C++ Files and Streams

In C++ programming we are using the **iostream** standard library, it provides **cin** and **cout** methods for reading from input and writing to output respectively.

To read and write from a file we are using the standard C++ library called **fstream**. Let us see the data types define in fstream library is:

Data Type	Description
fstream	It is used to create files, write information to files, and read information from files.
ifstream	It is used to read information from files.
ofstream	It is used to create files and write information to the files.

C++ FileStream example: writing to a file

Let's see the simple example of writing to a text file **testout.txt** using C++ FileStream programming.

```
#include <iostream>
#include <fstream>
using namespace std;
int main () {
  ofstream filestream("testout.txt");
  if (filestream.is_open())
  {
    filestream << "Welcome to javaTpoint.\n";
    filestream << "C++ Tutorial.\n";
    filestream.close();
  }
  else cout <<"File opening is fail.";
  return 0;
}
```

Output:

```
The content of a text file testout.txt is set with the data:
Welcome to javaTpoint.
C++ Tutorial.
```

C++ FileStream example: reading from a file

Let's see the simple example of reading from a text file **testout.txt** using C++ FileStream programming.

```
#include <iostream>
#include <fstream>
using namespace std;
int main () {
  string srg;
  ifstream filestream("testout.txt");
  if (filestream.is_open())
  {
    while ( getline (filestream,srg) )
    {
      cout << srg <<endl;
    }
    filestream.close();
  }
  else {
      cout << "File opening is fail."<<endl;
    }
  return 0;
}
```

> Note: Before running the code a text file named as **"testout.txt"** is need to be created and the content of a text file is given below:
> Welcome to javaTpoint.
> C++ Tutorial.

Output:

```
Welcome to javaTpoint.
C++ Tutorial.
```

C++ Read and Write Example

Let's see the simple example of writing the data to a text file **testout.txt** and then reading the data from the file using C++ FileStream programming.

```
#include <fstream>
#include <iostream>
using namespace std;
int main () {
  char input[75];
  ofstream os;
  os.open("testout.txt");
  cout <<"Writing to a text file:" << endl;
  cout << "Please Enter your name: ";
  cin.getline(input, 100);
  os << input << endl;
  cout << "Please Enter your age: ";
  cin >> input;
  cin.ignore();
  os << input << endl;
  os.close();
  ifstream is;
  string line;
  is.open("testout.txt");
  cout << "Reading from a text file:" << endl;
  while (getline (is,line))
  {
  cout << line << endl;
  }
  is.close();
  return 0;
}
```

Output:

```
Writing to a text file:
 Please Enter your name: Nakul Jain
Please Enter your age: 22
 Reading from a text file:   Nakul Jain
 22
```

← Prev Next →

Feedback

- Send your Feedback to feedback@javatpoint.com

Help Others, Please Share

Learn Latest Tutorials

C++ getline()

The cin is an object which is used to take input from the user but does not allow to take the input in multiple lines. To accept the multiple lines, we use the getline() function. It is a pre-defined function defined in a **<string.h>** header file used to accept a line or a string from the input stream until the delimiting character is encountered.

Syntax of getline() function:

There are two ways of representing a function:

- The first way of declaring is to pass three parameters.

```
istream& getline( istream& is, string& str, char delim );
```

The above syntax contains three parameters, i.e., **is, str**, and **delim**.

Where,

is: It is an object of the istream class that defines from where to read the input stream.

str: It is a string object in which string is stored.

delim: It is the delimiting character.

Return value

This function returns the input stream object, which is passed as a parameter to the function.

- The second way of declaring is to pass two parameters.


```
istream& getline( istream& is, string& str );
```

The above syntax contains two parameters, i.e., **is** and **str**. This syntax is almost similar to the above syntax; the only difference is that it does not have any delimiting character.

Where,

is: It is an object of the istream class that defines from where to read the input stream.

str: It is a string object in which string is stored.

Return value

This function also returns the input stream, which is passed as a parameter to the function.

Let's understand through an example.

First, we will look at an example where we take the user input without using getline() function.


```
#include <iostream>
#include<string.h>
using namespace std;
int main()
{
string name; // variable declaration
std::cout << "Enter your name :" << std::endl;
cin>>name;
cout<<"\nHello "<<name;
return 0;
}
```

In the above code, we take the user input by using the statement **cin>>name,** i.e., we have not used the **getline()** function.

Output

```
Enter your name :
John Miller
Hello John
```

In the above output, we gave the name 'John Miller' as user input, but only 'John' was displayed. Therefore, we conclude that cin does not consider the character when the space character is encountered.

Let's resolve the above problem by using getline() function.

```
#include <iostream>
#include<string.h>
using namespace std;
int main()
{
string name; // variable declaration.
std::cout << "Enter your name :" << std::endl;
getline(cin,name); // implementing a getline() function
cout<<"\nHello "<<name;
return 0;}
```

In the above code, we have used the **getline()** function to accept the character even when the space character is encountered.

Output

```
Enter your name :
John Miller
Hello John Miller
```

In the above output, we can observe that both the words, i.e., John and Miller, are displayed, which means that the getline() function considers the character after the space character also.

When we do not want to read the character after space then we use the following code:

```
#include <iostream>
#include<string.h>
using namespace std;
int main()
{
string profile; // variable declaration
std::cout << "Enter your profile :" << std::endl;
getline(cin,profile,' '); // implementing getline() function with a delimiting character.
cout<<"\nProfile is :"<<profile;
```

```
}
```

In the above code, we take the user input by using getline() function, but this time we also add the delimiting character('') in a third parameter. Here, delimiting character is a space character, means the character that appears after space will not be considered.

Output

```
Enter your profile :
Software Developer
Profile is: Software
```

Getline Character Array

We can also define the getline() function for character array, but its syntax is different from the previous one.

Syntax

```
istream& getline(char* , int size);
```

In the above syntax, there are two parameters; one is **char***, and the other is **size**.

Where,

char*: It is a character pointer that points to the array.

Size: It acts as a delimiter that defines the size of the array means input cannot cross this size.

Let's understand through an example.

```
#include <iostream>
#include<string.h>
using namespace std;
int main()
{
char fruits[50]; // array declaration
cout<< "Enter your favorite fruit: ";
cin.getline(fruits, 50); // implementing getline() function
std::cout << "\nYour favorite fruit is :"<<fruits << std::endl;
```

```
return 0;
}
```

Output

```
Enter your favorite fruit: Watermelon
Your favorite fruit is: Watermelon
```

← Prev Next →

Feedback

- Send your Feedback to feedback@javatpoint.com

Help Others, Please Share

Learn Latest Tutorials