TÌM HIỂU LỚP CỦA THƯ VIỆN CHUẨN STL

I. String

1. Đặc điểm

- String là một chuỗi sâu gồm một dãy ký tự. Lớp string lưu các ký tự dưới dạng một chuỗi các byte và cho phép truy cập vào từng byte một của chuỗi.

- Tổ chức các ký tự theo kiểu luồng (stream).

- Khác với chuỗi char và thông dụng hơn chuỗi char.

2. Thuộc tính

- Là các luồng ký tự.

3. Phương thức

- Nhập, gán (khởi tạo), xuất một chuỗi ký tự có khoảng trắng hoặc không khoảng trắng.

+ **string str ="";**

+ **string str("");**

+ Nhập chuỗi: **getline(cin, str);**

- Lấy length của chuỗi, lấy ký tự đầu, ký tự cuối của chuỗi, ký tự tại vị trí bất kỳ.

+ Lấy phần tử đầu chuỗi: **str.front();**

+ Lấy phần tử cuối chuỗi: **str.back();**

+ Lấy phần tử tại vị trí i: **str.at(i-1);**

- Thay dổi kích thước chuỗi.

+ **str.resize(n);**

- Ghép chuỗi 2 cho chuỗi 1, thay thế nội dung chuỗi 2 cho chuỗi 1.

+ Ghép str2 cho str: **str.append(str2);**

+ Thay thế nội dung str2 cho str: **str.assign(str2);**

- Xóa hoàn toàn một chuỗi, đẩy ký tự vào, ra khỏi chuỗi.

+ Xóa chuỗi: **str.erase();**

+ Đẩy ký tự s vào chuỗi vào cuối chuỗi: **str.push_back('s');**

+ Đẩy ký tự cuối chuỗi ra khỏi chuỗi: **str.pop_back();**

- So sánh hai chuỗi.

+ So sánh số ký tự chuỗi str với str2: **str.compare(str2);**

- Chèn chuỗi vào vị trí mong muốn.

+ Thêm str2 vào vị trí 0 của str: **str.insert(0, str2);**

+ Thêm 4 ký tự str2 bắt đầu từ ký tự 3 thêm vào vị trí (i+1) của str:

str(i, str2, 3, 4);

- Tìm kiếm trong string.

+ Tìm vị trí tồn tại của str2 trong str: **str.find(str2);**

- Trả về kích thước của không gian lưu trữ hiện được phân bổ cho chuỗi.

+ **str.capacity();**

- Copy chuỗi con trong string

+ copy chuỗi char ch chứa 10 phần tử đầu trong str: **str.copy(ch, 10, 0);**

- Swap string 1 và string 2

+ **str1.swap(str2);**

4. Cách sử dụng

a. Nhập, gán (khởi tạo) xuất một chuỗi ký tự có khoảng trắng hoặc không khoảng trắng.

```
string str("Phuong phap lap trinh huong doi tuong");
```

```
    string str = "Phuong phap lap trinh huong doi tuong"; //gán chuỗi
    getline(cin,str); //nhập chuỗi có ký tự space
    cin >> str; //nhập chuỗi không có ký tự space
    cout << str; //xuất chuỗi
```

b. *Lấy length của chuỗi, lấy ký tự đầu, ký tự cuối của chuỗi, ký tự tại vị trí bất kỳ.*

- Lấy độ dài chuỗi.

```
    int length = str.length();
    //Ngoài ra có thể dùng str.size() để lấy kích thước chuỗi, str.max_size để lấy kích thước lớn nhất của chuỗi
```

```
Demo:
    string str("1234231");
    cout << "\nLength is: " << str.length();
    cout << "\nSize is: " << str.size();
    cout << "\nMax size is: " << str.max_size();
```

```
Output:
Length is: 7
Size is: 7
Max size is: 9223372036854775807
```

- Lấy ký tự đầu chuỗi.

```
    char ch1 = str.front();//với chuỗi "toi khong thich oop", ký tự được lấy ra là: t
```

- Lấy ký tự cuối chuỗi.

```
    char ch2 = str.back();//cũng với ví dụ trên, ký tự được lấy ra là: p
```

- Lấy ký tự tại vị trí bất kỳ

```
    char ch3 = str.at(2);//lấy ví dụ trên, ký tự được lấy ra là ký tự ở vị trí thứ 2 + 1 của chuỗi, kết quả in ra là: i + 1
```

c. *Thay đổi kích thước chuỗi*

```
    string str("1234231");
    cout << "\nSize is: " << str.size();
    str.resize(13);
    cout << "\nAfter resizing, size is: " << str.size();
```

```
Output:
Size is: 7
After resizing, size is: 13
```

d. Ghép chuỗi 2 cho chuỗi 1, thay thế nội dung chuỗi 2 cho chuỗi 1.

- Ghép chuỗi 2 cho chuỗi 1

```
    string str = "Toi khong thich ";
    string str2 = "phuong phap lap trinh huong doi tuong";
    cout << "\nChuoi duoc in ra: " << str.append(str2);
```

```
Output:
Chuoi duoc in ra: Toi khong thich phuong phap lap trinh huong doi
tuong
```

- Thay thế nội dung chuỗi 2 cho chuỗi 1

```
    string str = "Toi khong thich ";
    string str2 = "phuong phap lap trinh huong doi tuong";

    cout << "Ket qua chuoi 1 sau khi duoc thay the: " <<
str.assign(str2);
    cout << "\nCap nhat chuoi 1: " << str;
```

```
Output:
Ket qua chuoi 1 sau khi duoc thay the: phuong phap lap trinh huong
doi tuong
Cap nhat chuoi 1: phuong phap lap trinh huong doi tuong
```

e. Xóa hoàn toàn một chuỗi, đẩy ký tự vào, ra khỏi chuỗi.

- Xóa hoàn toàn 1 chuỗi

```
    str.erase(); //nội dung của chuỗi str sẽ được xóa hết
```

- Đẩy ký tự vào chuỗi

```
    string str = "egg";
    str.push_back('S');
    cout << str;
```

```
Output:
eggS
```

- Đẩy ký tự ra khỏi chuỗi

```
    string str = "hello world!";
    str.pop_back();
    cout << str << '\n';
```

```
Output:
Hello world
```

f. So sánh hai chuỗi.

```
    str1.compare(str2);//so sánh số ký tự của chuỗi str1 với chuỗi str2
```

g. Chèn chuỗi vào vị trí mong muốn.

```
     string str1 = "OOP";
     string str2 = "Toi khong thich ";


     str1.insert(0, str2); //thêm chuỗi str2 vào vị trí 0 của chuỗi str1
     cout << str1;
```

```
Output:
Toi khong thich OOP
```

```
    string str1 = "OOP";
    string str2 = "Toi khong thich ";


    str1.insert(2, str2, 3, 4); // lấy 4 ký tự bắt đầu từ ký tự thứ 3 của chuỗi str2 chèn vào vị trí thứ (2+1) của chuỗi str1
```

```
    cout << str1;
```

```
Output:
OO khoP
```

h. Tìm kiếm trong string.

```
    string str1 = "Toi thich OOP";
    string str2 = "OOP";


    cout << "\nPhat hien str2 xuat hien trong chuoi str1 tai vi
tri: " << str1.find(str2) + 1;
    //tìm vị trí chuỗi str2 trong str1, nếu str2 không tồn tại
trong str 1 return 0;
```

```
Output:
Phat hien str2 xuat hien trong chuoi str1 tai vi tri: 11
```

- Ngoài ra còn có hàm lấy find_first_not_of(string) và find_last_not_of(string) để tìm vị trí của bắt đầu hoặc cuối cùng mà str2 không thuộc str1.

i. Trả về kích thước của không gian lưu trữ hiện được phân bổ cho chuỗi (có thể lớn hơn hoặc bằng kích thước chuỗi, tối ưu hóa cho việc bổ sung thêm ký tự vào chuỗi string)

```
    string str("1234231");
    cout << "\nLength is: " << str.length();
    cout << "\nSize is: " << str.size();
    cout << "\nCapacity is: " << str.capacity();
    cout << "\nMax size is: " << str.max_size();
```

```
Output:
Length is: 7
Size is: 7
Capacity is: 15
Max size is: 9223372036854775807
```

j. Hàm copy ký tự của chuỗi con trong string

```
    string str("Day la hoc phan: Phuong phap lap trinh huong doi tuong");
    char ch[100];
    str.copy(ch, 17, 0);//lấy chuỗi con có phần tử bắt đầu từ phần tử 0 đến phần tử 17
    cout << ch;
```

```
Output:
Day la hoc phan:
```

k. Swap tring 1 và string 2

```
    string str1("Day la OOP");
    string str2("Day la DSA");
    cout << "\nstring 1: " << str1;
    cout << "\nString 2: " << str2;
    str1.swap(str2);
    cout << "\nAfter swapping, string 1: " << str1;
    cout << "\nAfter swapping, string 2: " << str2;
```

```
Output:
string 1: Day la OOP
String 2: Day la DSA
After swapping, string 1: Day la DSA
After swapping, string 2: Day la OOP
```

II. Vector

1. Đặc điểm

- Được coi là một danh sách (list) để lưu trữ dữ liệu.

- Cú pháp: vetor <kiểu dữ liệu> Tên biến

2. Thuộc tính

Là số phần tử của vector đó (size).

3. Phương thức

- Khởi tạo một list bằng cú pháp

    + **vector <**kiểu dữ liệu**> list**;

- Đẩy dữ liệu vào và ra.

    + Vào: **list.push_back(n);**

    + Ra: **list.pop_back();**

- Lấy length của vector

    + list.size();

- Lấy phần tử vị trí mong muốn, phần tử đầu, phần tử cuối.

    + Lấy phần tử tại vị trí i: **list.at(i-1);**

    + Lấy phần tử đầu: **list.front();**

    + Lấy phần tử cuối: **list.back();**

- Kiểm tra list có dữ liệu hay trống

    + list.empty();

- Xóa một phần tử tại một vị trí, xóa toàn bộ

+ Xóa phần tử tại vị trí i: **list.erase(list.begin() + i -1);**

+ Xóa toàn bộ list: **list.clear();**

4. Cách sử dụng

a. Khởi tạo một list bằng cú pháp

```
vector <int> list;
vector <Date> list;
vector <string> list;
```

b. Đẩy dữ liệu vào và ra.

- Đẩy dữ liệu vào

```
vector <int> list;
list.push_back(2);
```

```
        list.push_back(3);
        list.push_back(4);
        list.push_back(6);
        //Các phần tử được thêm vào list {2,3,4,6}
```

- Đẩy dữ liệu ra

```
        vector <int> list;
        list.push_back(2);
        list.push_back(3);
        list.push_back(4);
        list.push_back(6);
        list.pop_back(); //đẩy phần tử đầu của mảng ra ngoài
        //Các phần tử sau khi thực hiện pop_back(), list có dạng
{3,4,6}
```

c. Lấy length của vector

```
        vector <int> list;
        list.push_back(2);
        list.push_back(3);
        list.push_back(4);
        list.push_back(6);
        cout << "Size of list: " << list.size();//lấy size của list

_____________________________________________________________________
Output:
Size of list: 4
```

d. Lấy phần tử vị trí bất kỳ, phần tử đầu, phần tử cuối.

```
        vector <int> list;
        list.push_back(2);
        list.push_back(3);
        list.push_back(4);
        list.push_back(6);
```

```
      cout << "\nFirst element is: " << list.front(); //lấy ra giá trị của phần tử ở vị trí đầu list
      cout << "\nLast element is: " << list.back(); //lấy ra giá trị của phần tử ở vị trí cuối cùng list
      cout << "\nElement 2 is: " << list.at(2 - 1); //lấy phần tử ở vị trí số 2
```

```
Output:
First element is: 2
Last element is: 6
Element 2 is: 3
```

e. Kiểm tra list có dữ liệu hay trống

```
      vector <int> list;
      // list.push_back(2);
      // list.push_back(3);
      // list.push_back(4);
      // list.push_back(6);

      if(list.empty() == true)
        cout << "\nList is empty";
      else cout << "\nList isn't empty";
```

```
Output:
List is empty
```

```
      vector <int> list;
      list.push_back(2);
      list.push_back(3);
      list.push_back(4);
      list.push_back(6);
```

```
      if(list.empty() == true)
        cout << "\nList is empty";
      else cout << "\nList isn't empty";
```

```
Output:
List isn't empty
```

f. Xóa một phần tử tại một vị trí, xóa toàn bộ

- Xóa phần tử tại một vị trí bất kỳ

```
      vector <int> list;
      list.push_back(2);
      list.push_back(3);
      list.push_back(4);
      list.push_back(6);

      list.erase(list.begin() + 1);  //Phần tử thứ 2 bị xóa
      //list lúc này {2,4,6}, sau đó lấy phần tử số 2 ra để kiểm chứng
      cout << "\nElement 2 is: " << list.at(2 - 1); //lấy phần tử ở vị trí số 2
```

```
Output:
Element 2 is: 4
```

- Xóa toàn bộ phần tử trong list

```
      vector <int> list;
      list.push_back(2);
      list.push_back(3);
      list.push_back(4);
      list.push_back(6);

      list.clear();//Xóa toàn bộ các phần tử
      if(list.empty() == true) //Kiểm chứng
        cout << "\nList is empty";
```

```
      else cout << "\nList isn't empty";
```

Output:

```
List is empty
```