编号： 2-3

山东师范大学

SHANDONG NORMAL UNIVERSITY

信息科学与工程学院实验报告

《面向对象程序设计》

Object-Oriented Programming

姓名：	张泽浩
学号：	202111000212
班级：	计工本 2102
导师:	张庆科
时间：	2021 年 10 月 25 日

《面向对象程序设计》实验报告

基本要求：实验报告包含实验目的、实验内容、实验过程（详细操作流程）、实验结果（程序运行结果高清截图）、实验分析总结五个部分。报告中若涉及代码程序，请在附录部分提供完整程序源码及源码托管地址(基于 Highlight 软件导入源码)。报告撰写完毕后请提交 PDF 格式版本报告到课程云班课系统。

一、实验目的

1. 理解指针和引用的区别
2. 掌握 C++**引用**的基本用法
3. 掌握 C++**动态内存申请**的基本用法
4. 理解 C++**异常**的基本用法
5. 掌握 visual studio 代码调试方法

二、实验内容

（一）任务一：阅读分析代码

建立 VS 项目，调试实验题目中所给代码，找出程序 Bug 并更正，分析程序正确执行结果。

（二）任务二：动态内存分配

建立 C++项目，统计区间[100,999]内**水仙花数**的个数 N，然后将这 N 个数以数组形式存储在动态内存空间，然后按照从小到大的顺序输出所有的水仙花数。

提示：水仙花数是指一个 3 位数，它的每个位上的数字的 3 次幂之和等于它本身。例如：1^3 + 5^3+ 3^3 = 153。

（三）任务三：综合编程分析

给定 m 根木棍，每根木棍的长度记为 L_i,(3<= i <=m)，下面欲从这 m 根木棍中选择 3 根木棍组成周长尽可能最长的三角形、面积尽可能最大的三角形，分别输出最大的周长和面积。如果怎么选都无法构成三角形，请直接输出 0。

要求：

(1) 算法具有良好的可读性、稳健性和通用性(适合整数长度，浮点数长度)。

(2) 给出算法的复杂度分析，算法复杂度尽可能越低越好。

(3) 算法设计时要求采用函数重载，引用传递，及动态内存申请等 C++核心特性。

输入样例：

```
01 请输入木棍的数目m = 5
02 请输入5根木棍长度：2 3 4 5 10
```

程序输出：

```
01 构成的最大三角形周长为：12 （选择3,4,5）
02 构成的最大三角形面积为：6  （选择3,4,5）
```

提示：周长最大的三角形其面积未必一定最大

三、实验过程

（一）任务一：阅读分析代码

该程序中存在 5 类错误，分别为“形参与实参不是一一对应的”、“函数重载错误”、“输入输出错误”、“初始化数组错误”、“缺少内存空间释放错误”。

修改如下：

```
int& function(int v, int* p, int& r)
{
    未找到相关问题
    //function(a, b, c); 错误1 应该传b的地址
    function(a, &b, c);
```

图 1　形参与实参不一一对应

```
void swap(int x, int y)
{
    int temp;
    temp = x; x = y; y = temp;

}
void swap(int* x, int* y)
{
    int temp;
    temp = *x; *x = *y; *y = temp;
}

//void swap(int &x, int &y)  错误2 不能与上面函数构成函数重载
//{
//    int temp;
//    temp = x; x = y; y = temp;
//}
```

图 2　函数重载错误

```
//cout << "r = " << endl; 错误3 没输出r
cout << "r = " << r << endl;
```

图 3　输入输出错误

```
//int *p1 = new int[10](10);
//int *p2 = new int[10](10,20,30); 错误4 初始化数组错误
int* p1 = new int[10] {10};
int* p2 = new int[10] {10, 20, 30};
```

图 4　初始化数组错误

```
// 错误5 缺少对申请内存空间的释放操作
delete[] p1;
delete[] p2;
```

图 5　缺少内存空间释放错误

（二）任务二：动态内存分配

1. 数据结构

设计了一个结构体 SXH，结构体中存储了一个 int 型的数据（水仙花数）和一个 SXH*类型的指针。构成链表数据结构，便于动态分配内存（如图）。

```
typedef struct sxhnums {
    int nums;
    sxhnums* next;
}SXH;
```

图 6　结构体

2. 判断水仙花数的算法

分别利用除法和取余获取当前数据各位的数字，通过判断立方和是否等于当前数据。若等于，则是水仙花数，并申请一空间将其存储在链表中；否则进行下一次循环判断。

（三）任务三：综合编程分析

1. 题目理解：

在已知木棍长度中找一个周长最大且组成三角形面积最大的三边，并注意判断构成三角形的条件。

2. 设计算法（时间复杂度 n^3）

当 n 不是很大时，用暴力求解的方法进行求解（如图 7）。并设计三个函数，包括判断三边能否构成三角形的函数、已知三边求三角形面积的函数和判断当前周长和面积是否最大的函数。其返回值类型分别为 bool 类型、double 类型、bool 类型。定义了 2 个全局变量，方便判断周长和面积是否最大的函数进行判断操作。

```
// 复杂度太高 n^3
//     for (int i = 0; i < m; ++i) {
//         for (int j = i + 1; j < m; ++j) {
//             for (int k = j + 1; k < m; ++k) {
//                 if (isTrangle(length[i],length[j],length[k])) {
//                     if (maxS_C(length[i],length[j],length[k])) {
//                         maxS = quadrature(length[i],length[j],length[k]);
//                         maxC = length[i] + length[j] + length[k];
//                         a = length[i];
//                         b = length[j];
//                         c = length[k];
//                     }
//                 }
//             }
//         }
//     }
```

图 7　高复杂度算法

但是该算法存在缺陷：时间复杂度太高（n^3），不利于 n 很大的情况。

3. 优化算法（时间复杂度 $nlog_2n$）

对于上述算法中的暴力求解算法部分，通过分析数据的关系，将其时间复杂度优化到 $nlog_2n$。

首先先把数组排序(复杂度 $nlog_2n$),然后从大往小找,假设 1<a<b<c<n-1，那么 s[a]<s[b]<s[c],如果它们不能组成三角形那么不论是 c 增大,还是 a、b 减小，都 s[a]、s[b]、s[c]都不可能组成三角形。因为是从大到小找过来的，一定是大的不符合了，才会来计算小的，所以向上也不存在一组数据可以组成三角形（如图 8）。

```
// 优化
sort(length.begin(), length.end());
for (int i = m - 1; i > 1; --i) {
    if (isTrangle(length[i], length[i - 1], length[i - 2]) && maxS_C(length[i], length[i - 1], length[i - 2])) {
        maxS = quadrature(length[i], length[i - 1], length[i - 2]);
        maxC = length[i] + length[i - 1] + length[i - 2];
        c = length[i];
        b = length[i - 1];
        a = length[i - 2];
        break;
    }
}
```

图 8　优化后的算法

四、实验结果

（一）任务一：阅读分析代码运行结果

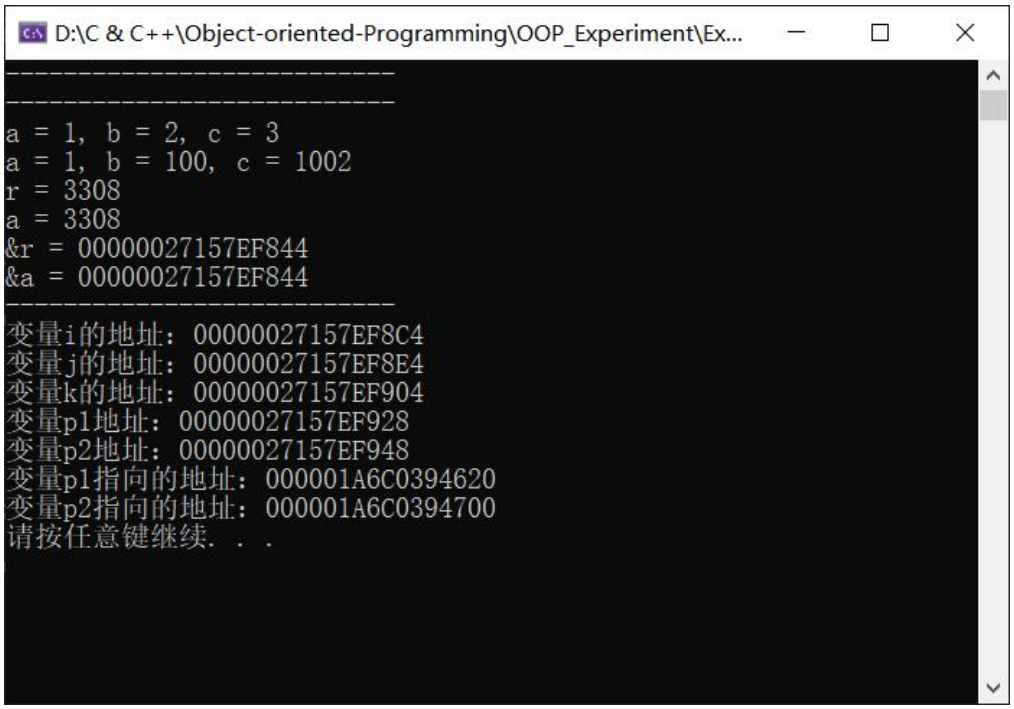

图 9　阅读分析代码运行结果

（二）任务二：动态内存分配运行结果

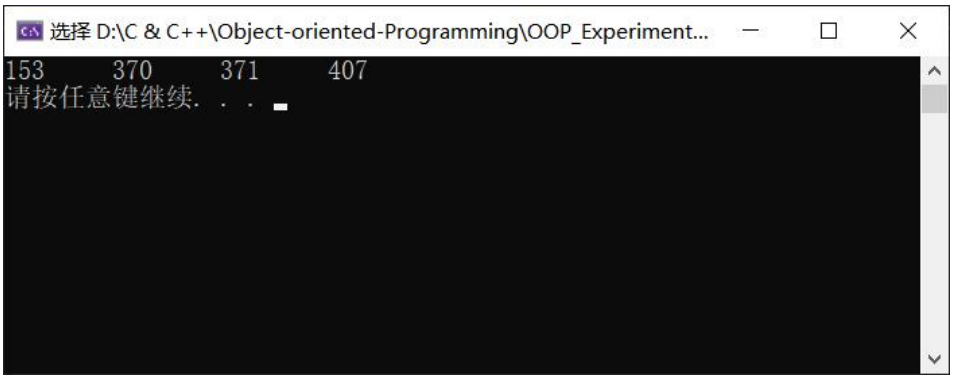

图 10　动态内存分配运行结果

（三）任务三：综合编程分析运行结果

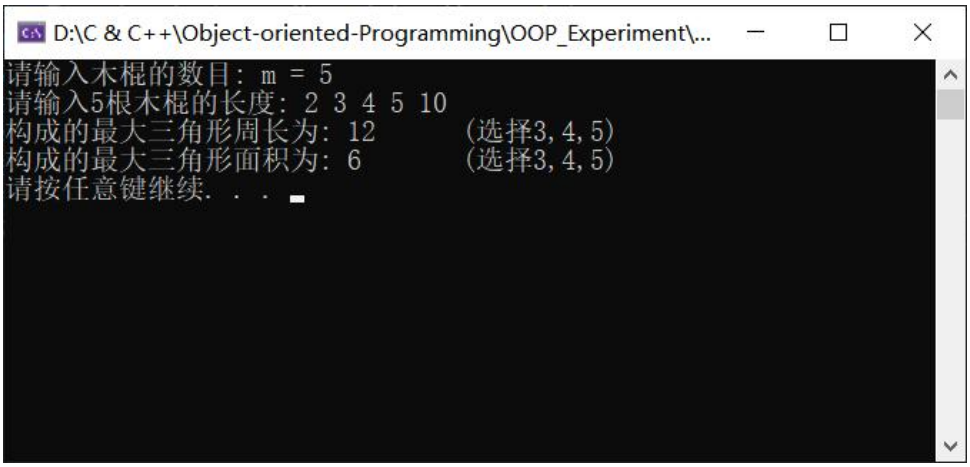

图 11　综合编程分析运行结果

五、实验总结

通过本次实验，更深刻的体会了函数重载与动态内存分配的相关操作。在任务三《综合编程分析》题目中，使用了 vector 容器直接指定所需内存空间的大小，也实现了随机访问，并体会到了提高代码效率的重要性。在编写程序之前，一定要考虑是否存在“捷径”，暴力解题的方法只能作为别无选择的选择。

附录：实验源代码（基于 Highlight 软件粘贴带有行号的源码）

代码托管地址：

Object-oriented-Programming/OOP_Experiment/Experiment-2_3 at master · keepIHDR/Object-oriented-Programming (github.com)

任务一：阅读分析代码改错

```
01 #include <iostream>
02 #include <cstdlib>
03 #include <windows.h>
04
05 using namespace std;
07
08 int& function(int v, int* p, int& r) {
09     v = (*p)++;
10
11     *p = 100;
12
13     p = new int;
14
15     *p = 1000;
16
17     r = v + (*p)++;
18
19     delete p;
20
21     return v;
22 }
24
25 void swap(int x, int y) {
26     int temp;
27     temp = x;
28     x = y;
29     y = temp;
31 }
32 void swap(int* x, int* y) {
33     int temp;
34     temp = *x;
35     *x = *y;
36     *y = temp;
37 }
38
39 /*
40     函数的重载的规则:
41         函数名称必须相同。
42         参数列表必须不同(个数不同、类型不同、参数排列顺序不)
```

```
45          函数的返回类型可以相同也可以不相同
46          仅仅返回类型不同不足以成为函数的重载
48 */
49
50 //void swap(int &x, int &y)  错误2 不能与上面函数构成函数重载
52 //{
53 //    int temp;
54 //    temp = x; x = y; y = temp;
55 //}
56
57 /******************** 主函数 ********************/
58
59 int main() {
61     cout << "----------------------------" << endl;
62
63     //1. 传值，传址和传引用分析
64     int a = 1, b = 2, c = 3;
65     cout << "----------------------------" << endl;
66     cout << "a = " << a << ", b = " << b << ", c = " << c <<
67     endl;
68     //function(a, b, c); 错误1 应该传b的地址
69     function(a, &b, c);
70     cout << "a = " << a << ", b = " << b << ", c = " << c <<
71     endl;
73
74     //2. 引用规律分析
75     int& r = a;
76     r = a + b + c;
77     a = r + a + b + c;
78     //cout << "r = " << endl; 错误3 没输出r
79     cout << "r = " << r << endl;
80     cout << "a = " << a << endl;
81
82     cout << "&r = " << &r << endl;
83     cout << "&a = " << &a << endl;
86
87     cout << "----------------------------" << endl;
88
89     //3. 局部变量存储分析
90     int i = 10, j = 20, k = 30;
91
92     cout << "变量i的地址: " << &i << endl;
93     cout << "变量j的地址: " << &j << endl;
94     cout << "变量k的地址: " << &k << endl;
```

```
96
97      //4. 动态内存申请分析
98      //int *p1 = new int[10](10);
99      //int *p2 = new int[10](10,20,30); 错误4
100      初始化数组错误
101      int* p1 = new int[10] {10};
102      int* p2 = new int[10] {10, 20, 30};
103
104      cout << "变量p1地址：" << &p1 << endl;
105      cout << "变量p2地址：" << &p2 << endl;
106
107      cout << "变量p1指向的地址：" << p1 << endl;
108      cout << "变量p2指向的地址：" << p2 << endl;
110
111      system("pause");
112
113      // 错误5 缺少对申请内存空间的释放操作
114      delete[] p1;
115      delete[] p2;
116
117      return 0;
118 }
```

任务二：动态内存分配

```
01 #include <iostream>
02 using namespace std;
03
04 typedef struct sxhnums {
05     int nums;
06     sxhnums* next;
07 } SXH;
08
09 int main() {
10     SXH* head = new SXH;
11     int length = 0;
12     SXH* p = head;
13     for (int i = 100; i < 1000; ++i) {
14         int n = i % 10;
15         int m = i / 10 % 10;
16         int l = i / 100 % 10;
17         if (n * n * n + m * m * m + l * l * l == i) {
18             if (length == 0) {
19                 p->nums = i;
20                 length++;
```

```
            } else {
                SXH* q = new SXH;
                p->next = q;
                q->nums = i;
                length++;
                p = q;
            }
        }
    }
    p = head;
    for (int i = 0; i < length; ++i) {
        cout << p->nums << "\t";
        p = p->next;
    }
    cout << endl;
    system("pause");
    return 0;
}
```

任务三：综合编程分析

```
#include <iostream>
#include <vector>
#include <cmath>
#include <algorithm>

using namespace std;

double maxS = 0;
double maxC = 0;

bool isTrangle(double a, double b, double c) {
    if (a + b > c && a + c > b && b + c > a) {
        return true;
    } else {
        return false;
    }
}

double quadrature(double a, double b, double c) {
    double p = (a + b + c) * 0.5;
    return sqrt(p * (p - a) * (p - b) * (p - c));
}

bool maxS_C(double &a, double &b, double &c) {
```

```
25      if (a + b + c > maxC && quadrature(a, b, c) > maxS) {
26          return true;
27      } else {
28          return false;
29      }
30  }
31
32  int main() {
33      double a = 0;
34      double b = 0;
35      double c = 0;
36      int m;
37      cout << "请输入木棍的数目: m = ";
38      cin >> m;
39      cout << "请输入" << m << "根木棍的长度: ";
40      vector<double> length(m);
41      for (int i = 0; i < m; ++i) {
42          cin >> length[i];
43      }
44  /*************************************************************/
46  // 复杂度太高 n^3
47  //    for (int i = 0; i < m; ++i) {
48  //        for (int j = i + 1; j < m; ++j) {
49  //            for (int k = j + 1; k < m; ++k) {
50  //                if (isTrangle(length[i],length[j],length[k])) {
52  //                    if (maxS_C(length[i],length[j],length[k])) {
54  //                    maxS = quadrature(length[i],length[j],length[k]);
56  //                        maxC = length[i] + length[j] + length[k];
58  //                        a = length[i];
59  //                        b = length[j];
60  //                        c = length[k];
61  //                    }
62  //                }
63  //            }
64  //        }
65  //    }
66  /*************************************************************/
68      // 优化
69      sort(length.begin(), length.end());
70      for (int i = m - 1; i > 1; --i) {
71          if (isTrangle(length[i], length[i - 1], length[i - 2]
72          ) && maxS_C(length[i], length[i - 1], length[i - 2]))
73          {
74              maxS = quadrature(length[i], length[i - 1],
```

```
75              length[i - 2]);
76              maxC = length[i] + length[i - 1] + length[i - 2];
77              c = length[i];
78              b = length[i - 1];
79              a = length[i - 2];
80              break;
81          }
82      }
83      if (maxS == 0.0 && maxC == 0.0) {
84          cout << 0 << endl;
85      } else {
86          cout << "构成的最大三角形周长为: " <<
87          maxC << "\t(选择" << a << "," << b << "," << c <<
88          ")" << endl;
89          cout << "构成的最大三角形面积为: " <<
90          maxS << "\t(选择" << a << "," << b << "," << c <<
91          ")" << endl;
92      }
93      system("pause");
94      return 0;
95  }
```