Object Oriented Programming with C++

8. Strings

By: Prof. Pandav Patel

Second Semester, 2020-21
Computer Engineering Department
Dharmsinh Desai University

Strings

- strings in C++
  - Two ways of using string
    1. Old C style - Array of characters
    2. C++ string class - defined in <string> library

- C++ string class
  - Huge class with many constructors, operators and member functions
  - Commonly used constructors
    1. string(); - creates empty string
       e.g. string str1;
    2. string(const char *str); - creates string object from C style string
       e.g. char c_str[10] = "Name";
       string str2(c_str);
       string str3 = string(c_str);
    3. string(const string &str); - copy constructor
       e.g. string str4(str3);
       string str5 = string(str4);

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




int main()
{
	std::string str1;
	std::cout << "str1: " << str1 << std::endl;

	char c_str[10] = "Name";
	std::string str2(c_str);
	std::cout << "str2: " << str2 << std::endl;
	std::string str3 = std::string(c_str);
	std::cout << "str3: " << str3 << std::endl;

	std::string str4(str3);
	std::cout << "str4: " << str4 << std::endl;
	std::string str5 = std::string(str4);
	std::cout << "str5: " << str5 << std::endl;

	return 0;
}
```

```
str1:
str2: Name
str3: Name
str4: Name
str5: Name
```

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

using std::cin;
using std::cout;
using std::endl;
using std::string;

int main()
{
	string str1;
	cout << "str1: " << str1 << endl;

	char c_str[10] = "Name";
	string str2(c_str);
	cout << "str2: " << str2 << endl;
	string str3 = string(c_str);
	cout << "str3: " << str3 << endl;

	string str4(str3);
	cout << "str4: " << str4 << endl;
	string str5 = string(str4);
	cout << "str5: " << str5 << endl;

	return 0;
}
```

```
str1:
str2: Name
str3: Name
str4: Name
str5: Name
```

Strings

- C++ string class
  - Commonly used operators

1. =	Assignment
2. +	Concatenation
3. +=	append at the end
4. ==	equality
5. !=	inequality
6. <	less than
7. <=	less than or equal
8. >	greater than
9. >=	greater than or equal
10. [i]	get character at index i
11. <<	output
12. >>	input

  - String operators compare strings lexicographically

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

using std::cin;
using std::cout;
using std::endl;
using std::string;

int main()
{
        string str1("str1");
        string str2 = string("str2");

        cout << "str1: " << str1 << endl;
        str1 = str2; // string copy
        cout << "str1: " << str1 << endl;

        str1 = " World!"; // string copy
        // string concatenation
        str2 = "Hello" + str1;
        cout << "str2: " << str2 << endl;
        str2 += str1; // append string
        cout << "str2: " << str2 << endl;
```

```
str1: str1
str1: str2
str2: Hello World!
str2: Hello World! World!
Equal
Not Equal
A is less than a
AB is greater than A
A is less than b
l
```

```
        // string operators compare strings lexicographically
        str1 = "Hello";
        if(str1 == "Hello") { // string compare - true
                cout << "Equal" << endl;
        }
        if(str1 != "hello") { // string compare - true
                cout << "Not Equal" << endl;
        }

        if("A" < "a") // true
                cout << "A is less than a" << endl;
        if("AB" > "A") // true
                cout << "AB is greater than A" << endl;
        if("B" < "a") // false
                cout << "B is less than a" << endl;
        if("A" < "b") // true
                cout << "A is less than b" << endl;

        str1 = "Hello";
        // [] has been overloaded by string class
        // Hence characters in the string can be
        // accessed using [ ] like character array
        cout << str1[2] << endl;

        return 0;
}
```

Strings

- C++ string class
  - Commonly used methods
    1. length
    2. find
    3. substr

- getline function is independent function and is not method of class string. But it is part of string library.
  - getline function can scan line from the input into string object, syntax is as follow
    - getline(cin, string_object);
  - It scans spaces and tabs as well, stops only when it encounters newline character and it also consumes that newline character

```
#include<iostream>
#include<string>
int main()
{
	std::string str1("Hello");
	// returns length of the string
	std::cout << str1.length() << std::endl;
	// returns index of the first character of the first match
	// If no matches were found, the function returns string::npos
	// npos is a static member constant value with the greatest possible value for an element of type size_t
	std::cout << str1.find("ell") << std::endl;
	// string substr (size_t pos = 0, size_t len = npos) const;
	// pos
		// Position of the first character to be copied as a substring.
		// If this is equal to the string length, the function returns an empty string.
		// If this is greater than the string length, it throws out_of_range.
	// len
		// Number of characters to include in the substring
		// If there are less than len characters in string starting from pos,
		// then it will return string with all characters starting from pos to end of string
		// A value of string::npos indicates all characters until the end of the string.
	// Returns a newly constructed string object with its value initialized to a copy of a substring of this object.
	std::cout << str1.substr( 1, 3) << std::endl;

	return 0;
}
```

```
5
1
ell
```

```
#include<iostream>
#include<string>
int main()
{
        // string can contain white space characters too
        // cout prints whole string including white space characters
        // In below statement, assignment of character array to string object is possible
        // because of converting constructor of string class which takes char * as the only argument.
        // A constructor that is not declared with the keyword explicit(To perform explicit casting) and which can be
        called
        // with a single parameter (until C++11) is called a converting constructor.
        std::string str6 = "New Name";
        std::cout << "str6: " << str6 << std::endl;

        // by default, cin stops at the occurrence of white space character
        std::cin >> str6;
        std::cout << "str6: " << str6 << std::endl;

        // getline consumes newline at the end of the line
        // care should be taken that first character in the input buffer is not newline while calling getline
        // otherwise getline will consume that new line character and will not read any further
        std::getline(std::cin, str6);
        std::cout << "str6: " << str6 << std::endl;
        std::getline(std::cin, str6);
        std::cout << "str6: " << str6 << std::endl;
        return 0;
}
```

```
str6: New Name
Pandav Patel
str6: Pandav
str6:  Patel
Roshan Patel
str6: Roshan Patel
```

```
#include<iostream>

class Test{
        int i;
public:
        Test() { }
        // Constructor has not been declared with explicit keyword
        // So it is a converting constructor
        // Hence it is calling this constructor when we try to assign in to its object
        // As this converting constructor can be called with one argument
        Test(int i){
                std::cout << "Constructor called" << i << std::endl;
                this->i = i;
        }
        void print(){
                std::cout << i << std::endl;
        }
};

int main(){
        Test t = 3;
        t.print();

        return 0;
}
```

```
Constructor called3
3
```

Interesting reads

- Converting Constructor
  - https://en.cppreference.com/w/cpp/language/converting_constructor
  - https://stackoverflow.com/questions/63299964/why-am-i-allowed-to-initialize-object-of-class-test-by-assigning-integer-constan/63302699#63302699