Practical Lecture : String

Quick Recap

Let’s take a quick recap of previous lecture –

A)

B)

C)

D)

E)

Today’s

Today we are going to cover -

- String Class
- String Operation
- Coding Question

Let’s Get Started-

String

C++ has in its definition a way to represent **sequence of characters as an object of class**. This class is called std:: string. String class stores the characters as a sequence of bytes with a functionality of allowing **access to single byte character**.

String vs Char Array

- A character array is simply an array of characters can terminated by a null character. A string is a class which defines objects that be represented as stream of characters.

- Size of the character array has to allocated statically, more memory cannot be allocated at run time if required. Unused allocated memory is wasted in case of character array. In case of strings, memory is allocated dynamically. More memory can be allocated at run time on demand. As no memory is preallocated, no memory is wasted

- There is a threat of array decay in case of character array. As strings are represented as objects, no array decay occurs.

String vs Char Array

- Implementation of character array is faster than std:: string. Strings are slower when compared to implementation than character array.

- Character array do not offer much inbuilt functions to manipulate strings. String class defines a number of functionalities which allow manifold operations on strings.

Operation on String

- **getline() :-** This function is used to store a stream of characters as entered by the user in the object memory.

- **push_back() :-** This function is used to input a character at the end of the string.

- **pop_back() :-** Introduced from C++11(for strings), this function is used to delete the last character from the string

Operation on String

```
#include<iostream>
#include<string>
using namespace std;
int main()
{
    string str;
    getline(cin,str);
    cout << "The initial string is : ";
    cout << str << endl;
    str.push_back('s');
    cout << "The string after push_back operation is : ";
    cout << str << endl;
    str.pop_back();
```

Operation on String

```
    cout << "The string after pop_back operation is : ";
    cout << str << endl;

    return 0;

}
```

Input:-

```
upgrad
```

Output:-

```
The initial string is : upgrad
The string after push_back operation is : upgrads
The string after pop_back operation is : upgrad
```

Operation on String

4. capacity() :- This function returns the capacity allocated to the string, which can be equal to or more than the size of the string. Additional space is allocated so that when the new characters are added to the string, the operations can be done efficiently.

5. resize() :- This function changes the size of string, the size can be increased or decreased.

6.length():-This function finds the length of the string

7.shrink_to_fit() :- This function decreases the capacity of the string and makes it equal to the minimum capacity of the string. This operation is useful to save additional memory if we are sure that no further addition of characters have to be made.

Operation on String

```
#include<iostream>
#include<string> // for string class
using namespace std;
int main()
{
    string str = "learn and grow and learn fast";

    cout << "The initial string is : ";
    cout << str << endl;
    str.resize(13);
    cout << "The string after resize operation is : ";
    cout << str << endl;
    cout << "The capacity of string is : ";
    cout << str.capacity() << endl;
    cout<<"The length of the string is :"<<str.length()<<endl;
```

Operation on String

```
  str.shrink_to_fit();
    cout << "The new capacity after shrinking is : ";
    cout << str.capacity() << endl;

    return 0;

}
```

Operation on String

8. begin() :- This function returns an iterator to beginning of the string.

9. end() :- This function returns an iterator to end of the string.

10. rbegin() :- This function returns a reverse iterator pointing at
the end of string.

11. rend() :- This function returns a reverse iterator pointing
at beginning of string.

Operation on String

```
#include<iostream>
#include<string> using namespace std;
int main()
{
    string str = "learnandgrow";
    std::string::iterator it;
    std::string::reverse_iterator it1;
    cout << "The string using forward iterators is : ";
    for (it=str.begin(); it!=str.end(); it++)
    cout << *it;
    cout << endl;
    cout << "The reverse string using reverse iterators is : ";
    for (it1=str.rbegin(); it1!=str.rend(); it1++)
    cout << *it1;
    cout << endl;
    return 0;  }
```

Operation on String

12. copy("char array", len, pos) :- This function copies the substring in target character array mentioned in its arguments. It takes 3 arguments, target char array, length to be copied and starting position in string to start copying.

13. swap() :- This function swaps one string with other.

Operation on String

```
#include<iostream>
#include<string>
using namespace std;
int main()
{
    string str1 = "learn grow and explore";
    string str2 = "learn grow";
    char ch[80];
    str1.copy(ch,13,0);
    cout << "The new copied character array is : ";
    cout << ch << endl << endl;
    cout << "The 1st string before swapping is : ";
    cout << str1 << endl;
    cout << "The 2nd string before swapping is : ";
```

Operation on String

```
    cout << str2 << endl;
    str1.swap(str2);
    cout << "The 1st string after swapping is : ";
    cout << str1 << endl;
    cout << "The 2nd string after swapping is : ";
    cout << str2 << endl;
    return 0;

}
```

Operation on String

```
    cout << str2 << endl;
    str1.swap(str2);
    cout << "The 1st string after swapping is : ";
    cout << str1 << endl;
    cout << "The 2nd string after swapping is : ";
    cout << str2 << endl;
    return 0;

}
```

Compare String

compare(string_to_compare) :- It is used to compare two strings. It returns the difference of second string and first string in integer.

```
#include<iostream>
#include<string>
using namespace std;
int main()
{
    string str("LearnAndGrow");
    string str1("LearnAndGrow");
    if ( str.compare(str1) == 0 )
        cout << "Strings are equal";
    else
        cout << "Strings are unequal";
    return 0;
}
```

String Operation

- **find(“string”):** Searches the string for the first occurrence of the substring specified in arguments. It returns the position of the first occurrence of substring.

- **find_first_of(“string”):** Searches the string for the first character that matches any of the characters specified in its arguments. It returns the position of the first character that matches.

- **find_last_of(“string”):** Searches the string for the last character that matches any of the characters specified in its arguments. It returns the position of the last character that matches.

- **rfind(“string”):** Searches the string for the last occurrence of the substring specified in arguments. It returns the position of the last occurrence of substring

String Operation

```
#include<iostream>
#include<string>
using namespace std;
int main()
{
    string str("Learn and Learn very fast");

    cout << "First occurrence of \"Learn\" starts from : ";
    cout << str.find("Learn") << endl;


    cout << "First occurrence of character from \"arn\" is at : ";
    cout << str.find_first_of("arn") << endl;
```

String Operation

```
  cout << "Last occurrence of character from \"arn\" is at : ";
    cout << str.find_last_of("arn") << endl;

    cout << "Last occurrence of \"Learn\" starts from : ";
    cout << str.rfind("Learn") << endl;

    return 0;

}
```

String Operation

```
  cout << "Last occurrence of character from \"arn\" is at : ";
    cout << str.find_last_of("arn") << endl;

    cout << "Last occurrence of \"Learn\" starts from : ";
    cout << str.rfind("Learn") << endl;

    return 0;

}
```

Practice Question

1. Write a C++ program to reverse a given string.
   Example:
   Sample Input: upgrad
   Sample Output: dargup

2.Write a C++ program to capitalize the first letter of each word of a given string. Words must be separated by only one space.
   Example:
   Sample Input: learn and grow
   Sample Output: Learn And Grow

3.Write a C++ program to count all the vowels in a given string.
   Example:
   Sample Input: eagerer
   Sample output: number of vowels -> 4

Any Questions ??

Thank You!

See you guys in next class.