|  |
| --- |
| **Programming Fundamentals** |
| **(CL 214)** |
| **LABORATORY MANUAL** |
| **Spring 2021** |
| **C:\Users\Aamer\Desktop\nu-new.png**  **LAB 12** |
| ***Strings & Header file Implementation***  **Engr. Ibrar Khan** |
| **Engr. Sana Saleh** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | \_\_\_\_\_\_\_\_\_\_ | | | \_\_\_ |
| STUDENT NAME | | ROLL NO | | | SEC |
|  | | | | | |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | | | |
| LAB ENGINEER SIGNATURE & DATE | | | | | |
| **MARKS AWARDED: /10** | | | | | |
|  | | | | | |
|  | | | | | |
| **NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES (NUCES), ISLAMABAD** | | | | | |
|  | | | | | |
| Last Edited by: | Engr. Sana Saleh | | Version: | 2.01 | |
| Prepared by: | Engr. Sana Saleh | | Date: | 12 Aug, 2017 | |
| Verified by: | Engr. Shahid Qureshi | | Date: | 19 Jan, 2019 | |

|  |  |
| --- | --- |
| **LAB 12** | **Strings & Header file Implementation** |

**Lab Objectives:**

1. To learn about use and built in functions of *strings.*
2. To learn how to create and use separate class header and implementation files.

**Software Required:**

* Visual Studio 2010

**Introduction:**

## Separate Header and Implementation Files

* 1. **Header File:**

Class declarations can be put in header files in order to facilitate reuse in multiple files or multiple projects. Traditionally, the class declaration is put in a header file of the same name as the class, with a .h extension.

Here’s a class Date, broken into a .cpp and .h file. Following is the header file for the class Date.

**Date.h:**

class Date

{

private:

int m\_year;

int m\_month;

int m\_day;

public:

Date(int year, int month, int day);

void SetDate(int year, int month, int day);

int getYear() { return m\_year; }

int getMonth() { return m\_month; }

int getDay() { return m\_day; }

};

* 1. **Implementation File:**

The member function definitions for a class are stored in a separate .cpp file, which is called the class implementation file. The file usually has the same name as the class, with the .cpp extension. For example the Time class member functions would be defined in the file Time.cpp.

**Date.cpp:**

#include "Date.h"

#include <iostream>

// Date constructor

Date::Date(int year, int month, int day)

{

SetDate(year, month, day);

}

// Date member function

void Date::SetDate(int year, int month, int day)

{

m\_month = month;

m\_day = day;

m\_year = year;

}

* 1. **Client/User Code:**

Client/user code, is the one that includes the main function.

#include <iostream>

using namespace std;

#include "Date.h"

int main()

{

Date t1(1990, 12, 5);

int yr=t1.getYear();

cout<<”The year is “<<yr<<endl;

return 0;

}

* 1. **Advantages:**

The advantages of storing class definition in separate file are

1. The class is reusable

2. The clients of the class know what member functions the class provides, how to call them and what return types to expect

3. The clients do not know how the class's member functions are implemented.

**Note:**

In Dev C++, Create a new project. In project add header file, implementation file and main.cpp.

1. ***Strings:***
   1. **Strings:**

In C++, string is an object of std::string class that represents sequence of characters.  To implement string using string class, we need to add header file. Basic difference between character array and a string is, in case of character array, the size has to be allocated at the time of declaration, i.e all memory once allocated is fixed and cannot be altered at run time. Whereas, for string, there is no need to specify the size and to allocate fixed memory at the time of declaration.

* + 1. **Initialization at Declaration:**

There are many overloaded constructor in the string library, through which we can initialize the string. Few built-in constructors are mentioned below:

|  |
| --- |
| ***(1) empty string constructor (default constructor)***  Constructs an [empty](https://www.cplusplus.com/string::empty) string, with a [length](https://www.cplusplus.com/string::length) of zero characters.  ***(2) copy constructor***  Constructs a copy of *str*.  ***(3) substring constructor***  Copies the portion of *str* that begins at the character position *pos* and spans *len* characters (or until the end of *str*, if either *str* is too short or if *len* is [string::npos](https://www.cplusplus.com/string::npos)).  ***(4) from c-string***  Copies the null-terminated character sequence (C-string) pointed by *s*.  ***(5) from buffer***  Copies the first *n* characters from the array of characters pointed by *s*.  ***(6) fill constructor***  Fills the string with *n* consecutive copies of character *c*.  ***(7) range constructor***  Copies the sequence of characters in the range [first,last), in the same order. |

**Example:**

Run the following code:

|  |
| --- |
| // string constructor  #include <iostream>  using namespace std;  int main ()  {  string empty;  string s0 ("Initial string");    string s1="1";  string s2 (s0);  string s3 (s0, 8, 3);  string s4 ("A character sequence");  string s5 ("Another character sequence", 12);  string s6a (10, 'x');  string s6b (10, 42); // 42 is the ASCII code for '\*'  string s7 (s0.begin(), s0.begin()+7);  cout << "s1: " << s1 << "\ns2: " << s2 << "\ns3: " << s3;  cout << "\ns4: " << s4 << "\ns5: " << s5 << "\ns6a: " << s6a;  cout << "\ns6b: " << s6b << "\ns7: " << s7 << '\n';  return 0;  } |

* + 1. **Overloaded Operators in the string class:**

C++ string class (in header <string>) overloads these operators to work on string objects:

* String comparison (==, !=, >, <, >=, <=): For example, you can use str1 == str2 to compare the contents of two string objects.
* Stream insertion and extraction (<<, >>): For example, you can use cout << str1 and cin >> str2 to output/input string objects.
* Strings concatenation (+, +=): For example, str1 + str2 concatenates two string objects to produce a new string object; str1 += str2 appends str2 into str1.
* Character indexing or subscripting []: For example, you can use str[*n*] to get the char at index *n*; or str[*n*] = c to modify the char at index *n*. Take note that [] operator does not perform index-bound check, i.e., you have to ensure that the index is within the bounds. To perform index-bound check, you can use string's at() member function.
* Assignment (=): For example, str1 = str2 assigns str2 into str1.

**Example:**

|  |
| --- |
| /\* Test overloaded operators in the C++ string class  (TestStringOverloadOperators.cpp) \*/  #include <iostream>  #include <iomanip>  #include <string> // needed to use the string class  using namespace std;    int main() {  string msg1("hello");  string msg2("HELLO");  string msg3("hello");    // Relational Operators (comparing the contents)  cout << boolalpha;  cout << (msg1 == msg2) << endl; // false  cout << (msg1 == msg3) << endl; // true  cout << (msg1 < msg2) << endl; // false (uppercases before lowercases)    // Assignment  string msg4 = msg1;  cout << msg4 << endl; // hello    // Concatenation  cout << (msg1 + " " + msg2) << endl; // hello HELLO  msg3 += msg2;  cout << msg3 << endl; // helloHELLO    // Indexing  cout << msg1[1] << endl; // 'e'  cout << msg1[99] << endl; // garbage (no index-bound check)  // cout << msg1.at(99) << endl; // out\_of\_range exception  } |

* + 1. **Strings Manipulation Functions:**

Some of the string manipulation functions are described below:

|  |  |
| --- | --- |
| **Function** | **Effect** |
| *S1.swap(s2)* | Exchanges the content of the container by the content of *str*, which is another [string](https://www.cplusplus.com/string) object. [Lengths](https://www.cplusplus.com/string::length) may differ. |
| *S1.Find(S2)*  *OR*  *S1.find(S2,pos)* | Searches the [string](https://www.cplusplus.com/string) for the first occurrence of the sequence specified by its arguments.  **\*** When *pos* is specified, the search only includes characters at or after position *pos*, ignoring any possible occurrences that include characters before *pos*. |
| *S1.rfind(s2)* | Searches the [string](https://www.cplusplus.com/string) for the last occurrence of the sequence specified by its arguments. |
| *S2=S1.substr(x1,x2)*  *OR*  *S2=S1.substr(x1)* | Returns a newly constructed [string](https://www.cplusplus.com/string) S2 object with its value initialized to a copy of a substring of this object i.e. S1. X1 represent the starting position of the first character to be copied and x2 is the no of characters to be copied after that.  \*If x2 is not mentioned, it will copy all characters till the  end. |
| *a.replace(pos,len,S2)* | Replaces the portion of the string that begins at character *pos* and spans *len* characters |
| *S1.len()* | Returns the length of the string S1. |
| *S1.insert(pos,S2)* | Inserts additional characters into the [string](https://www.cplusplus.com/string) S1 right before the character indicated by *pos.* |

**Exercise:**

1. The news that is spreading like wildfire on Whatsapp group is:

“*URGENT MSG: \'vaccinated people will die in two years\' claimed by Nobel Laureate Source: French Newspaper*”.

To stop the spread of this fake news, our task is to edit it. Store this sentence is a string and perform the following operations, so that the right news can reach the masses:

Find the length of this fake message.

Replace “Urgent” with “FAKE”.

Insert the name of persons, with whom the news is spreading i.e. “Luc Montagnier”. i.e.

“FAKE MSG: 'vaccinated people will die in two years' claimed by Nobel Laureate Luc Montagnier Source: French Newspaper”

Replace source name from “French newspaper” to “Whatsapp”.

Insert a sentence after that:

” Although the scientist is against vaccinations, but what is quoted on his name is the wrong information. His exact words were:” It is a massive mistake.”

To make the sentence more understandable for all readers, replace “it” with “COVID-19 Vaccination” i.e. Covid-19 vaccination is a massive mistake.

1. Your task is to create a class “mystring”, that behaves exactly like the string datatype. The prototype of this class is given below:

|  |
| --- |
| class mystring  {  char s1[100];  public:  //Define prototype of functions here  }  ////////////////////MAIN.CPP FILE////////////////////////  int main()  {  mystring s1(”Hello, how are you”);  mystring s2(”Hey, what are you doing?”);  mystring s3=s1+s2;  s1=s2;  if (s1!=s3)  cout<<”Strings are not equal\n”;  else  cout<<”Strings are equal\n”;  cout<<s3;  } |

Use header file for the implementation of this program and overload following operators:

Constructor

Addition

Extraction operator

Assignment operator

Not equal to operator

**Extra Task:**

Instead of declaring a character array of size 100 in mystring class, create a dynamic array whose size depend upon the string entered by the user. [Hint: Use pointers.]