Object Oriented Programming by C++

Large Data Processing

String, and File

2017. 8.

Sungwon Lee / Professor

Email: drsungwon@khu.ac.kr
Web: http://mobilelab.khu.ac.kr/

Modified & taught by Jinwoo Choi
in 2023 Spring
Email: jinwoochoi@khu.ac.kr
Webpage: https://sites.google.com/site/jchoivision/

Contents

- Component in C++
- String Object
- File Input & Output

Component in Software and Hardware

- Hardware Component:
  - Processor
  - Memory
  - Video Card
  - Disk Drive
  - Keyboard
  - Mouse
  - Monitor

- Software Component?:
  - *File* in Storage (Disk Drive, USB, HDD, SSD, RAM)
  - Character *String* in RAM, and so on …

Example: **File** Software Component?

- **File** software component (= *object, class*) consists of:
  - Data (= *member data*) :
    - Texts
    - Numbers
    - Images
    - Moving Pictures, etc.
  - Functions (= *member function or method*) :
    - Create
    - Open
    - Close
    - Remove
    - Search, etc.

String Software Component

- **String** software component (= *object, class*) consists of:
  - Data (= *member data*) :
    - Characters
  - Functions (= *member function or method*) :
    - String addition (= merge) “I am” + “a student” -> “I am a student”
    - Character search in the String
    - Get size of the String
    - Remove all characters in the String, etc.

Using String Object

```
#include <iostream>
#include <string>

using namespace std;

int main()
{
    string name1 = "joe";
    string name2;
    cout << name1 << '\n';
    cin >> name1;
    cout << name1 << '\n';
    name1 = "jane";
    cout << name1 << '\n';
    cout << name2 << '\n';
}
```

char c = “X”;

New include,
New variable,
Same operation !!

Member Functions in String Object (1/2)

And More!

string str="This is a string" 길이: 16
str2 = "!!!"

str[0] -> "T"
str[10] -> "s"
str[3] -> "s"

- `operator[]`—provides access to the value stored at a given index within the string
- `operator=`—assigns one string to another
- `operator+=`—appends a string or single character to the end of a `string` object

str += str2;
str -> "This is a string!!!"

- `at`—provides bounds-checking access to the character stored at a given index
- `length`—returns the number of characters that make up the string

str.length() -> 16

- `size`—returns the number of characters that make up the string (same as `length`)
- `find`—locates the index of a substring within a `string` object

str.find("s") -> 10

- `substr`—returns a new `string` object made of a substring of an existing `string` object
- `empty`—returns true if the string contains no characters; returns false if the string contains one or more characters
- `clear`—removes all the characters from a string

String Object

Member Functions in String Object (2/2)

Want More?

string - C++ Reference

www.cplusplus.com/reference/string/string/?kw=string

OpenSource WebSites TechTips GoormEDU 터키여행 이것저것

cplusplus.com Search: Go

Reference <string> string

C++
- Information
- Tutorials
- Reference
- Articles
- Forum

Reference
- C library:
- Containers:
- Input/Output:
- Multi-threading:
- Other:
  - <algorithm>
  - <bitset>
  - <chrono>
  - <codecvt>
  - <complex>
  - <exception>
  - <functional>
  - <initializer_list>
  - <iterator>
  - <limits>
  - <locale>
  - <memory>
  - <new>
  - <numeric>
  - <random>
  - <ratio>
  - <regex>
  - <stdexcept>
  - <string>
  - <system_error>
  - <tuple>
  - <typeindex>
  - <typeinfo>
  - <type_traits>
  - <utility>
  - <valarray>

<string>
- class templates:
  - basic_string
  - char_traits
- classes:
  - string
  - u16string
  - u32string
  - wstring
- functions:
  - stod

Ad closed

You were redirected to cplusplus.com/s

class

std::**string**

```
typedef basic_string<char> string;
```

String class

Strings are objects that represent sequences of characters

The standard `string` class provides support for such objec bytes, but adding features specifically designed to operate

The `string` class is an instantiation of the basic_string clas with its default char_traits and allocator types (see basic_s

Note that this class handles bytes independently of the enc variable-length characters (such as UTF-8), all members o will still operate in terms of bytes (not actual encoded char

Member types

member type	def
value_type	char
traits_type	char_traits<char>
allocator_type	allocator<char>
reference	char&
const_reference	const char&
pointer	char*
const_pointer	const char*
iterator	a random access iterator to cha
const_iterator	a random access iterator to con
reverse_iterator	reverse_iterator<iterator>
const_reverse_iterator	reverse_iterator<const_iter
difference_type	ptrdiff_t
size_type	size_t

Member functions

(constructor)	Construct string object (public
(destructor)	String destructor (public memb
operator=	String assignment (public mem

string - C++ Reference

www.cplusplus.com/reference/string/string/?kw=st

OpenSource WebSites TechTips GoormEDU 터키여행

- classes:
  - string
  - u16string
  - u32string
  - wstring
- functions:
  - stod
  - stof
  - stoi
  - stol
  - stold
  - stoll
  - stoul
  - stoull
  - to_string
  - to_wstring

string
- string::string
- string::~string
- member functions:
  - string::append
  - string::assign
  - string::at
  - string::back
  - string::begin
  - string::capacity
  - string::cbegin
  - string::cend
  - string::clear
  - string::compare
  - string::copy
  - string::crbegin
  - string::crend
  - string::c_str
  - string::data
  - string::empty
  - string::end
  - string::erase
  - string::find
  - string::find_first_not_of
  - string::find_first_of
  - string::find_last_not_of
  - string::find_last_of
  - string::front
  - string::get_allocator
  - string::insert
  - string::length
  - string::max_size
  - string::operator+=
  - string::operator=
  - string::operator[]
  - string::pop_back
  - string::push_back
  - string::rbegin
  - string::rend

Member functions

(constructor)	Construct string object (public member function)
(destructor)	String destructor (public member function)
operator=	String assignment (public member function)

Iterators:

begin	Return iterator to beginning (public member function)
end	Return iterator to end (public member function)
rbegin	Return reverse iterator to reverse beginning (public member function)
rend	Return reverse iterator to reverse end (public member function)
cbegin	Return const_iterator to beginning (public member function)
cend	Return const_iterator to end (public member function)
crbegin	Return const_reverse_iterator to reverse beginning (public member function)
crend	Return const_reverse_iterator to reverse end (public member function)

Capacity:

size	Return length of string (public member function)
length	Return length of string (public member function)
max_size	Return maximum size of string (public member function)
resize	Resize string (public member function)
capacity	Return size of allocated storage (public member function)
reserve	Request a change in capacity (public member function)
clear	Clear string (public member function)
empty	Test if string is empty (public member function)
shrink_to_fit	Shrink to fit (public member function)

Element access:

operator[]	Get character of string (public member function)
at	Get character in string (public member function)
back	Access last character (public member function)
front	Access first character (public member function)

Modifiers:

operator+=	Append to string (public member function)
append	Append to string (public member function)
push_back	Append character to string (public member function)
assign	Assign content to string (public member function)
insert	Insert into string (public member function)
erase	Erase characters from string (public member function)
replace	Replace portion of string (public member function)
swap	Swap string values (public member function)
pop_back	Delete last character (public member function)

Example using String Object (1/2)

Be Patient

Listing 13.1: stringoperations.cpp

```
#include <iostream>
#include <string>

int main() {
    // Declare a string object and initialize it
    std::string word = "fred";
    // Prints 4, since word contains four characters
    std::cout << word.length() << '\n';
    // Prints "not empty", since word is not empty
    if (word.empty())
        std::cout << "empty\n";
    else
        std::cout << "not empty\n";
    // Makes word empty
    word.clear();
    // Prints "empty", since word now is empty
    if (word.empty())
        std::cout << "empty\n";
    else
        std::cout << "not empty\n";
    // Assign a string using operator= method
    word = "good";
    // Prints "good"
    std::cout << word << '\n';
    // Append another string using operator+= method
    word += "-bye";
    // Prints "good-bye"
    std::cout << word << '\n';
```

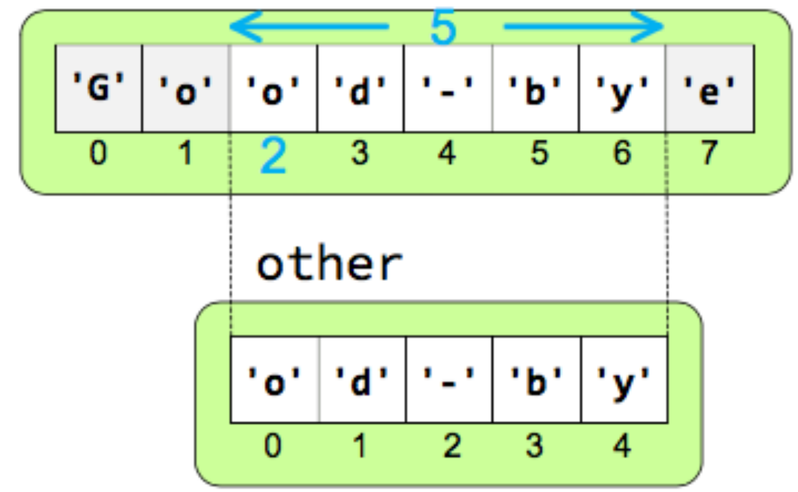
word -> “”

Example using String Object (2/2)

```
    // Print first character using operator[] method
    std::cout << word[0] << '\n';
    // Print last character
    std::cout << word[word.length() - 1] << '\n';
    // Prints "od-by", the substring starting at index 2 of length 5
    std::cout << word.substr(2, 5);
    std::string first = "ABC", last = "XYZ";
    // Splice two strings with + operator
    std::cout << first + last << '\n';
    std::cout << "Compare " << first << " and ABC: ";
    if (first == "ABC")
        std::cout << "equal\n";
    else
        std::cout << "not equal\n";
    std::cout << "Compare " << first << " and XYZ: ";
    if (first == "XYZ")
        std::cout << "equal\n";
    else
        std::cout << "not equal\n";
}
```

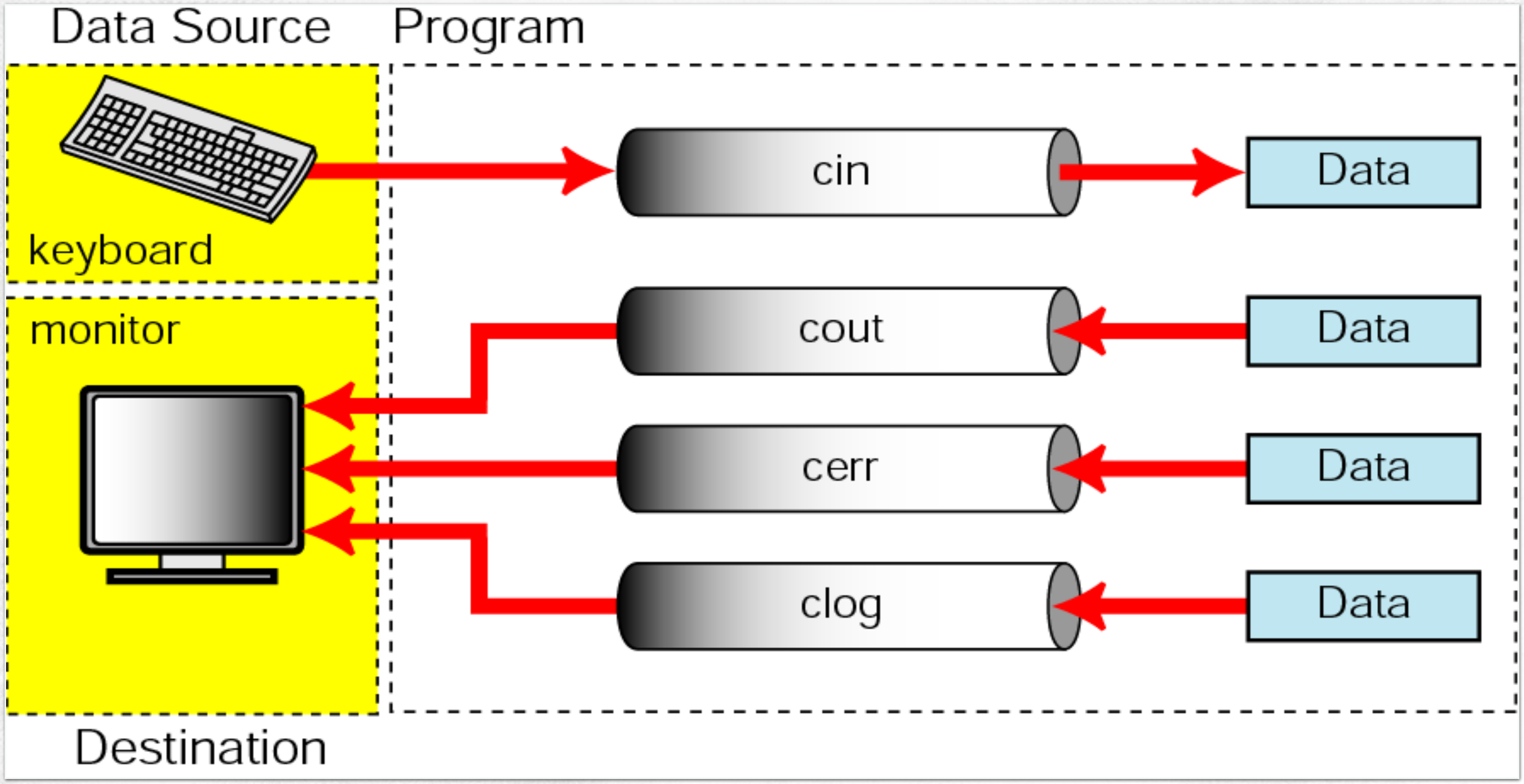
ABCXYZ first += “!!!” -> first = first + “!!!”

```
string word = "Good-bye";
string other = word.substr(2, 5);
```



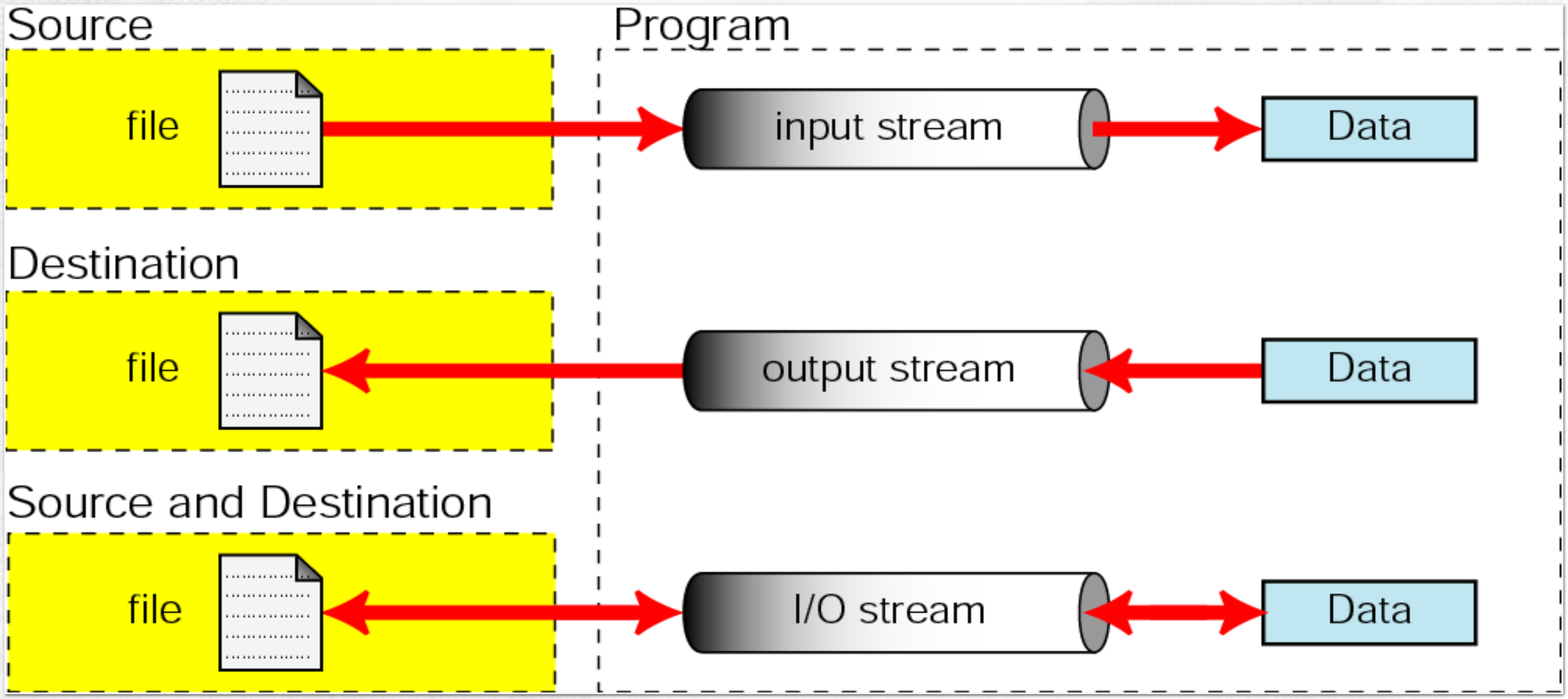
Console Stream

- Standard streams are created, connected, and disconnected automatically



File Stream

- File streams are created, connected to files, and disconnected from files by the programmer



File Stream and Basic Member Functions

- File Streams in C++
  - ifstream (for reading from file)
  - ofstream (for writing to file)
  - fstream (read and write)
- Creating and Disconnecting File Streams

  (1) First define stream objects
    - `ifstream [stream variable name];`
    - `ofstream [stream variable name];`
    - `fstream [stream variable name];`

  (2) Connecting file streams
    - `Open ()`

  (3) Disconnecting file streams
    - `Close ()`

Reading and Writing from/to File

- Reading from file

```
ifstream fsTemp;
int a;
fsTemp.open (“temp.txt");
fsTemp >> a;
cout << a;
```

```
10
8
7
1
```

- Writing to file

```
ofstream fsTemp;
int a;
fsTemp.open (“temp.txt”);
cin >> a;
fsTemp << a;
```

Reading and Writing from/to File

- Reading from file

```
ifstream fsTemp;
int a;
fsTemp.open (“temp.txt");
fsTemp.get(a);
cout << a;
```

```
10
8
7
1
```

- Writing to file

```
ofstream fsTemp;
int a;
fsTemp.open (“temp.txt”);
cin >> a;
fsTemp.put(a);
```

Simple Example for File Stream

```cpp
// basic file operations
#include <iostream>
#include <fstream>
using namespace std;

int main () {
  ofstream myfile;
  myfile.open ("example.txt");
  myfile << "Writing this to a file.\n";
  myfile.close();
  return 0;
}
```

```
[file example.txt]
Writing this to a file.
```

```cpp
// reading a text file
#include <iostream>
#include <fstream>
#include <string>
using namespace std;

int main () {
  string line;
  ifstream myfile ("example.txt");
  if (myfile.is_open())
  {
    while ( getline (myfile,line) )
    {
      cout << line << '\n';
    }
    myfile.close();
  }

  else cout << "Unable to open file";

  return 0;
}
```

```
This is a line.
This is another line.
```

For Visual Studio
(Desktop Environment)

Simple Example for File Stream

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

int main() {
    ofstream myfile1;
    string iLine = "Writing this to a file.";
    myfile1.open("example.txt");
    myfile1 << iLine << endl;

    // If you want, add more file output at here like :
    /*
    while(iLine != "quit")
    {
        cout << endl << "Please enter an word (type \"quit\" to stop): ";
        cin >> iLine;
        myfile1 << iLine << endl;
    }
    */

    myfile1.close();

    ifstream myfile2("example.txt");
    string oLine;
    if(myfile2.is_open())
    {
        while(getline(myfile2, oLine))
        {
            cout << endl << oLine;
        }
    }
    myfile2.close();

    return 0;
}
```

For Goorm EDU
(Cloud Environment)

 Reference: http://www.cplusplus.com/doc/tutorial/files/

Formatting Data using Control Variables

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

int main()
{
  ofstream fsTemp;
  fsTemp.open("temp.txt");
  int a=123;
  double b=12.12345678;

  fsTemp.width(15);
  fsTemp << a << endl;
  fsTemp.width(15);
  fsTemp.precision(10);
  fsTemp << b << endl;

  fsTemp.close();
  return 0;
}
```

- *.width( )*: determine how may display positions are to be used to display the data

- *.fill( )*: determines the nondata character that is to print when the print width is greater than the data width

- *.precision( )*: determines the number of digits to be displayed after the decimal point

Stream Flags (applied to console, file, and etc)

Flag	Meaning
boolalpha	Boolean values can be input/output using the words "true" and "false".
dec	Numeric values are displayed in decimal.
fixed	Display floating point values using normal notation (as opposed to scientific).
hex	Numeric values are displayed in hexidecimal.
internal	If a numeric value is padded to fill a field, spaces are inserted between the sign and base character.
left	Output is left justified.
oct	Numeric values are displayed in octal.
right	Output is right justified.
scientific	Display floating point values using scientific notation.
showbase	Display the base of all numeric values.
showpoint	Display a decimal and extra zeros, even when not needed.
showpos	Display a leading plus sign before positive numeric values.
skipws	Discard whitespace characters (spaces, tabs, newlines) when reading from a stream.
unitbuf	Flush the buffer after each insertion.
uppercase	Display the "e" of scientific notation and the "x" of hexidecimal notation as capital letters.

 Reference: https://doc.bccnsoft.com/docs/cppreference_en/io_flags.html

(FYI) Example using Stream Flags - Program.1

```
#include <iostream>
using namespace std;

int main ()
{
    char aChar;

    // Read Characters
    cout << "Enter <space z>       : ";
    cin  >> aChar;
    cout << "Character read is     : " << aChar;
    cin.unsetf (ios::skipws);
    cin  >> aChar;
    cout << "\nCharacter read is     : " << (int)aChar;
    cout << "\nEnter <space z> again: ";
    cin  >> aChar;
    cout << "Character read is     : " << (int)aChar;
    cin  >> aChar;
    cout << "\nCharacter read is     : " << (int)aChar;
    cin  >> aChar;
    cout << "\nCharacter read is     : " << (int)aChar;
    return 0;
}   // main

/* Results:
Enter <space z>       :  z
Character read is     : z
Character read is     : 10
Enter <space z> again:  z
Character read is     : 32
Character read is     : 122
Character read is     : 10

*/
```

(FYI) Example using Stream Flags - Program.2

```
#include <iostream>
using namespace std;

int main ()
{
    cout << "Default justification: |";
    cout.width(10);
    cout << 12345 << "|\n";
    cout.setf (ios::left, ios::adjustfield);
    cout << "Left justification   : |";
    cout.width(10);
    cout << 12345 << "|\n";

    cout.setf (ios::right, ios::adjustfield);
    cout << "Right justification  : |";
    cout.width(10);
    cout << 12345 << "|\n";
    return 0;
}   // main

/*  Results:
    Default justification: |     12345|
    Left justification   : |12345     |
    Right justification  : |     12345|
*/
```

(FYI) Example using Stream Flags - Program.3

```
#include <iostream>
#include <iomanip>
using namespace std;

int main ()
{
    cout.setf   (ios::scientific);
    cout << "...scientific format  : " << 3141.59 << endl;
    cout.unsetf (ios::scientific);

    cout.setf (ios::fixed);
    cout << "...with fixed point   : " << 3141.59 << endl;
    return 0;
}   // main

/*  Results
...scientific format  : 3.141590e+03
...with fixed point   : 3141.590000
*/
```

(FYI) Example using Stream Flags - Program.4 (1/5)

```
1  /* Create a grades file for transmission to Registrar.
2  Written by:
3  Date:
4  */
5  #include <iostream>
6  #include <fstream>
7  #include <iomanip>
8  #include <cstdlib>
9  //#include <StdAfx.h>
10 using namespace std;
11
12 bool getStu (ifstream& stuFile, int& stuID, int& exam1, int& exam2, int& final);
13 void writeStu (ofstream& gradesFile, int stuID, int avrg, char grade);
14 void calcGrade (int exam1, int exam2, int final, int& avrg, char& grade);
15
16 int main ()
17 {
18     ifstream stuFile;
19     cout << "Begin student grades\n";
20     stuFile.open ("ch7STUFL.DAT");
21     if (!stuFile)
22     {
23         cerr << "\aError opening student file\n";
24         exit (100);
25     } // if open file
26
```

(FYI) Example using Stream Flags - Program.4 (2/5)

```
27      ofstream gradesFile;
28      gradesFile.open ("ch7STUGR.DAT");
29      if (!gradesFile)
30      {
31          cerr << "\aError opening grades file\n";
32          exit (102);
33      } // if open fail
34
35      int stuID;
36      int exam1;
37      int exam2;
38      int final;
39      int avrg;
40      char grade;
41
42      while (getStu (stuFile, stuID, exam1, exam2, final))
43      {
44          calcGrade (exam1, exam2, final, avrg, grade);
45          writeStu (gradesFile, stuID, avrg, grade);
46      } // while
47
48      stuFile.close ();
49      gradesFile.close();
50      cout << "End student grades\n";
51      return 0;
52  } // main
```

(FYI) Example using Stream Flags - Program.4 (3/5)

```
53 /* ============================getStu=============================
54 Reads data from student file.
55
56 Pre stuFile is an open file stuID, exam1, exam2, final-ref's to int
57 Post reads student ID and exam scores into parameter references
58 Return true if data read -- false if end of file
59 */
60 bool getStu (ifstream& stuFile, int& stuID, int& exam1, int& exam2, int& final)
61 {
62     //!stuFile is either eof or bad file
63     stuFile >> stuID >> exam1 >> exam2 >> final;
64     if (!stuFile)
65         //Read problem or at end of file
66         return false;
67     return true;
68 } // getStu
69
70 /* ============================clacGrade============================
71 Determine student grade based on absolute scale.
72 Pre exam1, exam2, and final contain acores avrg and grade are references to variables
73 Post Average and grade copied to references
74 */
75 void calcGrade (int exam1, int exam2, int final, int& avrg, char& grade)
76 {
77     avrg = (exam1 + exam2 +final) / 3;
78     if (avrg >= 90)
```

(FYI) Example using Stream Flags - Program.4 (4/5)

```
79          grade = 'A';
80      else if (avrg >= 80)
81          grade = 'B';
82      else if (avrg >= 70)
83          grade = 'C';
84      else if (avrg >= 60)
85          grade = 'D';
86      else
87          grade = 'F';
88      return;
89  } // calcGrade
90  /* ==============================writeStu=============================
91  Writes student grade data to output fiel.
92  Pre gradesFiel is an open file stuID, avrg, and grade have values to write
93  Post Data written to file
94  */
95  void writeStu (ofstream& gradesFile, int stuID, int avrg, char grade)
96  {
97      gradesFile.fill ('0');
98      gradesFile << setw (4) << stuID;
99      gradesFile.fill (' ');
100     gradesFile << setw (3) << avrg;
101     gradesFile << ' ' << grade << endl;
102     return;
103 } // writeStu
```

(FYI) Example using Stream Flags - Program.4 (5/5)

```
/*  Results:
Begin student grades
End student grades

Input--------
0090 90 90 90
0089 88 90 89
0081 80 82 81
0079 79 79 79
0070 70 70 70
0069 69 69 69
0060 60 60 60
0059 59 59 59

Output----
0090 90 A
0089 89 B
0081 81 B
0079 79 C
0070 70 C
0069 69 D
0060 60 D
0059 59 F
*/
```

Reuse this for future usage.

File Input & Output

Member Functions in File Object

Want More?

fstream - C++ Reference

www.cplusplus.com/reference/fstream/fstream/?kw=fstream

OpenSource WebSites TechTips GoormEDU 터키여행 이것저것

cplusplus.com Search: Go Not log

Reference <fstream> fstream register log

C++

- Information
- Tutorials
- Reference
- Articles
- Forum

Reference

- *C library:*
- *Containers:*
- *Input/Output:*
  - <fstream>
  - <iomanip>
  - <ios>
  - <iosfwd>
  - <iostream>
  - <istream>
  - <ostream>
  - <sstream>
  - <streambuf>
- *Multi-threading:*
- *Other:*

<fstream>

- *class templates:*
  - basic_filebuf
  - basic_fstream
  - basic_ifstream
  - basic_ofstream
- *classes:*
  - filebuf
  - fstream
  - ifstream
  - ofstream
  - wfilebuf
  - wfstream
  - wifstream
  - wofstream

fstream

fstream::fstream

- *public member functions:*
  - fstream::close
  - fstream::is_open
  - fstream::open
  - fstream::operator=
  - fstream::rdbuf
  - fstream::swap
- *non-member overloads:*
  - swap (fstream)

You were redirected to cplusplus.com/**fstream** || See search results for: **"fstream"**

class

std::fstream <fstr

```
typedef basic_fstream<char> fstream;
```

Input/output file stream class

ios_base ← ios ← istream ← iostream ← fstream

ostream ← istream

Input/output stream class to operate on files.

Objects of this class maintain a `filebuf` object as their *internal stream buffer*, which performs input/output operat on the file they are associated with (if any).

File streams are associated with files either on construction, or by calling member `open`.

This is an instantiation of `basic_fstream` with the following template parameters:

template parameter	definition	comments
charT	char	Aliased as member char_type
traits	char_traits<char>	Aliased as member traits_type

Apart from the internal *file stream buffer*, objects of this class keep a set of internal fields inherited from `ios_base` and `istream`:

	field	member functions	description
Formatting	format flags	flags setf unsetf	A set of internal flags that affect how certain input/output operations a interpreted or generated. See member type fmtflags.
	field width	width	Width of the next formatted element to insert.
	display precision	precision	Decimal precision for the next floating-point value inserted.
	locale	getloc imbue	The locale object used by the function for formatted input/output operations affected by localization properties.
	fill character	fill	Character to pad a formatted field up to the *field width* (width).
State	error state	rdstate setstate clear	The current error state of the stream. Individual values may be obtained by calling good, eof, fail and bad. See member type iostate.
	exception mask	exceptions	The state flags for which a failure exception is thrown. See member type iostate.
Other	callback stack	register_callback	Stack of pointers to functions that are called when certain events occu
	extensible arrays	iword pword xalloc	Internal arrays to store objects of type long and void*.
	tied stream	tie	Pointer to output stream that is flushed before each i/o operation on t stream.
	stream buffer	rdbuf	Pointer to the associated streambuf object, which is charge of all input/output operations.
	character count	gcount	Count of characters read by last unformatted input operation.

fstream - C++ Reference

www.cplusplus.com/reference/

OpenSource WebSites TechTips

The class declares the following member types:

member type	definition
char_type	char
traits_type	char_traits<char>
int_type	int
pos_type	streampos
off_type	streamoff

These member types are inherited from its base classes `istream`, `ostream` and `ios_base`:

event	Type to indicate event type (public member type)
event_callback	Event callback function type (public member type)
failure	Base class for stream exceptions (public member class)
fmtflags	Type for stream format flags (public member type)
Init	Initialize standard stream objects (public member class)
iostate	Type for stream state flags (public member type)
openmode	Type for stream opening mode flags (public member type)
seekdir	Type for stream seeking direction flag (public member type)
sentry (istream)	Prepare stream for input (public member class)
sentry (ostream)	Prepare stream for output (public member class)

fx Public member functions

(constructor)	Construct object and optionally open file (public member function)
open	Open file (public member function)
is_open	Check if a file is open (public member function)
close	Close file (public member function)
rdbuf	Get the associated filebuf object (public member function)
operator= C++11	Move assignment (public member function)
swap C++11	Swap internals (public member function)

fx Public member functions inherited from `istream`

operator>>	Extract formatted input (public member function)
gcount	Get character count (public member function)
get	Get characters (public member function)
getline	Get line (public member function)
ignore	Extract and discard characters (public member function)
peek	Peek next character (public member function)
read	Read block of data (public member function)
readsome	Read data available in buffer (public member function)
putback	Put character back (public member function)
unget	Unget character (public member function)
tellg	Get position in input sequence (public member function)
seekg	Set position in input sequence (public member function)
sync	Synchronize input buffer (public member function)

fx Public member functions inherited from `ostream`

operator<<	Insert formatted output (public member function)
put	Put character (public member function)
write	Write block of data (public member function)

Object Oriented Programming by C++

Sungwon Lee / Professor

Email: drsungwon@khu.ac.kr

Web: http://mobilelab.khu.ac.kr/

...

- ...