|  |  |
| --- | --- |
| **编号:** | **2-2** |

****

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。

1. **实验目的**
2. 掌握指向常量指针的用法；
3. 掌握C++重载函数的基本用法；
4. 掌握C++内联函数的基本用法；
5. 掌握C++带有默认形参值函数的用法。
6. **实验内容**
7. 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

1. 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

1. 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

1. 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

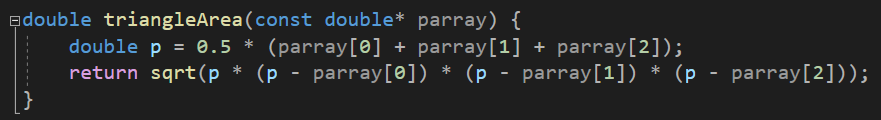
03 --------------------------------

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

1. **实验过程**
2. 任务一：const与指针
3. 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。



**图1 const定义形参的方式**

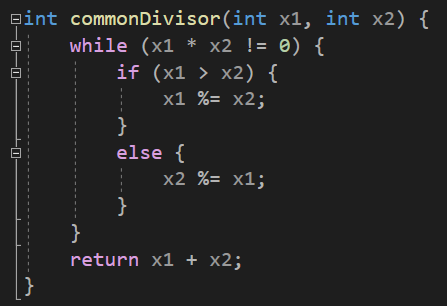


**图2 修改const修饰的参数报错**

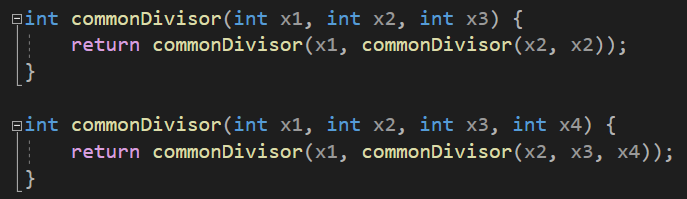
1. const与指针

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。

1. 任务二：重载函数
2. 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
3. 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）

****

**图3 2个参数的函数重载**

****

**图4 3个参数和4个参数的函数重载**

1. 任务三：内联函数
2. 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。
3. 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。

1. 任务四：带有默认参数值的函数

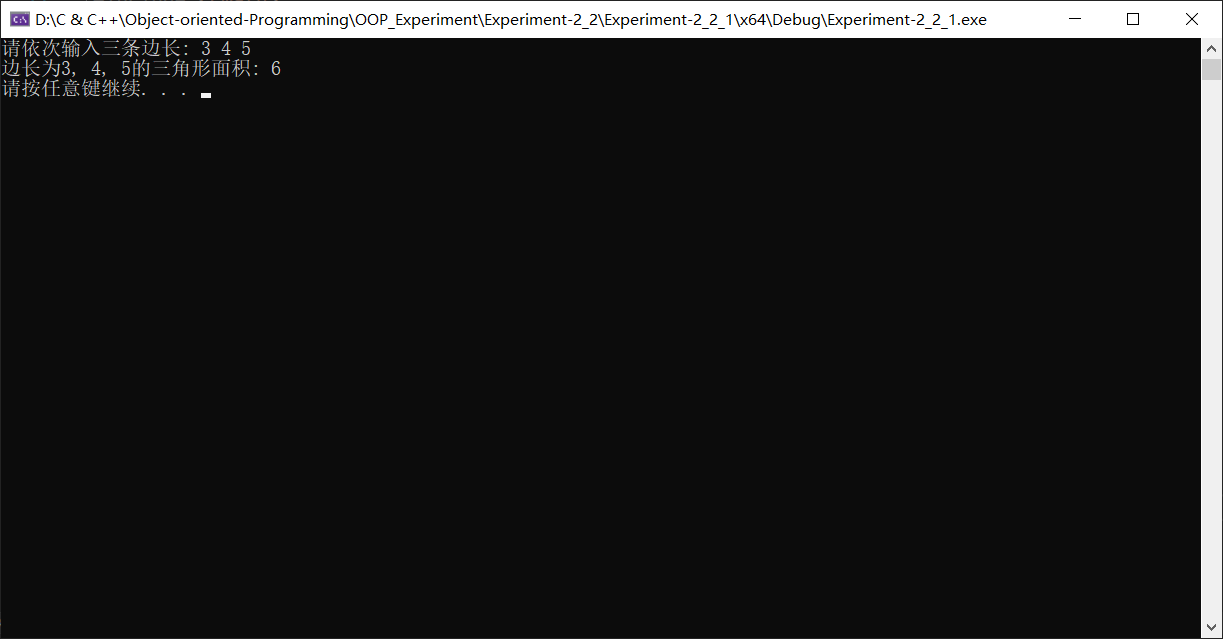
默认参数的规则

* 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。

1. **实验结果**
2. 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**

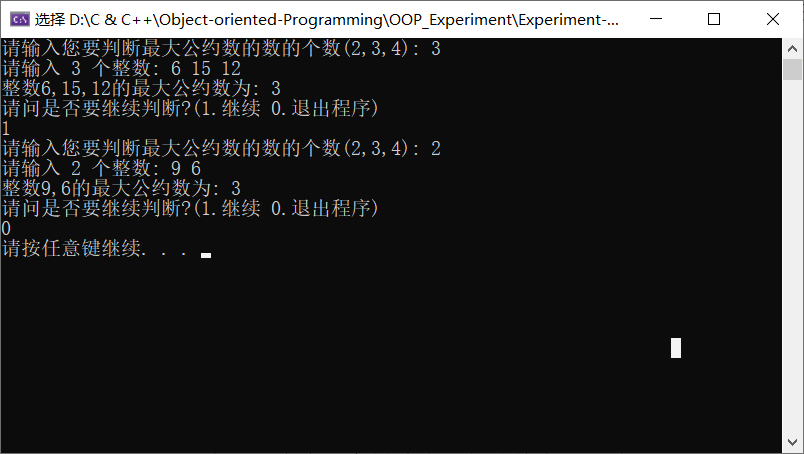
1. 三角形3边为3,4,8（不能构成三角形）

****

**图6 任务一不能构成三角形的情况**

1. 任务二：重载函数

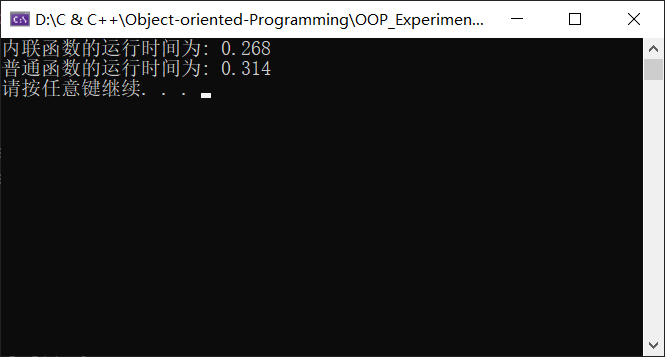
运行结果：

****

**图7 重载函数的运行结果**

1. 任务三：内联函数

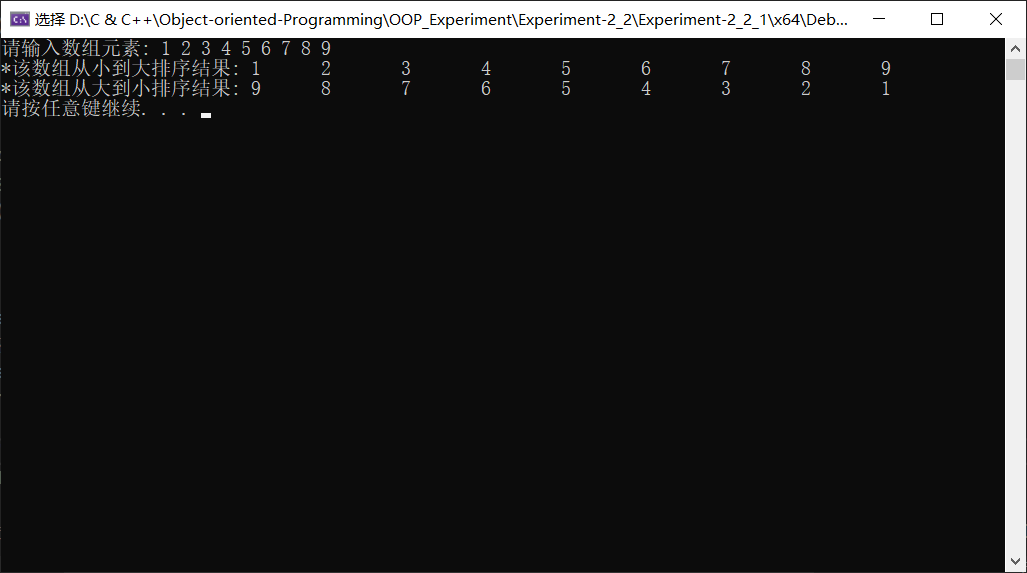
运行结果：

****

**图8 内联函数和普通函数的运行时间对比**

1. 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**

1. **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)](https://github.com/keepIHDR/Object-oriented-Programming/tree/master/OOP_Experiment/Experiment-2_2)
* 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;
* 08 } **else** {
* 09 x2 %= x1;
* 10 }
* 11 }
* 12 **return** x1 + x2;
* 13 }
* 14
* 15 **int** commonDivisor(**int** x1, **int** x2, **int** x3) {
* 16 **return** commonDivisor(x1, commonDivisor(x2, x2));
* 17 }
* 18
* 19 **int** commonDivisor(**int** x1, **int** x2, **int** x3, **int** x4) {
* 20 **return** commonDivisor(x1, commonDivisor(x2, x3, x4));
* 21 }
* 22
* 23 **int** main() {
* 24 **int** select = 0;
* 25 **int** x1, x2, x3, x4;
* 26 **while** (1) {
* 27 cout <<
* 28 "请输入您要判断最大公约数的数的个数(
* 29 2,3,4): ";
* 30 cin >> select;
* 31 **switch** (select) {
* 32 **case** 2:
* 33 cout << "请输入 " << select << " 个整数: ";
* 34 cin >> x1 >> x2;
* 35 cout << "整数" << x1 << "," << x2 <<
* 36 "的最大公约数为: " << commonDivisor(x1,
* 37 x2) << endl;
* 38 **break**;
* 39 **case** 3:
* 40 cout << "请输入 " << select << " 个整数: ";
* 41 cin >> x1 >> x2 >> x3;
* 42 cout << "整数" << x1 << "," << x2 << "," << x3
* 43 << "的最大公约数为: " << commonDivisor(x1,
* 44 x2, x3) << endl;
* 45 **break**;
* 46 **case** 4:
* 47 cout << "请输入 " << select << " 个整数: ";
* 48 cin >> x1 >> x2 >> x3 >> x4;
* 49 cout << "整数" << x1 << "," << x2 << "," << x3
* 50 << "," << x4 << "的最大公约数为: " <<
* 51 commonDivisor(x1, x2, x3, x4) << endl;
* 52 **break**;
* 53 **default**:
* 54 cout << "输入数据错误! " << endl;
* 55 **return** 0;
* 56 }
* 57 cout << "请问是否要继续判断?(1.继续 0.
* 58 退出程序)" << endl;
* 59 **int** temp;
* 60 cin >> temp;
* 61 **if** (temp == 0) {
* 62 system("pause");
* 63 **return** 0;
* 64 }
* 65 }
* 66 }
* 任务三：内联函数
* 01 #include <iostream>
* 02 #include <ctime>
* 03
* 04 **using namespace** std;
* 05
* 06 inline **void** test01() {
* 07 **int** a = 1;
* 08 **int** b = 2;
* 09 **int** temp = a;
* 10 a = b;
* 11 b = temp;
* 12 }
* 13
* 14 **void** test02() {
* 15 **int** a = 1;
* 16 **int** b = 2;
* 17 **int** temp = a;
* 18 a = b;
* 19 b = temp;
* 20 }
* 21
* 22 **int** main() {
* 23 **double** t1, t2;
* 24 t1 = (**double**)clock();
* 25 **for** (**int** i = 0; i < 100000000; ++i) {
* 26 test01();
* 27 }
* 28 t2 = (**double**)clock();
* 29 cout << "内联函数的运行时间为: " << (t2 - t1)
* 30 / CLK\_TCK << endl;
* 31 t1 = (**double**)clock();
* 32 **for** (**int** i = 0; i < 100000000; ++i) {
* 33 test02();
* 34 }
* 35 t2 = (**double**)clock();
* 36 cout << "普通函数的运行时间为: " << (t2 - t1)
* 37 / CLK\_TCK << endl;
* 38 system("pause");
* 39 **return** 0;
* 40 }
* 任务四：带有默认参数值的函数
* 01 #include <iostream>
* 02 #include <cstring>
* 03 **using namespace** std;
* 04
* 05 **const int** Count = 9;
* 06
* 07 **double** sortArray(**double**\* parray, **int** flag = 0) {
* 08 **int** n = Count;
* 09 **if** (flag == 0) {
* 10 **for** (**int** i = 0; i < n; ++i) {
* 11 **for** (**int** j = 1; j < n - i; ++j) {
* 12 **if** (parray[j] < parray[j - 1]) {
* 13 swap(parray[j], parray[j - 1]);
* 14 }
* 15 }
* 16 }
* 17 } **else if** (flag == 1) {
* 18 **for** (**int** i = 0; i < n; ++i) {
* 19 **for** (**int** j = 1; j < n - i; ++j) {
* 20 **if** (parray[j] > parray[j - 1]) {
* 21 swap(parray[j], parray[j - 1]);
* 22 }
* 23 }
* 24 }
* 25 }
* 26 **return** 0;
* 27 }
* 28
* 29 **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 }