String Class

The string class is used to store sequence of characters. It is used in place of character arrays. It is beneficial to use this class, especially when the string is huge, due to the various functions that it supports.

Header file
To use this class, we need to include this as a header file in our program
```
#include<string>
```

Creating String Objects / Variables
String objects are created as:
```
string str1, str2 = "Hello";
```
In this example, we have defined two variables '**str1**', and '**str2**' of type '**string**'. '**str1**' is not initialized and will not contain anything and '**str2**' is initialized to '**Hello**'. These can be printed using the '**cout**' statement i.e. '`cout<< "str1 = " << str1 << endl << "str2 =" << str2;`'.
This will print:
str1 =
str2 = **Hello**

The string can be taken as input from the user using 'getline'.
string str;
getline(cin, str); or cin>>str but this stops taking input at space

Given below is a list of functions of this class. Assume that all the header files are included and the objects(used in the table below) of string class have been already created. Henceforth, string object will be referred to as '**s_obj**' for describing the syntax.

No.	Function	Description

1 String Assignment

a) s_obj = (const char *s)
b) s_obj = (char c)
c) s_obj2 = s_obj1
This functions assigns a value to the string object. The previous content present would be replaced with the new one. The 's_obj' will contain the string on successful execution
E.g.
```
string str1, str2, str3;
char c='F';
str1="Hello ";
str2=c;
str3 = str1;
cout<<str1<<str2<<str3<<endl;
```
Output:
Hello FHello

Constructors

2 Empty Constructor

string s_obj;
This is an empty constructor that creates a string of 0 characters.
E.g.
```
string str1;
```

3 Copy Constructor

string s_obj2(s_obj1);

This constructor creates a copy of s_obj1 and stores in s_obj2.

E.g.

```
string str1="hello", str2;
str2=str1;
cout<<"str="<<str1<<" str2="<<str2<<endl;
```

Output

str=hello str2=hello

4 Substring Constructor

string s_obj2(s_obj1, int pos, int len);

Copies the string 'len' characters from position 'pos' in the string 's_obj1' to 's_obj2'

E.g.

```
string str1 ("Learning strings");
string str2 (str1, 5, 3);
cout << "str2 = " << str2<<endl;
```

Output

str2 = ing

5 Fill Constructor

string s_obj(int size, char c)

As the name suggests, this constructor fills up the string 's_obj' with 'int size' copies of 'char c'

E.g.

```
string str1 (7, 'C');
cout<<"str1 = " << str1;
```

Output

str1 = CCCCCCC

Iterators

Iterator is an object which points to a character in the string i.e. a container. It allows the programmers to traverse throught the string i.e. a container using different functions.

7 begin

string::iterator it_name = s_obj.begin();

The iterator 'begin' returns the iterator that points to the first character in the specified string object.

E.g.

```
string str1 ("Hello World");
string::iterator str2 = str1.begin();
cout<<"str2 = "<<*str2<<endl;
```

Output

str2 = H

8 end

string::iterator it_name = s_obj.end();

The iterator 'end' returns the iterator that points to the next location after the end of string. For example if my string is “Hello World”, this iterator will point to the location after the last character 'd' in the string. To access the last character directly, you must first use this iterator and then decrement the iterator by 1.

E.g.

```
string str1 ("Hello World");
string::iterator str2=str1.end();
str2--;
cout<<"str2 = "<<*str2<<endl;
```

Output

str2 = d

9 rbegin

string::reverse_iterator it_name = s_obj.rbgein();

'rbegin' will return a 'reverse_iterator' that points to the last character in the string. This iterator increments backwards. For example, if the last character is at position 10, 'reverse_interator' will return 10. If this was incremented by 1, instead of pointing at 11 position, it will point to 9. This will be incremented till the reverse_iterator points to beginning of the string.

E.g.

```
string str1 ("Desserts");
string::reverse_iterator str2 = str1.rbegin();
cout<<"str2 = "<<*str2<<endl;
str2++;
cout<<"str2 = "<<*str2<<endl;
str2++;
cout<<"str2 = "<<*str2<<endl;
```

Output

str2 = s
str2 = t
str2 = r

10 rend

string::reverse_iterator it_name = s_obj.rend();

'rend' will return a 'reverse_iterator' that points to the previous location of the first character in the string. To access all the characters of the string from the first character till the end, we need to decrement the 'reverse_iterator'. For example, 'rend' will initially point to the location preceeding the first character. If we want to access the first character, we will decrment the 'reverse_iterator' by 1. Now, since the 'reverse_iterator' is pointing to first character, if we again decrement by 1, we will now point to the second character.

E.g.

```
string str1 ("Desserts");
string::reverse_iterator str2 = str1.rend();
cout<<"str2 = "<<*str2<<endl;
str2--;
cout<<"str2 = "<<*str2<<endl;
str2--;
cout<<"str2 = "<<*str2<<endl;
```

Output

str2 =
str2 = D
str2 = e

Capacity

11 Size / length

size_t s_obj.size()

size_t s_obj.length()

The length() and size() return the length of the string in bytes.

E.g.

```
string str1 ("Tic Tac");
cout<<"Size of str1 using size = " << str1.size() << endl;
cout<<"Size of str1 using length = " << str1.length() << endl;
```

Output

Size of str1 using size = 7
Size of str1 using length = 7

12 capacity

size_t s_obj.capacity()

This returns the currently allocated space for strings in bytes. It may or may not be equal to the length of the string. It might have some extra space for operations to be performed at a later stage.

E.g.

```
string str1 ("Tic Tac");
cout<<"Capacity of str1 = " << str1.capacity()<<endl;
```

Output

Capacity of str1 = 7

13 clear

void s_obj.clear()

This clears the content of the string object.

E.g.

```
string str1 ("Tic Tac");
cout<<"str1 = " << str1 << endl;
str1.clear();
cout<<"str1 = " << str1 << endl;
```

Output

str1 = Tic Tac
str1 =

14 empty

bool s_obj.empty()

This returns whether the string is empty or not. A string is called as empty when it has 0 characters. It returns true if it is empty, else it returns false.

E.g.

```
string str1 ("Tic Tac");
string str2;
cout<< "Is str1 empty? " << str1.empty() << endl;
cout<< "Is str2 empty? " << str2.empty() << endl;
```

Output

Is str1 empty? 0
Is str2 empty? 1

Element Access

15 Operator[ ]

s_obj[int pos];

The positon 'pos' is used to access the character at that position in the string.

E.g.

```
string str1 ("Learning");
cout << str1[0] << str1[2] << str1[4] << str1[6] << endl;
```

Output

Lann

16 at

s_obj.at(int pos);

This returns the reference to the character at the position 'pos'. It is used to access the character at that position in the string

E.g.

```
string str1 ("Learning");
cout<< str1.at(0) << str1.at(2) << str1.at(4) << str1.at(6) << endl;
```

Output

Lann

Modifiers

17 append

a) s_obj.append(s_obj2); or s1 = s1+s2;
b) s_obj.append(const char*s)
c) s_obj.append(const char *s, size_t n)
d) s_obj.append(s_obj2, size_t pos, size_t len)
(a) This appends the string mentioned in s_obj2 to s_obj
(b) This appends the string to s_obj
(c) This appends first 'n' characters of string 's' to s_obj
(d) This appends
E.g.

```
string str1 ("Learning");
string str2 ("C++");
string str3 ("this is useful info");
str1.append(str2);
str1.append(" is fun");
str1.append(".isn't it????????",10);
str1.append(str3,8,3);
cout<<"str1 = "<<str1<<endl;
```

Output
str1 = LearningC++ is fun.isn't it?use

18 erase

a) s_obj.erase(size_t pos, size_t len)
b) s_obj.erase(cons_iterator it)
c) s_obj.erase(cons_iterator it1,
const_iterator it2)
(a) This erases 'len' number of characters from position 'pos'
(b) This erases the character pointed by iterator 'it'
(c) This erases the sequence of characters ranging from iterator 'it1' to iterator 'it2'.
E.g.

```
string str1 ("ThisIsC++StringConcept");
cout<<"Origintal String: "<<str1<<endl;
str1.erase(5,3);
cout<<"Erased 3 characters from 5th position: "<<str1<<endl;
str1.erase(str1.begin()+3);
cout<<"Erased 4th character (i.e. position 3): "<< str1<<endl;
str1.erase(str1.begin()+2,str1.end()-2);
cout<<"Erased characters from 2nd position to 2nd last position: "
                                                    <<str1<<endl;
```

Output
Origintal String: ThisIsC++StringConcept
Erased 3 characters from 5th position: ThisI+StringConcept
Erased 4th character (i.e. position 3): ThiI+StringConcept
Erased characters from 2nd position to 2nd last position: Thpt

String Operations

19 copy

str.copy(char *s, len, pos)

Copies 'len' characters from string object 'str' to the character array '*s' from position 'pos'.

E.g.

```
char str1[] = {'a','b','c','d','e','f','g','h'};
string str2("World");
str2.copy(str1,3,1);
cout<<"str1 = "<<str1<<endl;
```

Output

str1 = orldefgh

20 find

a) s_obj.find(s_obj2)

b) s_obj.find(char c)

(a) This finds whether the string given in 's_obj2' is present in the string 's_obj'.

(b) This finds whether the character 'c' is present in the string 's_obj'.

If the string / character is found, it returns the position (first occurrence) at which it was found, else it returns string::npos.

string::npos is an invalid value that can never represent the index.

E.g.

```
int position;
string str1 ("hello world");
string str2 ("world");
position=str1.find(str2);
cout<<"world found at "<<position<<endl;
position=str1.find('o');
cout<<"first occurrence of o is at " << position<<endl;
```

Output

world found at 6

first occurrence of o is at 4

For more details, please refer to the following reference links:

http://www.cplusplus.com/reference

http://en.wikipedia.org/wiki/C++_Standard_Library