计算机学院 高级程序语言设计 课程实验报告

实验题目：标准模板库 STL		学号：202300130150
日期：5.9	班级： 4	姓名： 王成意

实验目的：
了解 C++标准模板库 STL 使用

实验步骤与内容：
参照《C++语言程序设计》学生用书和课件 PPT，完成以下内容：

1. 结合 PPT 例 10.2 做以下实验
（1）运行并截图

```cpp
//10_2.cpp
#include <iterator>
#include <iostream>
#include <algorithm>
using namespace std;
//求平方的函数
double square(double x) {
    return x * x;
}
int main() {
    //从标准输入读入若干个实数，分别将它们的平方输出
    transform(istream_iterator<double>(cin), istream_iterator<double>(),
        ostream_iterator<double>(cout, "\t"), square);
    cout << endl;
    return 0;
}
```

```
问题  输出  调试控制台  终端  端口

PS D:\C++ programs>  & 'd:\VsCode-extentions\.vscode-insiders\extensions\ms-vscode.cpptools-1.20.4-win32-x64\debugAdapter
db1slvxr.hzr' '--stdout=Microsoft-MIEngine-Out-5ccneq2t.2z1' '--stderr=Microsoft-MIEngine-Error-dzjjnspp.ud0' '--pid=Micr
xe' '--interpreter=mi'
2
4
6
36
19
361
100
10000
45
2025
^Z
```

（2）结合 transform 算法和 squre 函数，重新写一个 main 函数实现以下功能。初始化一个 int 数组 a，数组内容为{1, 2, 3, 4, 5, 6}。再定义一个 int 数组 b，与 a 有同样多的数据。通过调用 transform 函数实现将 a 中的每个元素求平方并放到 b 中对应位置，最后将 b 中的内容输出。transform 函数的每一个参数应该是什么？

```cpp
// 10_2.cpp
#include <iterator>
#include <iostream>
#include <algorithm>
using namespace std;
// 求平方的函数
double square(double x)
{
    return x * x;
}
double a[] = {1, 2, 3, 4, 5, 6};
double b[6];
int main()
{
    // 从标准输入读入若干个实数，分别将它们的平方输出
    transform(a, a + 6, b, square);
    for (int i = 0; i < 6; ++i)
    {
        cout << b[i] << " ";
    }
    return 0;
}
```

```
问题  输出  调试控制台  终端  端口

PS D:\C++ programs>
PS D:\C++ programs>  & 'd:\VsCode-extentions\.vscode-insiders\extensions\ms-vscode.cpptools-1.20.4-win32-x64\
d5rxm0ke.mig' '--stdout=Microsoft-MIEngine-Out-y2slocjj.u51' '--stderr=Microsoft-MIEngine-Error-ktosopjr.ptq'
xe' '--interpreter=mi'
1 4 9 16 25 36
PS D:\C++ programs> ^C
PS D:\C++ programs>
PS D:\C++ programs>  & 'd:\VsCode-extentions\.vscode-insiders\extensions\ms-vscode.cpptools-1.20.4-win32-x64\
tv3z4tle.rmn' '--stdout=Microsoft-MIEngine-Out-zt35fhmu.4tv' '--stderr=Microsoft-MIEngine-Error-i01xga23.kea'
xe' '--interpreter=mi'
1 4 9 16 25 36
```

Transform 参数：a, a + 6, b, square

（3）利用 vector 而不是普通的数组来实现（2）的要求。注意 begin()和 end()函数的使用，b 最开始是空的，可以通过 b.resize(a.size())将 b 的数组大小强行调整为与 a 相同。

```cpp
#include <iterator>
#include <iostream>
#include <algorithm>
#include <vector>
using namespace std;
// 求平方的函数
double square(double x)
{
    return x * x;
}
vector<double> a{1, 2, 3, 4, 5, 6}, b;
int main()
{
    // 从标准输入读入若干个实数，分别将它们的平方输出
    b.resize(a.size());
    transform(a.begin(), a.end(), b.begin(), square);
    for (int i = 0; i < 6; ++i)
    {
        cout << b[i] << " ";
    }
    return 0;
}
```

```
问题  输出  调试控制台  终端  端口

PS D:\C++ programs>
PS D:\C++ programs>  & 'd:\VsCode-extentions\.vscode-insiders\extensions\ms-vscode.cpptools-1.20.4-win32-x64\debu
d5rxm0ke.mig' '--stdout=Microsoft-MIEngine-Out-y2slocjj.u5l' '--stderr=Microsoft-MIEngine-Error-ktosopjr.ptq' '--
xe' '--interpreter=mi'
1 4 9 16 25 36
PS D:\C++ programs> ^C
PS D:\C++ programs>
PS D:\C++ programs>  & 'd:\VsCode-extentions\.vscode-insiders\extensions\ms-vscode.cpptools-1.20.4-win32-x64\debu
tv3z4tle.rmn' '--stdout=Microsoft-MIEngine-Out-zt35fhmu.4tv' '--stderr=Microsoft-MIEngine-Error-i01xga23.kea' '--
xe' '--interpreter=mi'
1 4 9 16 25 36
PS D:\C++ programs> ^C
PS D:\C++ programs>
PS D:\C++ programs>  & 'd:\VsCode-extentions\.vscode-insiders\extensions\ms-vscode.cpptools-1.20.4-win32-x64\debu
cxtemnv4.2zx' '--stdout=Microsoft-MIEngine-Out-xheif0y0.bxx' '--stderr=Microsoft-MIEngine-Error-juecbeee.boo' '--
xe' '--interpreter=mi'
1 4 9 16 25 36
PS D:\C++ programs>
```

2. PPT 例 10.7 给出了一个实验利用栈反向输出单词，请修改这个程序，实现以下功能。我们输入一个字符串，中间有英文字母和英文的逗号如“terry,is,good”。写一个程序对该字符串进行处理，利用栈实现单词的反向输出，各单词之间通过回车隔开。上述例子的输出结果如下

good
is
terry

一些 tips：通过#include <stack>;和#include <string>;可以使用 stack 类和 string 类。如果 str 是一个 string 类，str += 'a';操作可以在 str 后面接一个 a 字符。str.clear();可以将 str 的内容清空。

代码：（运用了 stack 和 string）

```cpp
#include <cstdio>

#include <iostream>
```

```cpp
#include <stack>
#include<string>
using namespace std;
int main()
{
    stack<char> s; // include<stack>
    string str;    // include<string>
    cin >> str;    // 从键盘输入一个字符串
    // 将字符串的每个元素顺序压入栈中
    for (string::iterator iter = str.begin(); iter != str.end(); ++iter)
    {
        if(*iter==',')
        {
            *iter = '\n';
        }
        s.push(*iter);
    }
    // 将栈中的元素顺序弹出并输出
    stack<char> s2;
    while (!s.empty())
    {
```

```
        if(s.top()=='\n')
        {
            while(!s2.empty())
            {
                cout << s2.top();
                s2.pop();
            }
            puts("");
            s.pop();
        }
        else
        {
            s2.push(s.top());
            s.pop();
        }
    }
    while(!s2.empty())
    {
        cout << s2.top();
        s2.pop();
    }
    return 0;
```

```
}
```

运行结果：

```
xe' '--interpreter
terry,is,good
good
is
terry
```

3. PPT 例 10.8 通过优先级队列模拟了细胞分裂，实践一下。请问这个程序能不能顺利编译运行？不能的话看一下问题出在哪里，解决问题并回答为什么。

不能顺利运行，报错如下：

```
D:\C++ programs\1.cpp: In function 'int main()':
D:\C++ programs\1.cpp:47:31: error: passing 'const value_type' {aka 'const Cell'} as 'this' argument discards qualifiers [-fpermissive]
         cellQueue.top().split(); // 模拟下一个细胞的分裂
                                ^
D:\C++ programs\1.cpp:28:10: note:        in call to 'void Cell::split()'
     void split()
```

可见是因为丢弃限定符，当尝试修改 const 类型时，会报 discards qualifiers。

修改：把 split 函数改成常函数：

```
    }
    void split() const
    {                                           // 细胞分
        Cell child1(time), child2(time); // 建立两
        cout << time << "s: Cell #" << id << " spl
             << child1.getId() << " and #" << chil
```

，顺利运行。

总代码：

```
#include <cstdio>

#include <queue>

#include <iostream>

#include<time.h>
```

```
using namespace std;
const int SPLIT_TIME_MIN = 500;  // 细胞分裂最短时间
const int SPLIT_TIME_MAX = 2000; // 细胞分裂最长时间
class Cell;
priority_queue<Cell> cellQueue;
class Cell
{ // 细胞类
private:
    static int count; // 细胞总数
    int id;           // 当前细胞编号
    int time;         // 细胞分裂时间
public:
    Cell(int birth) : id(count++)
    { // birth 为细胞诞生时间
        // 初始化，确定细胞分裂时间
        time = birth + (rand() % (SPLIT_TIME_MAX -
SPLIT_TIME_MIN)) + SPLIT_TIME_MIN;
    }
    int getId() const { return id; }          // 得到细
胞编号
    int getSplitTime() const { return time; } // 得到细
胞分裂时间
```

```
    bool operator<(const Cell &s) const       // 定义
“<”
    {
        return time > s.time;
    }
    void split() const
    {                                        // 细胞分裂
        Cell child1(time), child2(time); // 建立两个子
细胞
        cout << time << "s: Cell #" << id << " splits
to #"
             << child1.getId() << " and #" <<
child2.getId() << endl;
        cellQueue.push(child1); // 将第一个子细胞压入优
先级队列
        cellQueue.push(child2); // 将第二个子细胞压入优
先级队列
    }
};
int Cell::count = 0;
int main()
{
```

```
    srand(static_cast<unsigned>(time(0)));
    int t; // 模拟时间长度
    cout << "Simulation time: ";
    cin >> t;
    cellQueue.push(Cell(0)); // 将第一个细胞压入优先级队
列
    while (cellQueue.top().getSplitTime() <= t)
    {
        cellQueue.top().split(); // 模拟下一个细胞的分裂
        cellQueue.pop();         // 将刚刚分裂的细胞弹出
    }
    return 0;
}
```

运行结果：

```
xe' '--interpreter=mi'
Simulation time: 5000
853s: Cell #0 splits to #1 and #2
1867s: Cell #2 splits to #3 and #4
2023s: Cell #1 splits to #5 and #6
2612s: Cell #6 splits to #7 and #8
2692s: Cell #5 splits to #9 and #10
2834s: Cell #4 splits to #11 and #12
3066s: Cell #3 splits to #13 and #14
3210s: Cell #10 splits to #15 and #16
3646s: Cell #13 splits to #17 and #18
3722s: Cell #16 splits to #19 and #20
3739s: Cell #8 splits to #21 and #22
3749s: Cell #15 splits to #23 and #24
3813s: Cell #9 splits to #25 and #26
3930s: Cell #7 splits to #27 and #28
4037s: Cell #11 splits to #29 and #30
4346s: Cell #12 splits to #31 and #32
4359s: Cell #17 splits to #33 and #34
4386s: Cell #14 splits to #35 and #36
4496s: Cell #21 splits to #37 and #38
4648s: Cell #24 splits to #39 and #40
4810s: Cell #23 splits to #41 and #42
4870s: Cell #26 splits to #43 and #44
4900s: Cell #34 splits to #45 and #46
4947s: Cell #20 splits to #47 and #48
4996s: Cell #18 splits to #49 and #50
PS D:\C++ programs>
```

4. 例 10.16 介绍了如何通过 sort 函数直接对 vector 类中的元素进行排序，请将 sort 函数扩展到 Point 类中。Point 有两个公有的 int 数据 x 和 y。通过 vector<Point> pts = {Point(1, 2), Point(4, 10), Point(5, 1), Point(1, 10), Point(3, 2), Point(2, 6), Point(4, 3), Point(2, 1)}; 建立一个 Point 类型的 vector 数组，之后将 pts 进行排序。注意要自己写一个返回值为 bool 类型的函数作为 sort 的第三个参数，输入是两个 const Point &类型的引用。比较 p1 和 p2 大小时，两者先比较 x，相同的时候再比较 y。输出如下：

1 2
1 10
2 1
2 6
3 2
4 3
4 10
5 1

代码：

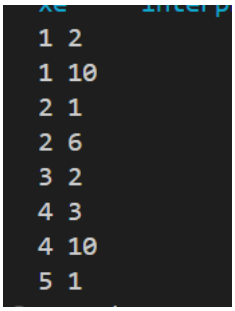
```
#include <functional>
```

```cpp
#include <iostream>
#include <vector>
#include <iterator>
#include <algorithm>
using namespace std;
class Point
{
private:
    int x,y;

public:
    Point(int x, int y) : x(x), y(y) {}
    int getx() const
    {
        return x;
    }
    int gety() const
    {
        return y;
    }
};
bool cmp(const Point &p,const Point &q)
```

```
{
    if(p.getx()==q.getx())
    {
        return p.gety() < q.gety();
    }
    return p.getx() < q.getx();
}
vector<Point> pts = {Point(1, 2), Point(4, 10),
Point(5, 1), Point(1, 10), Point(3, 2), Point(2, 6),
Point(4, 3), Point(2, 1)};
int main()
{
    sort(pts.begin(), pts.end(), cmp);
    for (auto it = pts.begin(); it != pts.end(); ++it)
    {
        cout << it->getx() << " " << it->gety() <<
endl;
    }
        return 0;
}
```

运行结果：



结论分析与体会：

非常好的实验，让我进一步认识了 transform 函数，STL 的各种容器和迭代器。