stringstream in C++ and its applications

A stringstream associates a string object with a stream allowing you to read from the string as if it were a stream (like cin).

Basic methods are –

*clear() — to clear the stream  
str() — to get and set string object whose content is present in stream.  
operator << — add a string to the stringstream object.  
operator >> — read something from the stringstream object,*

stringstream class is extremely useful in parsing input.

# **Count number of words in a string:**

#include <iostream>

#include <string>

#include <vector>

#include <sstream> // This class includes all stringstream's member fn. This is necessary.

using namespace std;

int countWord(string str, vector<string> **&**v\_str){ // vector<string> &v\_str -> Pass by ref is needed to send back the filled vector to caling fn.

stringstream obj(str);

string word;

int count =0;

while(obj >> word){

v\_str.push\_back(word);

count++;

}

return count;

}

int main() {

string str = "I am Saurav Kumar. I work in Wipor's office.";

vector<string> wordlist;

vector<string>::iterator v\_str\_itr;

int idx =1;

cout<<"No of words = "<<countWord(str,wordlist)<<endl;

for(v\_str\_itr = wordlist.begin(); v\_str\_itr != wordlist.end(); v\_str\_itr++){

cout<<"word# "<<idx<<" = "<<\*v\_str\_itr<<endl;

idx++;

}

return 0;

}

**Print frequencies of individual words in a string**

Input : Geeks For Geeks Quiz Geeks Quiz Practice Practice

Output : For -> 1

Geeks -> 3

Practice -> 2

Quiz -> 2

Input : Word String Frequency String

Output : Frequency -> 1

String -> 2

Word -> 1

Count frequencies of words

|  |
| --- |
| // CPP program to demonstrate use of stringstream  // to count frequencies of words.  #include <bits/stdc++.h>  using namespace std;    void printFrequency(string st)  {      // each word it mapped to it's frequency      map<string, int> FW;      stringstream ss(st); // Used for breaking words      string Word; // To store individual words        while (ss >> Word)          FW[Word]++;        map<string, int>::iterator m;      for (m = FW.begin(); m != FW.end(); m++)          cout << m->first << " -> "               << m->second << "\n";  }    // Driver code  int main()  {      string s = "Geeks For Geeks Ide";      printFrequency(s);      return 0;  } |

Output:

For -> 1

Geeks -> 2

Ide -> 1

# Removing spaces from a string using Stringstream

Solution to removing spaces from a string is already posted [here](https://www.geeksforgeeks.org/remove-spaces-from-a-given-string/). In this article another solution using **stringstream** is discussed.

**Algorithm**

1. Enter the whole string into stringstream.

2. Empty the string.

3. Extract word by word and concatenate to the string.

|  |
| --- |
| // C++ program to remove spaces using stringstream  #include<bits/stdc++.h>  using namespace std;    /\* Function to remove spaces \*/  string removeSpaces(string str)  {      stringstream ss;      string temp;        /\* Storing the whole string into string stream \*/      ss << str;        /\* Making the string empty \*/      str = "";        /\* Running loop till end of stream \*/      while (!ss.eof())      {          /\* extracting word by word from stream \*/          ss >> temp;            /\* concatenating in the string to be            returned\*/          str = str + temp;      }      return str;  }    /\*Driver function \*/  int main()  {      string s = "This is a test";      cout << removeSpaces(s) << endl;        s = "geeks for geeks";      cout << removeSpaces(s) << endl;        s = "geeks quiz is awsome!";      cout << removeSpaces(s) << endl;        s = "I   love     to     code";      cout << removeSpaces(s) << endl;        return 0;  } |

Output:

Thisisatest

geeksforgeeks

geeksquizisawsome!

Ilovetocode

# <sstream>

String streams

Header providing string stream classes:

### Classes

#### Narrow characters (char)

[**istringstream**](http://www.cplusplus.com/reference/sstream/istringstream/)

Input string stream (class )

[**ostringstream**](http://www.cplusplus.com/reference/sstream/ostringstream/)

Output string stream (class )

[**stringstream**](http://www.cplusplus.com/reference/sstream/stringstream/)

Input/output string stream (class )

[**stringbuf**](http://www.cplusplus.com/reference/sstream/stringbuf/)

String stream buffer (class )

# std::[istringstream](http://www.cplusplus.com/reference/sstream/istringstream/)

**Input string stream**

Input stream class to operate on strings.  
  
Objects of this class use a [*string buffer*](http://www.cplusplus.com/stringbuf) that contains a sequence of characters. This sequence of characters can be accessed directly as a [string](http://www.cplusplus.com/string) object, using member [str](http://www.cplusplus.com/istringstream::str).

The characters in the sequence can be extracted from the stream using any operation allowed on [*input streams*](http://www.cplusplus.com/istream).

# istringstream

**Construct object**

Constructs a [istringstream](http://www.cplusplus.com/istringstream) object:

***(1) empty constructor (default constructor)***

Constructs an [istringstream](http://www.cplusplus.com/istringstream) object with an empty sequence as content.  
Internally, its [istream](http://www.cplusplus.com/istream::istream) base constructor is passed a pointer to a [stringbuf](http://www.cplusplus.com/stringbuf) object constructed with an argument based on which.

***(2) initialization constructor***

Constructs an [istringstream](http://www.cplusplus.com/istringstream) object with a copy of str as content.  
Internally, its [istream](http://www.cplusplus.com/istream::istream) base constructor is passed a pointer to a [stringbuf](http://www.cplusplus.com/stringbuf) object constructed with arguments based on str and which.

***(3) copy constructor (deleted)***

Deleted (no copy constructor).

***(4) move constructor***

Acquires the contents of x.  
First, the function move-constructs both its base [istream](http://www.cplusplus.com/istream::istream) class from x and a [stringbuf](http://www.cplusplus.com/stringbuf) object from x's internal [streambuf](http://www.cplusplus.com/streambuf) object, and then associates them by calling member [set\_rdbuf](http://www.cplusplus.com/ios::set_rdbuf).  
x is left in an unspecified but valid state.  
It is unspecified whether the sequence controlled by the internal [stringbuf](http://www.cplusplus.com/stringbuf) object is the one in x before the call, or a copy of it. In any case, both objects have internal *string buffers* that use independent sequences after the call.

|  |  |
| --- | --- |
| *default (1)* | explicit istringstream (ios\_base::openmode which = ios\_base::in); |
| *initialization (2)* | explicit istringstream (const string& str,  ios\_base::openmode which = ios\_base::in); |
| *copy (3)* | istringstream (const istringstream&) = delete; |
| *move (4)* | istringstream (istringstream&& x); |

### Parameters

str

A [string](http://www.cplusplus.com/string) object, whose content is copied.

x

A [istringstream](http://www.cplusplus.com/istringstream) object, whose value is moved.

which

Open mode: Access given by the internal [stringbuf](http://www.cplusplus.com/stringbuf) object to its internal sequence of characters. It is an object of member type [openmode](http://www.cplusplus.com/ios_base::openmode) for which any combination of the following member values is significant:

* [C++98](javascript:switch2.select(1))
* [C++11](javascript:switch2.select(2))

|  |  |  |
| --- | --- | --- |
| **member constant** | **stands for** | **access** |
| ios\_base::in\* | **in**put | The sequence supports input operations. |
| ios\_base::out | **out**put | The sequence supports output operations. |
| ios\_base::ate | **at e**nd | The writing position is at the end after construction, and after every time the content is reset with member [str](http://www.cplusplus.com/istringstream::str). |

Other values of type [ios\_base::openmode](http://www.cplusplus.com/ios_base::openmode) (such as [ios\_base::app](http://www.cplusplus.com/ios_base::app)) may also be specified, although whether they have an effect on [istringstream](http://www.cplusplus.com/istringstream) objects depends on the library implementation.

\* ios\_base::in is always set for [istringstream](http://www.cplusplus.com/istringstream) objects (even if explicitly not set in argument which).  
Note that even though [istringstream](http://www.cplusplus.com/istringstream) is an input stream, its internal [stringbuf](http://www.cplusplus.com/stringbuf) object may be set to also support output operations. This influences certain operations, such as [putback](http://www.cplusplus.com/istream::putback), that in [istringstream](http://www.cplusplus.com/istringstream) may alter the contents of the sequence.

### Example

|  |  |  |
| --- | --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 | // istringstream constructors.  #include <iostream> // std::cout  #include <sstream> // std::istringstream  #include <string> // std::string  int main () {  std::string stringvalues = "125 320 512 750 333";  std::istringstream iss (stringvalues);  for (int n=0; n<5; n++)  {  int val;  iss >> val;  std::cout << val\*2 << '\n';  }  return 0;  } | [Edit & Run](http://www.cplusplus.com/reference/sstream/istringstream/istringstream/) |

Output:

|  |
| --- |
| 250  640  1024  1500  666 |

# Converting a string to other data type: stringstream

A string variable cannot be converted to other type of data by casting because it's not a built-in data type. To help this issue, C++ provides us a <sstream> library.

A value can be loaded into a **stringstream** object using output stream operator(<<) with **cout**, and then the content can be extracted from the **stringstream** object with input operator(>>) with **cin**.

If we want to re-use a **stringstream** object, it must be back in the original state. In other words, it should be set as empty string via **str()** method, and its status bit flag (**good**, **bad**, **eof**,or **fail**) needs to be cleared via **clear()** method of the **stringstream** class.

#include <sstream>;

#include <string>

#include <iostream>

int main()

{

std::string number\_s = "99";

int ss\_to\_number;

std::stringstream ss;

ss << number\_s; // load a string to ss

ss >> ss\_to\_number; // extract an integer from ss

int n = ss\_to\_number / 3;

std::cout << "The extracted integer is " << n << std::endl;

// to re-use ss, we need to empty and clear

ss.str("");

ss.clear();

int number\_i = 99;

std::string ss\_to\_string;

std::string text\_out =

"We extracted a string by converting an int sstream";

ss << number\_i; // load an integer

ss >> ss\_to\_string; // extract a string from ss

text\_out += " from " + ss\_to\_string;

std::cout << text\_out << std::endl;

return 0;

}

Output is:

The extracted integer is 33

We extracted a string by converting an int sstream from 99