C++
3. string

-Pandav Patel

Introduction

- There are two types of strings in C++
  - Old C style string (array of char)
    - **char str[10] = “Hero”;**
  - New C++ style string (Object of string class)
    - **string str = “Hero”;**
    - Must include string - **#include<string>**
    - string class is defined within **std** namespace
    - So it should be used as **std::string** unless it included in scope with use of **using** statement
    - In next slides we will assume that string header is included and it has been added to current scope with use of using statement - so we will use **string** instead of **std::string**
- We should **prefer new C++ style string** whenever feasible
  - No need to worry about size of the string
  - Overloaded operators and string class methods make it easier to work with

Declaration, initialization and concatenation

```
string str1 = "DDU ";
string str2 = "Nadiad";

string str = str1 + str2;
cout << str << endl;

str1 = "DDU";
str = str1 + " " + str2;
cout << str << endl;

str = str1 + ' ' + str2;
cout << str << endl;
```

OUTPUT:

DDU Nadiad
DDU Nadiad
DDU Nadiad

append method

```
string str1 = "DDU ";
string str2 = "Nadiad";

string str = str1.append(str2); // This changes str1
// string str = str1 + str2; // This does not change str1

cout << str1 << endl;
cout << str2 << endl;
cout << str << endl;
```

OUTPUT:

```
DDU Nadiad
Nadiad
DDU Nadiad
```

Numbers and strings

```
string str1 = "10";
string str2 = "20";

string str = str1 + str2;
cout << str << endl;

str = str1 + "20";
cout << str << endl;

// error: no match for 'operator+'
// str = str1 + 20;
// cout << str << endl;
```

OUTPUT:

```
1020
1020
```

Determining length of string

- Both **size()** and **length()** methods return length of the string that is the number of characters stored in the string

```
string str1 = "DDU";
cout << "Lenght is: " << str1.length() << endl;
cout << "Size is: " << str1.size() << endl;
```

OUTPUT:

```
Lenght is: 3
Size is: 3
```

Copy and compare

```
string str1 = "I am a string.";
string str2;
str2 = str1;
string str3 = "I am a string.";

if(str1 == str2)
    cout << "str1 and str2 are same." << endl;
if(str2 == str3)
    cout << "str2 and str3 are same." << endl;
if(str1 == str3)
    cout << "str1 and str3 are same." << endl;

str1[0] = 'i';

if(str1 == str2)
    cout << "str1 and str2 are same." << endl;
if(str2 == str3)
    cout << "str2 and str3 are same." << endl;
if(str1 == str3)
    cout << "str1 and str3 are same." << endl;
```

OUTPUT:

```
str1 and str2 are same.
str2 and str3 are same.
str1 and str3 are same.
str2 and str3 are same.
```

Input

```
string str1;

cin >> str1; // Stops scanning on whitespace
cout << str1;
```

INPUT:
Hello world

OUTPUT:
Hello

Input line

```
string str1;

getline(cin, str1);
cout << str1;
```

INPUT:
Hello world

OUTPUT:
Hello world

Input line and number - I

```
int i;
string str1;

cin >> i;      // Leaves newline in the input

// Finds newline as first character
// Ends further scanning and removes newline char
getline(cin, str1);

cout << i << str1;
```

INPUT:
10
Hello World

OUTPUT:
10

Input line and number - II

```
int i;
string str1;

// Removes whitespaces at the beginning of input
// before scanning into variables
cin >> i >> str1;

cout << i << str1;
```

INPUT:
10
Hello World

OUTPUT:
10Hello

Input line and number - III

```
int i;
string str1;

cin >> i;      // Leaves newline in the input

// cin >> ws removes all the whitespaces
// at the beginning of the input
getline(cin >> ws, str1);

cout << i << str1;
```

INPUT:
10
Hello World

OUTPUT:
10Hello World

Input line and number - IV

```
int i;
string str1, str2;

cin >> i;      // Leaves newline in the input

// Finds newline as first character
// Ends further scanning and removes newline char
getline(cin, str1);

// Scans Hello World from second line
getline(cin, str2);

cout << i << str1 << str2;
```

INPUT:
10
Hello World

OUTPUT:
10Hello world

Input line and number - V

```
int i = 1000, j = 2000;
string str1;

cin >> str1 >> i >> j;

cout << i << " " << j << " "  << str1;

// cin is in failed state so this will not scan into str1
cin >> str1;
cout << endl << str1;
```

INPUT:
10
Hello World

OUTPUT:
0 2000 10
10

Input line and number - VI

```
int i;

// This condition will be true until cin is not in fail state.
// cin will go to fail state once it can not scan into int i.
// may be because of end of file or non number in the input.
while(cin >> i) // scans into i and condition is true if cin does not go to fail state
    cout << i << " ";

// We can not continue to use cin unless we fix it. How to fix it?
cin.clear(); //cin.clear() clears fail state of cin

string temp;
while(true) {
    cin >> temp;
    if(cin.fail())  // cin.fail() returns true if cin is in fail state
        break;
    cout << temp << " ";
}
```

INPUT:
10
20
30
He
llo
[ctrl+d]

OUTPUT:
10 20 30 He llo

Range for - I

```
string str1;

getline(cin >> ws, str1);

for(auto c: str1) {
    if(islower(c))
        c = toupper(c); // This will not change original string str1
    cout << c;
}

cout << endl << str1;
```

INPUT:
Hello

OUTPUT:
HELLO
Hello

Range for - II

```
string str1;

getline(cin >> ws, str1);

for(auto &c: str1) { // c is ref now
    if(islower(c))
        c = toupper(c); // This will change original string str1
    cout << c;
}

cout << endl << str1;
```

INPUT:
Hello

OUTPUT:
HELLO
HELLO