C++ Basic Input and Output (I/O)

C++ handles input and output using streams, which are flows of data (bytes) to and from devices like the screen, keyboard, or files. This makes input/output operations faster and more organized.

Input and Output Operations

1. Output: Sending data from the program (memory) to a device like the screen or a file.
2. Input: Receiving data from a device like the keyboard or file into the program (memory).

Header Files for Input/Output

C++ provides header files that contain prewritten functions and classes for I/O tasks.

- <iostream>: Handles basic input (cin), output (cout), and error messages (cerr).
- <iomanip>: Helps in formatting input/output (e.g., setting precision, alignment).
- <fstream>: Manages file input/output (reading from or writing to files).

Key Header Files and Their Uses

1. <iostream>
   - Contains cin (read input), cout (display output), and cerr (error messages).
   - Example:

```
#include <iostream>
using namespace std;
int main() {
   int num;
   cout << "Enter a number: ";
   cin >> num;
   cout << "You entered: " << num << endl;
   return 0;
}
```

Output:

```
Enter a number: 42
You entered: 42
```

2. <iomanip>
   - Used to format data (e.g., setting decimal precision).
   - Example:

```
#include <iostream>
#include <iomanip>
using namespace std;
int main() {
   double pi = 3.14159;
   cout << fixed << setprecision(2) << "Value of pi: " << pi << endl;
   return 0;
}
```

Output:

```
Value of pi: 3.14
```

3. <fstream>
   - Handles file operations using ifstream (read from files) and ofstream (write to files).
   - Example:

```
#include <iostream>
#include <fstream>
using namespace std;
int main() {
   ofstream file("output.txt");
   if (file.is_open()) {
      file << "Hello, File I/O!";
      file.close();
      cout << "File written successfully." << endl;
   } else {
      cout << "Failed to open the file." << endl;
   }
   return 0;
}
```

Output:

```
File written successfully.
```

Special Keywords in C++ I/O

- namespace std: Simplifies coding by removing the need to write std:: before library functions like cout or cin.

```
#include <iostream>

using namespace std;

int main() {

   cout << "Hello, world!" << endl;

   return 0;

}
```

- endl: Adds a new line and flushes the output buffer.

```
cout << "Hello" << endl << "World!";
```

- cerr: Outputs error messages without delay (unbuffered).

```
cerr << "This is an error message." << endl;
```

- clog: Outputs less urgent messages (buffered).

```
clog << "This is an informational message." << endl;
```

Detailed Explanation of cin, cout, and endl :-

---

1. cout (Standard Output Stream)

- **Definition**:
  cout is a predefined object of the ostream class in C++. It is used to display output on the standard output device, usually the console.
- **How it Works**:
  cout is used with the **insertion operator** (<<) to send data from the program to the screen.
- **Syntax**:

```
cout << "Your message here";
```

- **Features**:
    - Displays text, numbers, and variables.
    - Can be combined with other output formatting like endl or manipulators from <iomanip>.
- **Example**:

```
#include <iostream>

using namespace std;

int main() {

   cout << "Welcome to C++!" << endl;

   int num = 42;

   cout << "The number is: " << num << endl;

   return 0;

}
```

Output:

```
Welcome to C++!

The number is: 42
```

---

2. cin (Standard Input Stream)

- **Definition**:
  cin is a predefined object of the istream class in C++. It is used to take input from the standard input device, usually the keyboard.
- **How it Works**:
  cin is used with the **extraction operator** (>>) to get data from the user and store it in a variable.
- **Syntax**:

```
cin >> variable_name;
```

- **Features**:
    - Takes input of different data types (e.g., int, float, string).
    - Stops reading input when it encounters a space, newline, or invalid input for the variable's type.
- **Example**:

```
#include <iostream>
```

```
using namespace std;

int main() {

    int age;

    cout << "Enter your age: ";

    cin >> age;

    cout << "Your age is: " << age << endl;

    return 0;

}
```

Output:

```
Enter your age: 25

Your age is: 25
```

---

3. endl (End Line)

- **Definition**:
  endl is a manipulator in the ostream class. It is used to insert a newline character (\n) and flush the output buffer.
- **Why Use endl**:
  - Ensures the output is displayed immediately by flushing the stream.
  - Adds a newline for better readability of the output.
- **Syntax**:

```
cout << "Line 1" << endl << "Line 2";
```

- **Features**:
  - Acts like \n for a new line but also flushes the output buffer.
  - Slower than \n when many lines are printed because of the buffer flush.
- **Example**:

```
#include <iostream>

using namespace std;

int main() {

    cout << "Hello, world!" << endl;

    cout << "C++ Programming is fun!" << endl;

    return 0;
}
```

Output:

```
Hello, world!
C++ Programming is fun!
```

- **Comparison with \n**:

```
cout << "Line 1\nLine 2";
```

Output:

```
Line 1
Line 2
```

The difference is that \n only moves to a new line but doesn't flush the buffer, making it faster in some cases.

Combined Example

Here's a program showing cin, cout, and endl together:

```
#include <iostream>
using namespace std;
int main() {
    string name;
    int age;
    // Prompt the user for their name
    cout << "Enter your name: ";
    cin >> name;
    // Prompt the user for their age
    cout << "Enter your age: ";
    cin >> age;
    // Display the output
    cout << "Hello, " << name << "!" << endl;
    cout << "You are " << age << " years old." << endl;
    return 0;
}
```

Input:

```
Enter your name: John
Enter your age: 30
```

Output:

```
Hello, John!
You are 30 years old.
```

Key Points

- cout writes output to the console using <<.
- cin reads input from the console using >>.
- endl moves to a new line and flushes the stream, ensuring the output is displayed immediately.