编号： 2-2

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 23 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. 掌握指向常量指针的用法；
2. 掌握 C++重载函数的基本用法；
3. 掌握 C++内联函数的基本用法；
4. 掌握 C++带有默认形参值函数的用法。

二、实验内容

（一）任务一：const 与指针

建立 VS 项目，设计带有 const 常量指针的函数 double triangleArea(const double *parray)，通过该函数计算输出三角形的面积。

要求主函数通过数组方式传递三角形边长，基于实验结果分析 const 在该函数中的作用。

输入输出样例：

```
01 --------------------------------
02 请依次输入三条边长：3 4 5
03 --------------------------------
04 边长为3,4,5的三角形面积：6
```

（二）任务二：重载函数

建立 VS 项目，重载函数 int commonDivisor(int x1, int x2,...)，计算输出 2 个，3 个，4 个整数的最大公约数。

基于实验结果分析重载函数的特点和用法。

输入输出样例：

```
01 --------------------------------
02 请输入3个整数：6 15 12
03 --------------------------------
04 整数6,15,12的最大公约数为：3
```

（三）任务三：内联函数

建立 VS 项目，设计实现两个功能相同的简短函数(无递归和循环操作)，

将其中一个设置为内联函数。

要求在主函数内调用各函数 10000 次，计算比较两种函数各自的执行时间，基于实验结果分析内联函数和普通函数的区别。

输入输出样例：

```
01 --------------------------------
02 请输入函数执行次数N: 10000
03 --------------------------------
04 普通函数执行N次时间为：xxxx
05 内联函数执行N次时间为：xxxx
```

（四）任务四：带有默认参数值的函数

建立 VS 项目，设计多功能排序函数 double sortArray(double *parray, int flag)，其中 parray 为指向浮点型数组的指针变量，flag 为标志位。

要求该函数默认实现对数组元素从小到大排序（flag 默认为 0），如果 flag 数值为 1，则实现对数组元素从大到小排序方式。

输入输出样例：

```
01 --------------------------------
02 请输入数组元素： 2 8 9 4 6 7 3 1 5
03 --------------------------------
04 *该数组从小到大排序结果：1 2 3 4 5 6 7 8 9
05 *该数组从大到小排序结果：9 8 7 6 5 4 3 2 1
```

三、实验过程

（一）任务一：const 与指针

1. const 在本程序中的作用

在本程序中，定义形参的方式如图 1 。该函数主要是为了计算三角形的面积，在执行过程中，不能出现如图 2 的结果，否则会对 parray 指向的值进行改变，从而违背 const 对它形容的目的。

```
double triangleArea(const double* parray) {
    double p = 0.5 * (parray[0] + parray[1] + parray[2]);
    return sqrt(p * (p - parray[0]) * (p - parray[1]) * (p - parray[2]));
}
```

图 1　const 定义形参的方式

```
parray[0] = 3;
```

图 2　修改 const 修饰的参数报错

2. const 与指针

const 修饰形参的作用主要是为了保护传值。由于以指针的方式传参时，效率比较低。为了提高参数传递的效率，采用引用传参的方式。但是在引用过程中，我们如果不希望改变参数的值，就需要引入 const 修饰形参，从而达到保护形参的目的。

（二）任务二：重载函数

1. 重载函数的特点

(1) 方法名必须相同

(2) 形参的个数不同

(3) 如果形参的个数相同，形参的数据类型必须不同

(4) 函数重载与返回值类型无关

2. 重载函数的用法

(1) 可以对不同的数据类型进行处理，使我们在编程使可以对不同的功能赋予相同的函数名，编译时计算机会根据实参的类型与个数来自动匹配具体的功能。

(2) 函数重载可以解决多种数据类型的计算，可以用于解决预设函数太多，调用时较为麻烦，不好调用等问题。

(3) 具体例子见下图（图 3、图 4）

```
int commonDivisor(int x1, int x2) {
    while (x1 * x2 != 0) {
        if (x1 > x2) {
            x1 %= x2;
        }
        else {
            x2 %= x1;
        }
    }
    return x1 + x2;
}
```

图 3　2 个参数的函数重载

```
int commonDivisor(int x1, int x2, int x3) {
    return commonDivisor(x1, commonDivisor(x2, x2));
}

int commonDivisor(int x1, int x2, int x3, int x4) {
    return commonDivisor(x1, commonDivisor(x2, x3, x4));
}
```

图 4　3 个参数和 4 个参数的函数重载

（三）任务三：内联函数

1. 内联函数与普通函数的区别

(1) 内联含函数比普通函数在前面多一个 inline 修饰符。

(2) 内联函数是直接复制“镶嵌”到主函数中去的。

(3) 普通函数的代码段只有一份，放在内存中的某个位置上，当程序调用它是，指令就跳转过来；当下一次程序调用它是，指令又跳转过来；而内联函数是程序中调用几次内联函数，内联函数的代码就会复制几份放在对应的位置上。

2. 本程序中内联函数的优点

内联函数加快了程序的执行速度，避免了指令的来回跳转。

（四）任务四：带有默认参数值的函数

默认参数的规则

(1) 自右向左，依次赋值。（调用参数传递参数是从左到右的，所以没有默认参数都必须传值进来）

(2) 默认值只能赋值一次。

(3) 默认值可以使用全局变量、函数返回值。

四、实验结果

（一）任务一：const 与指针

运行结果：

1. 三角形 3 边为 3,4,5（能构成三角形）

图 5　任务一能构成三角形的情况

2. 三角形 3 边为 3,4,8（不能构成三角形）

图 6　任务一不能构成三角形的情况

（二）任务二：重载函数

运行结果：

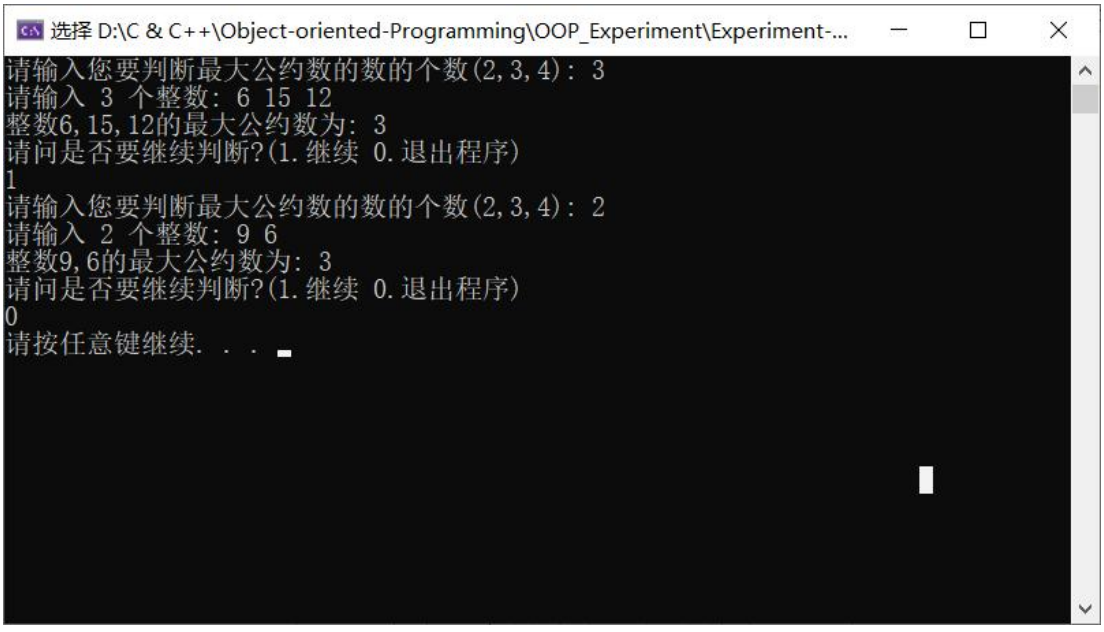

图 7　重载函数的运行结果

（三）任务三：内联函数

运行结果：

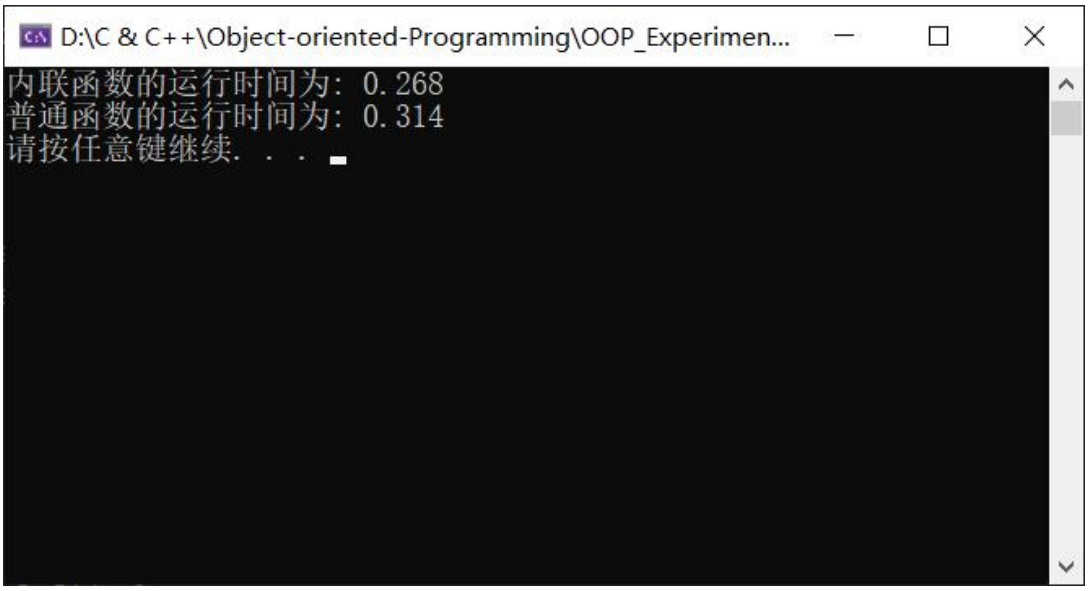

图 8　内联函数和普通函数的运行时间对比

（四）任务四：带有默认参数值的函数

运行结果：

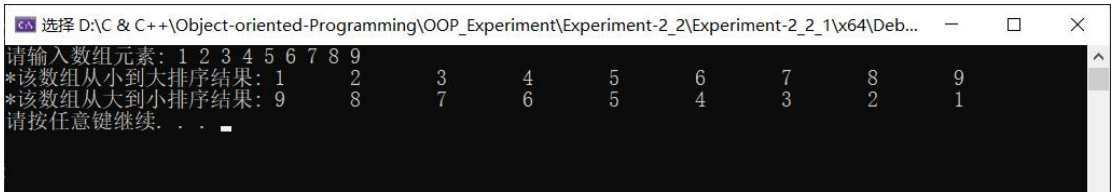

图 9　带有默认参数值的函数的运行结果

五、实验总结

在本实验中，体会了 const 修饰形参的情况，在这种情况下，函数不能修改形参的值，从而达到保护传值的目的。通过对重载函数、内联函数以及带有默认形参值的函数进行实现，理解了重载函数的特征、带有默认形参值的函数的特征和内联函数的高效。

通过本次实验，还了解了一种计算程序运行时间的方法，即通过 clock( ) 函数获取当前程序时间。

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_2 at master · keepIHDR/Object-oriented-Programming (github.com)

任务一： const与指针

```
01 #include <iostream>
02 #include <cmath>
03 using namespace std;
04
05 bool isTriangle(double x, double y, double z) {
06     if (x + y > z && x + z > y && z + y > x) return true;
07     else return false;
08 }
09
10 double triangleArea(const double* parray) {
11     double p = 0.5 * (parray[0] + parray[1] + parray[2]);
12     return sqrt(p * (p - parray[0]) * (p - parray[1]) * (p -
13     parray[2]));
14 }
15
16 int main() {
17     double parray[3];
18     cout << "请依次输入三条边长: ";
19     cin >> parray[0] >> parray[1] >> parray[2];
20     if (!isTriangle(parray[0], parray[1], parray[2])) {
21         cout << "输入的三边边长不能构成三角形! " << endl;
22         system("pause");
23         return 1;
24     }
25     cout << "边长为" << parray[0] << ", " << parray[1] <<
26     ", " << parray[2] << "的三角形面积: " <<
27     triangleArea(parray) << endl;
28     system("pause");
29     return 0;
30 }
```

任务二：重载函数

```
01 #include <iostream>
02 using namespace std;
03
04 int commonDivisor(int x1, int x2) {
05     while (x1 * x2 != 0) {
06         if (x1 > x2) {
07             x1 %= x2;
```

```
        } else {
            x2 %= x1;
        }
    }
    return x1 + x2;
}

int commonDivisor(int x1, int x2, int x3) {
    return commonDivisor(x1, commonDivisor(x2, x2));
}

int commonDivisor(int x1, int x2, int x3, int x4) {
    return commonDivisor(x1, commonDivisor(x2, x3, x4));
}

int main() {
    int select = 0;
    int x1, x2, x3, x4;
    while (1) {
        cout <<
        "请输入您要判断最大公约数的数的个数(
        2,3,4): ";
        cin >> select;
        switch (select) {
        case 2:
            cout << "请输入 " << select << " 个整数: ";
            cin >> x1 >> x2;
            cout << "整数" << x1 << "," << x2 <<
            "的最大公约数为: " << commonDivisor(x1,
            x2) << endl;
            break;
        case 3:
            cout << "请输入 " << select << " 个整数: ";
            cin >> x1 >> x2 >> x3;
            cout << "整数" << x1 << "," << x2 << "," << x3
            << "的最大公约数为: " << commonDivisor(x1,
            x2, x3) << endl;
            break;
        case 4:
            cout << "请输入 " << select << " 个整数: ";
            cin >> x1 >> x2 >> x3 >> x4;
            cout << "整数" << x1 << "," << x2 << "," << x3
            << "," << x4 << "的最大公约数为: " <<
            commonDivisor(x1, x2, x3, x4) << endl;
```

```
            break;
        default:
            cout << "输入数据错误！ " << endl;
            return 0;
        }
        cout << "请问是否要继续判断?(1.继续 0.
        退出程序)" << endl;
        int temp;
        cin >> temp;
        if (temp == 0) {
            system("pause");
            return 0;
        }
    }
}
```

任务三：内联函数

```
#include <iostream>
#include <ctime>

using namespace std;

inline void test01() {
    int a = 1;
    int b = 2;
    int temp = a;
    a = b;
    b = temp;
}

void test02() {
    int a = 1;
    int b = 2;
    int temp = a;
    a = b;
    b = temp;
}

int main() {
    double t1, t2;
    t1 = (double)clock();
    for (int i = 0; i < 100000000; ++i) {
        test01();
    }
```

```
    t2 = (double)clock();
    cout << "内联函数的运行时间为: " << (t2 - t1)
    / CLK_TCK << endl;
    t1 = (double)clock();
    for (int i = 0; i < 100000000; ++i) {
        test02();
    }
    t2 = (double)clock();
    cout << "普通函数的运行时间为: " << (t2 - t1)
    / CLK_TCK << endl;
    system("pause");
    return 0;
}
```

任务四：带有默认参数值的函数

```
#include <iostream>
#include <cstring>
using namespace std;

const int Count = 9;

double sortArray(double* parray, int flag = 0) {
    int n = Count;
    if (flag == 0) {
        for (int i = 0; i < n; ++i) {
            for (int j = 1; j < n - i; ++j) {
                if (parray[j] < parray[j - 1]) {
                    swap(parray[j], parray[j - 1]);
                }
            }
        }
    } else if (flag == 1) {
        for (int i = 0; i < n; ++i) {
            for (int j = 1; j < n - i; ++j) {
                if (parray[j] > parray[j - 1]) {
                    swap(parray[j], parray[j - 1]);
                }
            }
        }
    }
    return 0;
}

int main() {
```

```
30      auto* parray = new double[Count];
31      cout << "请输入数组元素: ";
32      for (int i = 0; i < Count; ++i) {
33          cin >> parray[i];
34      }
35      sortArray(parray);
36      cout << "*该数组从小到大排序结果: ";
37      for (int i = 0; i < Count; ++i) {
38          cout << parray[i] << "\t";
39      }
40      cout << endl;
41      sortArray(parray, 1);
42      cout << "*该数组从大到小排序结果: ";
43      for (int i = 0; i < Count; ++i) {
44          cout << parray[i] << "\t";
45      }
46      cout << endl;
47      system("pause");
48      return 0;
49  }
```