

深圳大学实验报告

课程名称： 面向对象程序设计

实验项目名称： 实验四 函数调用方式

学院： 医学院

专业： 生物医学工程

指导教师： 李乔亮、邓云

报告人： 陈焕鑫 学号： 2016222042 班级： 生工 2 班

实验时间： 2018.10.24

实验报告提交时间： 2018.11.1

教务部制

实验目的:

熟练掌握 C++ 中函数的调用方法

实验内容:

1. 采用至少 12 种方法求 $A*A+B*B=C$, 其中 A, B, C 均为整型, 每种方法写一个函数, 在 main 中调用, 并输出结果, 可采用传值, 传址, 传引用的参数传递方式, 及这三种方式的组合。

要求:

- 1) 实验报告中写出自己对函数不同形参传递方式的理解。
- 2) 说明理论上最多有多少种可能。(不必每种都写出)
- 3) 分析所编写的函数中, 哪些能构成重载并说明原因。

2. 利用所有知识, 分析以下程序存在的问题

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

int *Square(int a)
{
    int b;
    b = a * a;

    return &b;
}

void main()
{
    int c = 10;
    int *p;

    p = Square(c);

    cout << *p << endl;
}
```

实验环境与程序代码:

实验环境：win7 系统下的 Visual C++ 6.0

1、程序代码如下所示:

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

int method0(int *a, int &b)
{
    int c;

    c = (*a)*(*a)+b*b;
    return c;
}

int method1(int a, int b)
{
    int c;

    c = a*a+b*b;
    return c;
}

int method2(int &a, int &b)
{
    int c;

    c = a*a+b*b;
    return c;
}

int method3(int *a, int *b)
{
    int c;

    c = (*a)*(*a)+(*b)*(*b);
    return c;
}

int method4(int a, int &b)
{
    int c;

    c = a*a+b*b;
    return c;
}
```

```
}

int method5(int a, int *b)
{
    int c;

    c = a*a+(*b)*(*b);
    return c;
}

int &method5(int a, int b)
{
    int c;

    c = a*a+b*b;
    return c;
}

int &method6(int &a, int &b)
{
    int c;

    c = a*a+b*b;
    return c;
}

void method7(int a, int b, int &c)
{
    c = a*a+b*b;
}

void method8(int &a, int &b, int &c)
{
    c = a*a+b*b;
}

void method9(int a, int b, int *c)
{
    *c = a*a+b*b;
}

void method10(int *a, int *b, int *c)
{
    *c = (*a)*(*a)+(*b)*(*b);
}
```

```

}

void method11(int a, int &b, int *c)
{
    *c = a*a+b*b;
}

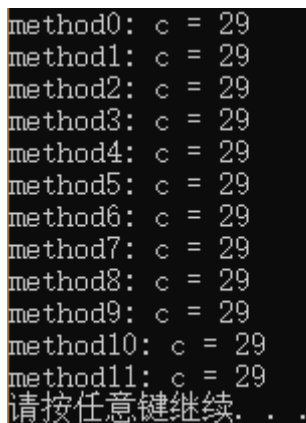
int main()
{
    int a = 2, b = 5, c;
    int &q_a = a, &q_b = b, &q_c = c;
    int *p_a = &a, *p_b = &b, *p_c = &c;

    c = method0(p_a, b);
    cout << "method0: c = " << c << endl;
    c = method1(a, b);
    cout << "method1: c = " << c << endl;
    c = method2(a, b);
    cout << "method2: c = " << c << endl;
    c = method3(p_a, p_b);
    cout << "method3: c = " << c << endl;
    c = method4(a, b);
    cout << "method4: c = " << c << endl;
    c = method5(a, b);
    cout << "method5: c = " << c << endl;
    c = method6(a, b);
    cout << "method6: c = " << c << endl;
    method7(a, b, c);
    cout << "method7: c = " << c << endl;
    method8(a, b, c);
    cout << "method8: c = " << c << endl;
    method9(a, b, p_c);
    cout << "method9: c = " << c << endl;
    method10(p_a, p_b, p_c);
    cout << "method10: c = " << c << endl;
    method11(a, b, p_c);
    cout << "method11: c = " << c << endl;
    system("pause");
    return 0;
}

```

实验结果与分析:

1、程序运行得到的结果如图 1-1 所示



```
method0: c = 29
method1: c = 29
method2: c = 29
method3: c = 29
method4: c = 29
method5: c = 29
method6: c = 29
method7: c = 29
method8: c = 29
method9: c = 29
method10: c = 29
method11: c = 29
请按任意键继续...
```

图 1-1

1) 函数传递参数的方法有三种，分别是传值，传址和传引用。

- 传值的方法，调用函数本身不会对实参进行操作，也就是说，即使形参的值在函数中发生了变化，实参的值也完全不会受到影响，仍为调用前的值。这相当于把实参复制一份给形参，形参调用结束后内存回收。
- 传址的方法与传值的不同在于，它把实参的存储地址传送给对应的形参，从而使得形参指针和实参指针指向同一个地址。因此，被调用函数中对形参指针所指向的地址的内容的任何改变都会影响到实参。
- 传引用的方法，以引用为参数，则既可以使得对形参的任何操作都能改变相应的数据，又使得函数调用显得方便、自然。引用相当于实参的别名，它和实参是同一个变量或数值，它的改变就是实参的改变。

2) 理论上可以写出 45 种不同的函数，其中有 27 种是带返回值的函数，18 种返回类型为 void 的函数函数。27 种之中，返回类型、形参列表中的第一个参数、第二个参数可以是传值，传址和传引用三种方式中任意一种，经过排列组合可以有 $3 \times 3 \times 3 = 27$ 种可能。而 18 种返回类型为 void 的函数中，是将 c 作为形参传递到函数中进行修改，因此，c 可以选择的方式必须是能够对值修改，并在函数结束后还有有效的传递方式，只能是传址或者传引用，其他两个位置仍可以使用三种方式，有 $2 \times 3 \times 3 = 18$ 种。

3) method1, method2 和 method4 之间不能构成重载，method7 和 method8 之间不能构成重载，其他的任意两个函数之间都能构成重载。构成重载的条件是两个函数的函数名相同，参数个数不同或者参数类型不同，参数个数和参数类型相同（传值和传引用虽然方法不同，但只要是类型相同，就算作是参数类型相同，因为在实际调用函数的时候，无法把传值和传引用区分开来），则这两个函数为相同函数，无法重载。

2、函数 Square 中，声明了 int 型的变量 b，函数结束时，会返回 b 的地址。然而，在函数 Square 调用结束之后，原本声明的变量 b 会被释放掉，b 所占用的那部分内存会被系统回收，返回的 b 原来的地址被 p 接收，但是 p 所指向的地址的内容却没有向系统申请内存空间，这就导致不知道系统会在什么时候又使用了这块内存区域，因此，p 指向的内存是非合法的，其地址的内容随时都可能被修改，这叫“悬挂指针”。悬挂指针是很危险的，if 不能判断一个指针是正常指针还是悬挂指针。

指导教师批阅意见：

成绩评定：

指导教师签字：

年 月 日

备注：

- 注： 1、报告内的项目或内容设置，可根据实际情况加以调整和补充。
2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。