Lambda

用于传递匿名函数

[capture list] (params list) mutable exception-> return type { function body }

Unique

去掉容器中相邻元素的重复元素（不一定要求数组有序），它会把重复的元素添加到容器末尾（所以数组大小并没有改变），而返回值是去重之后的尾地址

用法：

由于返回的是容器末尾，所以如果想得到去重后的size，需要减去初始地址，lower\_bound是得到地址，稍微不同

IMG_256 unique和lower\_bound对比

unique

sz = unique(b + 1,b + n + 1)-(b + 1);

sz = unique(a,a + n) - a;

Lowerbound

Pos = lower\_bound(b + 1,b + sz + 1,a[i]) ;

lower\_bound()返回一个 iterator 它指向在[first,last)标记的有序序列中可以插入value，而不会破坏容器顺序的第一个位置，而这个位置标记了一个不小于value 的值

Upper\_bound

upper\_bound()返回一个 iterator 它指向在[first,last)标记的有序序列中可以插入value，而不会破坏容器顺序的最后一个位置，而这个位置标记了一个不小于value 的值

Prority\_queue

三个参数，基本数据结构 + 承载底层数据堆的容器 + 比较类/可以用写在struct中的operator < 重载

单独写则是函数类

IMG_257 Strtok()

Char buf[] = "hello@boy@this@heima";

Char \* temp = strtok(buf,"@");

While(temp){

Printf("%s ",temp);

Temp = strtok(NULL,"@");

}

IMG_258 Input function

<http://www.cnblogs.com/qiang-wei/p/9332201.html>

1.scanf()函数

如果a,b只读取成功了一个，则返回1；如果a,b都没有读取成功，则返回0.

2.cin>>函数

说明：当我们从键盘输入时，有一个缓冲区，当输入结束时会将所输入的数据存到缓冲区。而cin>>的作用就是从缓冲区读数据，所以当缓冲区数据本来就有残留时，会出现读取出错的问题。值得一提的是：cin函数遇到空格、TAB、换行时读取结束。

#include <iostream>  
using namespace std;  
void main ()  
{  
int a,b;  
cin>>a>>b;  
cout<<a+b<<endl;

}

输入：aaabbbccc

输出：aaabbbccc

输入：aaa bbb ccc

输出：aaa

3.cin.get()函数

原型：该函数有三种形式，分别是char ch = cin.get(), cin.get(char ch), cin.get(array,length)

    1.cin.get() 和 cin.get(char ch)的用法

    这两个函数功能基本一样，都是读取缓冲区的一个字符，遇到换行符结束，但是这两个函数都不会丢弃缓冲区里边的空格和换行符。char ch = cin.get(), cin.get(char ch)这两句可得到相同的结果

   #include <iostream>  
      using namespace std;  
      void main()  
     {  
        char ch1,ch2;  
        cin.get(ch1);  
        ch2 = cin.get();

        cout<<ch1<<endl;  
        cout<<ch2<<endl;  
    }

    输入：a换行

    输出：a

    （此处有一行空行，因为不会丢弃缓冲区里边的空行）

4.cin.get(array,length)

    这个函数是从缓冲区读取字符串，其中array是字符串接受地址，length-1是字符串长度，因为最后一个字符必须存  储'\0'结束符，所以此函数只能接受length-1个字符。

#include <iostream>  
using namespace std;  
    void main()  
    {  
        char ch[5];  
        cin.get(ch,5);  
        cout<<ch<<endl;  
    }

输入：abcdefghi

输出：abcd

5.cin.getline(string str,int length)函数

说明：接收一个长度为length-1的字符串，包括空格，遇到换行结束。

#include<iostream>  
using namespace std;  
int main()  
{  
char ch[10];  
cin.getline(ch,5);  
cout<<ch<<endl;  
return 0;  
}

输入：aaaaaaaa

输出：aaaa

getline()函数

原型：ssize\_t getline(char \*\*lineptr,size\_t \*n,FILE \*stream);(在Ｃ语言的GCC扩展定义中)

    　　getline(cin,string s);（在C++中）

说明：getline不是C语言的库函数，而是GCC的扩展定义或者C++库函数，在Ｃ语言中和C++中的使用是不同的.

用例１：（在C++中,应包含头文件<string>，读取字符串包含空格，遇到换行结束，不包括换行）

＃include <iostream>  
using namespace std;  
int main()  
{  
    string s;  
    getline(cin,s);  
    cout<<s<<endl;

return 0;  
}

输入：abcdefgh换行

输出：abcdefgh

gets()函数

原型：char \*gets(char \*buffer);

入口参数：从缓冲区读取字符串后的写入地址

返回值：读取成功后返回与入口参数buffer相同的地址。读取错误返回NULL。

头文件：<string>

说明:接收输入的字符串，没有上限，但是要保证buffer足够大，以换行结束，并将换行符转化为'\0'。

#include<iostream>  
#include<string>  
using namespace std;  
int main()  
{  
char ch[10];  
gets(ch);  
cout<<ch<<endl;  
return 0;  
}

输入：abcdefg回车

输出：abcdefg

getchar()函数

原型：int getchar(void);

返回值：读取成功返回用户输入的ASCII码，读取失败返回EOF.

头文件：<string>或者<stdio.h>

说明：接收一个字符的输入，包含空格，遇到换行符结束。

#include<iostream>  
#include<string>  
using namespace std;  
int main()  
{  
char ch;  
ch = getchar();  
cout<<ch<<endl;  
return 0;  
}

输入：abc回车

输出：a

IMG_259 Dynamic\_cast

用途：执行安全地向下类型转换

将一个基类对象指针（或引用）cast到继承类指针，dynamic\_cast会根据基类指针是否真正指向继承类指针来做相应处理

Example

在类层次间进行上行转换时，dynamic\_cast和[static\_cast](https://baike.baidu.com/item/static_cast)的效果是一样的；

在进行下行转换时，dynamic\_cast具有类型检查的功能，比static\_cast更安全。

Dynamic\_cast<new\_type>(expression)

Class B{

Public:

Int m\_iNum;

Virtual void foo();

};

Class D:public B{

Public:

Char \* m\_szName[100];

};

Void func(B\* pb){

D \* pb1 = static\_cast<D\*>(pb);

D \* pb2 = dynamic\_cast<D\*>(pb);

}

在上面的code segments中，如果 pb 指向一个 D 类型的对象，pd1 和 pd2 是一样的，并且对这两个pointer执行 D 类型的任何操作都是安全的；但是，如果 pb 指向的是一个 B 类型的对象，那么 pd1 将是一个指向该对象的指针，对它进行 D 类型的操作将是不安全的（如访问 m\_szName），而 pd2 将是一个空指针。

另外要注意：B 要有virtual function，否则会编译出错；static\_cast则没有这个限制。

IMG_260 Example

String的隐式类型转换（把一个类转换成string)

Operate string(){

…

Return str;

}

IMG_261 时间函数

//方案— 优点：仅使用C标准库；缺点：只能精确到秒级

#include <time.h>

#include <stdio.h>

int main( void )

{

    time\_t t = time(0);

    char tmp[64];

    strftime( tmp, sizeof(tmp), "%Y/%m/%d %X %A 本年第%j天 %z",localtime(&t) );

    puts( tmp );

    return 0;

}

size\_t strftime(char \*strDest, size\_t maxsize, const char \*format, const struct tm \*timeptr);

根据格式字符串生成字符串。

struct tm \*localtime(const time\_t \*timer);

取得当地时间，localtime获取的结果由结构tm返回

返回的字符串可以依下列的格式而定：

%a 星期几的缩写。Eg:Tue

%A 星期几的全名。 Eg: Tuesday

%b 月份名称的缩写。

%B 月份名称的全名。

%c 本地端日期时间较佳表示字符串。

%d 用数字表示本月的第几天 (范围为 00 至 31)。日期

%H 用 24 小时制数字表示小时数 (范围为 00 至 23)。

%I 用 12 小时制数字表示小时数 (范围为 01 至 12)。

%j 以数字表示当年度的第几天 (范围为 001 至 366)。

%m 月份的数字 (范围由 1 至 12)。

%M 分钟。

%p 以 ''AM'' 或 ''PM'' 表示本地端时间。

%S 秒数。

%U 数字表示为本年度的第几周，第一个星期由第一个周日开始。

%W 数字表示为本年度的第几周，第一个星期由第一个周一开始。

%w 用数字表示本周的第几天 ( 0 为周日)。

%x 不含时间的日期表示法。

%X 不含日期的时间表示法。 Eg: 15:26:30

%y 二位数字表示年份 (范围由 00 至 99)。

%Y 完整的年份数字表示，即四位数。 Eg:2008

%Z(%z) 时区或名称缩写。Eg:中国标准时间

%% % 字符。

//方案二 优点：能精确到毫秒级；缺点：使用了windows API

#include <windows.h>

#include <stdio.h>

int main( void )

{

SYSTEMTIME sys;

GetLocalTime( &sys );

printf( "%4d/%02d/%02d %02d:%02d:%02d.%03d 星期%1d\n",sys.wYear,sys.wMonth,sys.wDay,sys.wHour,sys.wMinute, sys.wSecond,sys.wMilliseconds,sys.wDayOfWeek);

return 0;

}

//方案三,优点：利用系统函数,还能修改系统时间

//此文件必须是c++文件

#include<stdlib.h>

#include<iostream>

using namespace std;

void main()

{

    system("time");

}

//方案四,将当前时间折算为秒级，再通过相应的时间换算即可

//此文件必须是c++文件

#include<iostream>

#include<ctime>

using namespace std;

int main()

{

time\_t now\_time;

now\_time = time(NULL);

cout<<now\_time;

return 0;

}