string的使用

各种基本操作

3. 字符串查找（find）

```
    //3.字符串查找(find)
    std::string str = "Hello, World!";
    size_t pos = str.find("World!");//查找子字符串的位置
    if (pos != std::string::npos){
        std::cout << "Substring found at position :" << pos << std::endl;
    }
    else{
        std::cout << "Substring not found." << std::endl;
    }
```

来源CSDN上关于size_t和npos的讲解请看后文的拓展知识1

4. 字符串替换（replace）

```
    std::string str = "Hello, World!"
    str.replace(7, 5, "Universe");//替换子字符串
    std::cout << "Result: " << str << std::endl;
```

std::replace(字符串起始位置, 长度, 要替换的内容)

5. 提取子字符串（substr）

```
std::string str = "Hello, World!"
std::string subStr = str.substr(7, 5);
std::cout << "Substring: " << subStr << std::endl;
```

substr(位置, 长度)，注意，长度不要越界。

6. 字符串的比较（compare）

```
std::string str1 = "Hello";
std::string str2 = "World";
int result = str1.compare(str2);//比较字符串
if (result == 0){
    std::cout << "Strings are equal." << std::endl;
} else if (result < 0) {
    std::cout << "String1 is less than String2." << std::endl;
} else {
    std::cout << "String1 is greater than String2." << std::endl;
}
```

string重载了不等号，所以可以直接使用`s1<s2`的方式来比较string的大小，比较的规则是按照字典序大小进行比较。

字典序的比较方法是从小到大一个一个比较，一旦遇到不相等的字符就确定大小关系。

```
aaaa < bbbb
azz < baaa
azzzzzzzzzzzzzzz < b
lanqiao == lanqiao
```

7. 遍历string

常用的遍历string的方法有两种：

1. 循环枚举下标
2. auto枚举（其中&表示取引用类型，如果对i修改将会改变原来的值）(**C++11特性**)

```
string s = "Hello";

for (int i = 0;i < s.length(); ++ i)cout << s[i];
cout << '\n';
for (auto i : s)
{
    cout << i;
    i = 'a';//此处的修改无效，因为这个i是拷贝出来的，而不是引用s的
}
cout << '\n';//此时s = "Hello"
for (auto &i : s)
{
    cout << i;
    i = 'a';//此处修改会改变s的字符值
}
cout << '\n';//此时s = "aaaaa"
cout << s << '\n';
```

拓展知识

1. size_t 和 npos

size_t

size_t 是一些C/C++标准在stddef.h中定义的，size_t类型表示**C中任何对象所能达到的最大长度**，它是**无符号整数**。

打印size_t类型的值时要小心，因为这可是一个无符号的值，如果选错格式说明符，有可能会得到一个不可靠的结果。推荐的格式说明符是`%zu`，作为替代，可以考虑`%u`或`%lu`。

```
size_t sizet = -5;
printf("%d\n", sizet);
printf("%zu\n", sizet);
```

因为size_t本来是用于表示正整数的,如果用来表示负数就会出现问题。如果为其赋一个负数，然后用上述代码中的格式说明符进行打印，就会得到如下结果：

```
-5
4294967291
```

npos

在MSDN的说明中，npos是一个常数，表示**size_t的最大值**，许多容器都提供这个东西，用来表示不存在的位置，类型一般是std::container_type::size_type>。

2. 字典序

字典序(dictionary order)，又称字母序，原意是表示英文单词在字典中的先后顺序，在计算机领域扩展成两个任意字符串的大小关系。
英文中的字母表按照ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz顺序排列，在字典中，单词是按照首字母在字母表中的顺序进行排列的，比如alpha在beta前面。而第一个字母都相同时，会去比较两个单词的第二个字母在字母表中的顺序，比如account在advance之前，以此类推。下列单词就是按照字典序进行排列的：

```
as
aster
astrolabe
astronomy
astrophysics
at
ataman
```

在计算机领域当中，这个字典序就不仅仅用来比较英语单词了，而是比较任意字符串。对于两个字符串，大小关系取决于两个字符串从左到右第一个不同字符的ASCII值的大小关系。比如ahlx小于ahb，而Z5小于a3。