C++_STL记录

1.字符串比较

> 比较两个char* 字符串

```
int strcmp(const char* s1,const char* s2);
```

- cstring头文件
- s1<s2，返回负数

> 比较两个string对象

```
s1==s2
s1.compare(s2)
```

> 比较两个字符串

- 不能用==，那样是比地址。比string对象是可以的。

2.find函数

> 下指string对象

```
string s1 = "abcdef";
string s2 = "de";
// 子串第一次出现的位置，从下标2开始找
int ans = s1.find(s2, 2);    // 如果没找到，返回s1.npos
// 第一次出现的位置
int ans1 = s1.find_first_of(s2);
// 最后一次出现的位置
int ans2 = s1.find_last_of(s2);
// 反向查找子串在母串中出现的位置
int ans3 = s1.rfind(s2);
```

```
//查找s 中flag 出现的所有位置。
string s = "abcaca";
string flag = "a";
int pos = 0;
while((pos = s.find(flag, pos) != string::npos)){
    cout << pos << endl;
    pos++;
}
```

3.字符串长度

> string长度

```
// （1）length()
#include <string>
string user;
int len = user.length();
// （2）size()
int len = user.size();
```

> char * 长度

```
// （3）strlen()
#include <cstring>
char* user = new char;
int len = strlen(user);
delete user;
```

4.字符串切割

```
string s = "abc";
s.substr(begin, len);
```

5.判定字母、数字

```
//stl库函数判断
//字母（不区分大小写）：
isalpha();
//大写字母：
isupper();
//小写字母：
islower();
//数字：
isdigit();
//字母和数字：
isalnum();
//大小写字母转化：
toupper();
tolower();
```

6.数字加入字符串

```
string temp = "123";
temp += to_string(4);
temp += '5';
cout << temp;
```

7.字符串转换

```
//字符串转数字
#include<cstdlib>
int atoi(const char* str);
//string 转 const char *
#include<cstring>
const char *c;
string s = "1234";
c = s.c_str();
// const char* 赋值给char *
char a[4];
const char*c = "asd";
strcpy_s(a, c);
// const char* 赋值给string
const char* s = "123";
string str(s);
string str2 = s;
//
```

8.双指针模板

```
for (int i = 0; i < n; i++) {
    int j = i;
    // condition是j递增的条件函数
    while (j < n && condition(j))   j++;
    // 函数体
    i = j;
}
```

9.erase

（1）erase(pos,n); 删除从pos开始的n个字符，比如erase(0,1)就是删除第一个字符

（2）erase(position);删除position处的一个字符(position是个string类型的迭代器)

（3）erase(first,last);删除从first到last之间的字符（first和last都是迭代器）

```
#include<string>
using namespace std;
int main ()
{
string str ("This is an example phrase.");
string::iterator it;
//第（1）种方法
str.erase (10,8);                            // "This is an phrase."
//第（2）种方法
it=str.begin()+9;
str.erase (it);                              // "This is a phrase."
//第（3）种方法
```

```
str.erase (str.begin()+5, str.end()-7);     // "This phrase."
}
```

10.读长度未知的行输入

```
#include<string>
using namespace std;
int main(){
    string line;
    getline(cin, line);     // 默认 \n 终止
}
```

11.StringStream

```
#include<string>
#include<iostream>
#include<sstream>
#include<vector>
#include<array>
using namespace std;
int main(){
    string strvals = "32 240 2 1450";
    istringstream iss;      // istringstream iss(strvals);
    iss.str(strvals);       // str是get或set iss的
    int val;
    for (int i = 0; i < 4; i++) {
        iss >> val;
        cout << val << endl;
    }
    cout << iss.str();
    //
    stringstream ss;
    ss << 100 << " " << 200;
    int foo, bar;
    ss >> foo >> bar;
    cout << "foo: " << foo << endl;
    cout << "bar: " << bar << endl;
    // istream 也有一个getline
    istringstream input("abc|def|gh");
    vector<array<char, 4>> va;
    for (array<char, 4> arr; input.getline(&arr[0], 4, '|');) {
        va.push_back(arr);
    }
    for (auto& a : va) {
        cout << &a[0] << endl;
    }
}
```

12.strstr函数

```
#include<cstring>
char *strstr(char *str1, const char *str2);
//
char str[] = "1234xyz";
char* str1 = strstr(str, "34");
cout << str1 << endl;               // 34xyz
```

13.fstream

```
#include<fstream>
#include<string>

int main(){
    ifstream ifs("五子棋.txt");
    //
    char c;
    ifs.get(c);
    cout << c;
    //
    ifstream ifs("五子棋.txt");
    string line;
    getline(ifs, line);
    cout << line;
    //
    char ln[50];
    ifs.getline(ln, 50);
    cout << ln;
}
```

999.unistd.h库文件

- unistd.h 是 C 和 C++ 程序设计语言中提供对 POSIX 操作系统 API 的访问功能的 头文件的名称。该头文件由 POSIX.1 标准（单一UNIX规范的基础）提出。
- 对于类 Unix 系统，unistd.h 中所定义的接口通常都是大量针对 系统调用的封装（ 英语：wrapper functions），如 f**ork、pipe 以及各种 I/O 原语（read、write、close 等等）**。

```
#include<unistd.h>
int main(){
    sleep(1);
}
```

1000.时间函数

- time_t是定义在time.h中的一个类型，表示一个日历时间，也就是从1970年1月1日0时0分0秒到此时的秒数。

```
#include<time.h>
/* 是一个长整型 */
typedef long time_t;
// 获取当前时间
time_t nowtime = time(NULL);
//获取格林尼治时间
struct tm* gm = gmtime(&nowtime);            // 2022年10月24日 12:1:38  星期一 当年的第296天
dsptime(gm);
// 获取当前系统时间
struct tm* local = localtime(&nowtime);      // 2022年10月24日 20:1:38  星期一 当年的第296天
dsptime(local);
// 用字符串格式表示时间的函数
char * asctime(const struct tm *)        // Sun Mar 14 13:59:23 2010
char * ctime(const time_t *)             // Sun Mar 14 21:59:23 2010
// clock_t也是一个长整型，表示的是从程序开始运行到执行clock函数时所经过的cpu时钟计时单元数。
typedef long clock_t;
clock_t clock(void);
//
clock_t cstart, cends;
cstart = clock();
...
cends = clock();
cout << "Clock时间差：" << cends - cstart << endl;      // Clock时间差：3094
// 自定义时间
size_t strftime(char *strDest, size_t maxsize, const char *format, const struct tm *timeptr);
```

```
void dsptime(const struct tm *ptm)
{
    char *pxq[] = {"日", "一", "二", "三", "四", "五", "六"};
    cout << ptm->tm_year + 1900 << "年" << ptm->tm_mon + 1 << "月" << ptm->tm_mday << "日 " ;
    cout << ptm->tm_hour << ":" << ptm->tm_min << ":" << ptm->tm_sec << " " ;
    cout << " 星期" << pxq[ptm->tm_wday] << " 当年的第" << ptm->tm_yday << "天 " << endl;
}
```