**Using the Overloaded Operators of Standard Library Class string**

#include "stdafx.h"

#include <iostream>

#include <string>

using namespace std;

int main()

{

std::string s1{"happy"};

std::string s2{ " birthday" };

std::string s3; //creates an empty string

// test overloaded equality and relational operators

std::cout << "s1 is \"" << s1 << "\"; s2 is \"" << s2

<< "\"; s3 is \"" << s3 << '\"'

<< "\n\nThe results of comparing s2 and s1:" << std::boolalpha

<< "\ns2 == s1 yields " << (s2 == s1)

<< "\ns2 != s1 yields " << (s2 != s1)

<< "\ns2 > s1 yields " << (s2 > s1)

<< "\ns2 < s1 yields " << (s2 < s1)

<< "\ns2 >= s1 yields " << (s2 >= s1)

<< "\ns2 <= s1 yields " << (s2 <= s1);

// test string member function empty

std::cout << "\n\nTesting s3.empty(): \n";

if (s3.empty()) {

std::cout << "s3 is empty; assigning s1 to s3;\n";

s3 = s1; // assign s1 to s3

std::cout << "s3 is \"" << s3 << "\"";

}

// test overloaded string concatenation assignment operator

std::cout << "\n\ns1 += s2 yields s1 = ";

s1 += s2; // test overloaded concatenation

std::cout << s1;

// test string concatenation with a C string

std::cout << "\n\ns1 += \" to you\" yields\n";

s1 += " to you";

std::cout << "s1 = " << s1;

// test string concatenation with a C++14 string-object literal

std::cout << "\n\ns1 += \", have a great day!\"" << " yields\n";

s1 += ", have a great day!"s; // s after " for string-object literal

std::cout << "s1 = " << s1 << "\n\n";

// test string member function substr

std::cout << "The substring of s1 starting at location 0 for \n"

<< "14 characters, s1.substr(0, 14), is:\n"

<< s1.substr(0, 14) << "\n\n";

// test substr "to-end-of-string" option

std::cout << "The substring of s1 starting at\n"

<< "location 15, s1.substr(15), is:\n" << s1.substr(15) << "\n";

//test copy constructor

std::string s4{s1};

std::cout << "\ns4 = " << s4 << "\n\n";

// test overloaded copy assignment (=) operator with self-assignment

std::cout << "assigning s4 to s4\n";

s4 = s4;

std::cout << "s4 is " << s4;

// test using overloaded subscript operator to create lvalue

s1[0] = 'H';

s1[6] = 'B';

std::cout << "\n\ns1 after s1[0] = 'H' and s1[6] = 'B' is:\n"

<< s1 << "\n\n";

// test subscript out of range with string member function "at"

try {

std::cout << "Attempt to assign 'd' to s1.at(100) yields: \n";

s1.at(100) = 'd'; // ERROR: subscript out of range

}

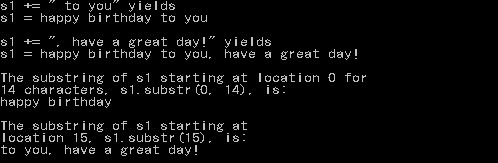
catch (std::out\_of\_range& ex) {

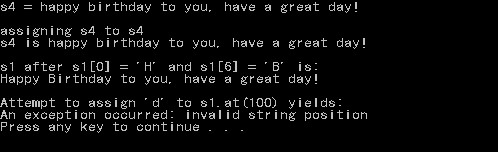
std::cout << "An exception occurred: " << ex.what() << std::endl;

}

}

**Result:**





**Important notes:**

* Look into string literal
* To put (“) inside a string, use (\”). Example:

std::cout << "\n\ns1 += \" to you\" yields\n";