        cout << "\nEnter details for item " << i + 1 << ":\n";
        cout << "Item ID: ";
        cin >> temp.item_ID;
        cin.ignore();
        cout << "Name: ";
        getline(cin, temp.name);

        cout << "Price: ";
        cin >> temp.price;
        cin.ignore();

        cout << "Manufacturing Date (YYYY-MM-DD): ";
        getline(cin, temp.mfd_date);

        cout << "Company: ";
```

```
        getline(cin, temp.company);

        inventory.push_back(temp);
    }

    ofstream fout("inventory.txt");
    if (!fout) {
        cout << "Error opening file for writing.\n";
        return 1;
    }
    for (const auto& item : inventory) {
        fout << item.item_ID << "," << item.name << "," << item.price << "," <<
item.mfd_date << "," << item.company << endl;
    }
    fout.close();

    cout << "\n--- Inventory Records ---\n";
    for (const auto& item : inventory) {
        cout << "Item ID: " << item.item_ID << endl;
        cout << "Name: " << item.name << endl;
        cout << "Price: $" << item.price << endl;
        cout << "Manufacturing Date: " << item.mfd_date << endl;
        cout << "Company: " << item.company << endl;
        cout << "--------------------------\n";
    }
    cout << "Inventory has been saved to inventory.txt\n";
    return 0;
}
```



```
"C:\work\OOP\lab 9\bin\Debu
Enter number of items: 3

Enter details for item 1:
Item ID: 101
Name: Basketball
Price: 250
Manufacturing Date (YYYY-MM-DD): 2025-02-14
Company: James

Enter details for item 2:
Item ID: 102
Name: Ring
Price: 1000
Manufacturing Date (YYYY-MM-DD): 1900-12-12
Company: LOR

Enter details for item 3:
Item ID: 103
Name: Oreo
Price: 20
Manufacturing Date (YYYY-MM-DD): 2025-05-12
Company: Cadbury

--- Inventory Records ---
Item ID: 101
Name: Basketball
Price: $250
Manufacturing Date: 2025-02-14
Company: James
--------------------------
Item ID: 102
Name: Ring
Price: $1000
Manufacturing Date: 1900-12-12
Company: LOR
--------------------------
Item ID: 103
Name: Oreo
Price: $20
Manufacturing Date: 2025-05-12
Company: Cadbury
--------------------------
Inventory has been saved to inventory.txt

Process returned 0 (0x0)   execution time : 117.139 s
Press any key to continue.
```

Qn2.

```
#include <iostream>
#include <fstream>
#include <vector>
#include <string>
using namespace std;

struct Student {
    int roll;
    string name;
    int age;
    string course;
};
int main() {
    int n;
    cout << "Enter number of students: ";
    cin >> n;
    cin.ignore();

    vector<Student> students;

    for (int i = 0; i < n; ++i) {
        Student s;
        cout << "\nEnter details for student " << i + 1 << ":\n";
        cout << "Roll number: ";
        cin >> s.roll;
        cin.ignore();
        cout << "Name: ";
        getline(cin, s.name);
        cout << "Age: ";
        cin >> s.age;
        cin.ignore();
        cout << "Course: ";
        getline(cin, s.course);
        students.push_back(s);
    }

    ofstream fout("students.txt");
    if (!fout) {
        cout << "Error opening file for writing.\n";
        return 1;
    }
    for (const auto& s : students) {
        fout << s.roll << "," << s.name << "," << s.age << "," << s.course << endl;
    }
    fout.close();
```

```
    cout << "\n--- Student Details ---\n";
    for (const auto& s : students) {
        cout << "Roll: " << s.roll << endl;
        cout << "Name: " << s.name << endl;
        cout << "Age: " << s.age << endl;
        cout << "Course: " << s.course << endl;
        cout << "----------\n";
    }
    return 0;
}
```

```
Enter number of students: 2

Enter details for student 1:
Roll number: 1
Name: Aditya
Age: 19
Course: BEI

Enter details for student 2:
Roll number: 2
Name: Abir
Age: 19
Course: BEE

--- Student Details ---
Roll: 1
Name: Aditya
Age: 19
Course: BEI
----------
Roll: 2
Name: Abir
Age: 19
Course: BEE
----------

Process returned 0 (0x0)   execution time : 32.970 s
Press any key to continue.
```

Conclusion:

The lab on stream computation in C++ provided a solid understanding of how input and output operations are handled in a C++ program. We learned about the importance of cin, cout, and how file streams are used for reading from and writing to files. The use of stream classes from the and libraries make input/output operations in C++ both efficient and flexible. We also realized the importance of checking file status, handling errors, and using appropriate file modes. Mastering stream operations is crucial for developing real-world applications where data storage and retrieval are necessary.