

C++ strings

In C, strings are considered as array of character where in C++, strings are regarded as an object. With this new paradigm, strings in C++ can be very powerful since string object provides a set of useful function to modify the string easily.

Creating and Initializing C++ strings

It's fairly easy to create strings in C++, below is the sample how to create and initialize strings. Don't forget to put **#include <string>** header.

Source Code

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

using namespace std;

int main()
{
    string blank;
    string s1 = "First way";
    string s2("Second way");

    cout << blank << endl;
    cout << s1 << endl;
    cout << s2 << endl;

    return 0;
}
```

Output

```
First way
Second way
```

Copying and Concatenating strings

With strings object it's very easy to copy and concatenate strings. It's also introduced how to use the string object's **substr ()** member function to create a substring. Below is the code sample.

Source Code

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

using namespace std;

int main()
{
    // copy whole string
    string s1 = "C++ strings";
    string s2 = s1;
    string s3(s1);

    // copy substring
    string sub1(s1,0,3);
    string sub2 = s1.substr(4);

    // string concatenation
    string concat1 = "I like " + s1.substr(0,3);
    string concat2 = "I like ";
    concat2 += s1;

    // output
    cout << s2 << endl;
    cout << s3 << endl;
    cout << sub1 << endl;
    cout << sub2 << endl;
    cout << concat1 << endl;
    cout << concat2 << endl;

    return 0;
}
```

Output

```
C++ strings
C++ strings
C++
strings
I like C++
I like C++ strings
```

Note that **substr()** is overloaded.

```
substr(fromIndex, toIndex);
substr(fromIndex);
```

The first one gets the substring from index *fromIndex* to *toIndex - 1*. The second gets substring from *fromIndex* until *the end of the string*.

Filling strings with Characters

We can fill a string with n numbers of character. Below is the sample of filling string with character.

Source Code

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

using namespace std;

int main()
{
    string mask(5, '*');
    cout << mask << endl;

    return 0;
}
```

Output

```
*****
```

Additional note :

We can get the i-th position character of the string using index in brackets.

```
string s = "get character";
s[0] would be 'g'
s[4] would be 'c'
```

More on string Operations

There are other string operations which are commonly used, such as append, insert, replace, etc. We will discuss them one by one. One of the C++ strings advantage is that the strings grow as they need. There are several functions to monitor and manage the strings' size

size();	returns number of char stored
length();	identical to size();
capacity();	returns the size of current allocation
reserve();	specify certain amount for future usage
resize();	resize the strings' size

Source code

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

using namespace std;

int main()
{
    string s = "C++ strings";

    cout << "Initial Condition" << endl;
    cout << "Size      : " << s.size() << endl;
    cout << "Capacity : " << s.capacity() << endl;

    s.reserve(100);

    cout << "After reserve" << endl;
    cout << "Size      : " << s.size() << endl;
    cout << "Capacity : " << s.capacity() << endl;

    s.resize(20);

    cout << "After resize" << endl;
    cout << "Size      : " << s.size() << endl;
    cout << "Capacity : " << s.capacity() << endl;

    return 0;
}
```

Output

```
Initial Condition
Size      : 11
Capacity : 31
After reserve
Size      : 11
Capacity : 127
After resize
Size      : 20
Capacity : 127
```

Append, Insert, Find and Replace

There are quite a number of overloaded versions of these operations. We will demonstrate the relatively more common one. The function **find()** will return **string::npos** if it doesn't find the desired string.

Source Code

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

using namespace std;

int main()
{
    string s    = "C++";
    string tag  = "good";

    // append
    s.append(" is a good language");
    cout << s << endl;

    // insert
    s.insert(14, "programming ");
    cout << s << endl;

    // find and replace
    int start = s.find(tag);
    s.replace(start, tag.size(), "powerful");
    cout << s << endl;

    return 0;
}
```

Output

```
C++ is a good language
C++ is a good programming language
C++ is a powerful programming language
```

Find and replace is a common operation that we usually use in text editing. Below we demonstrate how to replace all the occurrence of the word with a given word.

Source Code

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

using namespace std;

int main()
{
    string s          = "All this will be replaced by this";
    string toFind     = "this";
    string replaceWith = "that";

    // before find and replace
    cout << s << endl;
```

```
// find and replace operation
int pos;
while ((pos = s.find(toFind)) != string::npos)
    s.replace(pos,toFind.size(),replaceWith);

// after find and replace
cout << s << endl;

return 0;
}
```

Output

```
All this will be replaced by this
All that will be replaced by that
```