

组号: _____



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》

姓名: 王淑贤
学号: 201811010135
班级: 计本 1801
时间: _____

面向对象程序设计实验报告

姓名	王淑贤	班级	计本 1801	学号	201811010135	组号	
时间		地点		周次		页码	
源码	<input type="checkbox"/> 无源码 <input type="checkbox"/> 文档源码 <input type="checkbox"/> 托管源码						

实验报告要求：请围绕实验目的、实验内容、实验过程（图文并茂）、实验结果（高清截图）、实验总结（重点阐述）五个部分进行撰写。报告中若涉及源代码内容，请在附录部分提供完整源码及源码托管地址。

报告撰写完毕后请提交 PDF 格式版本到云班课。

一、实验目的

1.演示教学：c++特色函数、指针、引用

2.掌握 Highlight 代码高亮软件的使用

3.掌握 visual studio 代码调试方法

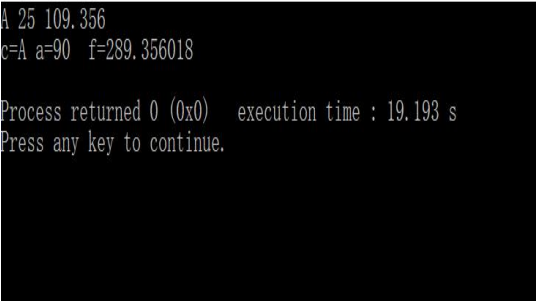
二、实验内容

结合教材第 2 章所有案例代码，分析解答各案例程序后面的 思考与练习 题目，给出必要的 Visual Studio 程序执行结果，并解释问题原因。

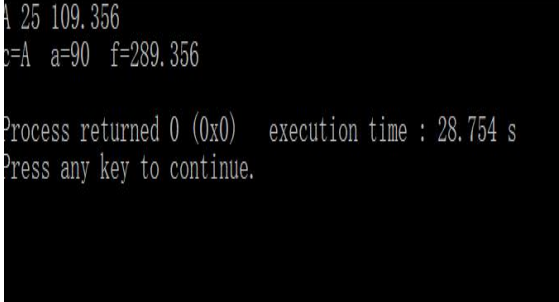
三、实验过程

E310 上机

四、实验结果



example2_01.c



example2_01.cpp

C:\Users\ASUS\Desktop\c语言\fggh\bin\Debug\fggh.exe

```
T1=10 T2=10  
Process returned 0 (0x0) execution time : 0.700 s  
Press any key to continue.
```

example2_02_method1

C:\Users\ASUS\Desktop\c语言\fggh\bin\Debug\fggh.exe

```
T1=10 T2=0  
Process returned 0 (0x0) execution time : 0.844 s  
Press any key to continue.
```

example2_02_method2.cpp

C:\Users\ASUS\Desktop\c语言\fggh\bin\Debug\fggh.exe

```
1 true 1  
Process returned 0 (0x0) execution time : 0.682 s  
Press any key to continue.
```

example2_03.cpp

C:\Users\ASUS\Desktop\c语言\fggh\bin\Debug\fggh.exe

```
10  
200  
-200  
-100  
Process returned 0 (0x0) execution time : 0.742 s  
Press any key to continue.
```

example2_04.cpp

```
i=123  
i=2  
i=2  
i=123  
i=3  
53 63 1  
i=3, sum=117  
Press any key to continue
```

example2_05.cpp

C:\Users\ASUS\Desktop\c语言\fggh\bin\Debug\fggh.exe

```
sum=60  
全局sum=5060  
sum=10120  
sum=200  
全局sum=10320  
Process returned 0 (0x0) execution time : 0.772 s  
Press any key to continue.
```

example2_06.cpp

C:\Users\ASUS\Desktop\c语言\fggh\bin\Debug\fggh.exe

```
20 5 10  
20 30 10  
20 30 40  
Process returned 0 (0x0) execution time : 0.836 s  
Press any key to continue.
```

example2_07.cpp

C:\Users\ASUS\Desktop\c语言\fggh\bin\Debug\fggh.exe

```
a=14  
Process returned 0 (0x0) execution time : 0.790 s  
Press any key to continue.
```

example2_08_1.cpp

C:\Users\ASUS\Desktop\c语言\fggh\bin\Debug\fggh.exe

a=35

Process returned 0 (0x0) execution time : 0.784 s
Press any key to continue.

example2_08_2.cpp

C:\Users\ASUS\Desktop\c语言\fggh\bin\Debug\fggh.exe

square()=2.25
square(10)=100
square(2.5f)=6.25
square(1.1)=1.21

Process returned 0 (0x0) execution time : 0.800 s
Press any key to continue.

example2_09.cpp

C:\Users\ASUS\Desktop\c语言\fggh\bin\Debug\fggh.exe

x=5 y=10 r=5
Address of x 0x4b8000
Address of y 0x4b8004
Address of r 0x4b8000
x=10 y=100 r=10
Address of x 0x4b8000
Address of y 0x4b8004
Address of r 0x4b8000
x=200 y=100 r=200
Address of x 0x4b8000
Address of y 0x4b8004
Address of r 0x4b8000

Process returned 0 (0x0) execution time : 0.811 s
Press any key to continue.

example2_10.cpp

C:\Users\ASUS\Desktop\c语言\fggh\bin\Debug\fggh.exe

a=3 b=5
a=5 b=3
c=10 d=20
c=20 d=10

Process returned 0 (0x0) execution time : 0.890 s
Press any key to continue.

example2_11.cpp

C:\Users\ASUS\Desktop\c语言\fggh\bin\Debug\fggh.exe

a=1 b=2 c=3 d=0
a=1 b=7 c=3 d=7

Process returned 0 (0x0) execution time : 0.801 s
Press any key to continue.

example2_12.cpp

C:\Users\ASUS\Desktop\c语言\fggh\bin\Debug\fggh.exe

a=1 b=2 c=3 d=0
a=1 b=7 c=3 d=0
a=1 b=12 c=3 d=12
a=1 b=20 c=3 d=12

Process returned 0 (0x0) execution time : 0.840 s
Press any key to continue.

example2_13.cpp

C:\Users\ASUS\Desktop\c语言\fggh\bin\Debug\fggh.exe

A -1215
97 a

Process returned 0 (0x0) execution time : 0.835 s
Press any key to continue.

example2_14.cpp

C:\Users\ASUS\Desktop\c语言\fggh\bin\Debug\fggh.exe

a/b=2
b/a=0
except of divide zero
calculate finished

Process returned 0 (0x0) execution time : 0.882 s
Press any key to continue.

example2_15.cpp

五、实验总结

无

■ 附录：程序源码

/*C 语言风格的源程序 example2_01.c，实现读入几个变量，运算后再输出*/

```
#include <stdio.h>
int main()
{   char c;                /*定义变量 c、a、f */
    int a;
    float f;
    scanf("%c%d%f",&c,&a,&f);    /*输入变量 c、a、f 的值*/
    a=a+c;
    f=f+2*a;
    printf("c=%c a=%d f=%f\n",c,a,f);    /*输出变量 c、a、f 的值*/
    return 0;
}
```

/*C++源程序 example2_01.cpp，与 example2_01.c 实现同样的功能*/

```
#include <iostream>
using namespace std;
int main()
{   char c ;                //定义变量 c、a、f 的值
    int a ;
    float f ;
    cin>>c>>a>>f ;        //输入变量 c、a、f 的值
    a=a+c;
    f=f+2*a;
    cout<<"c="<<c<<"  a="<<a<<"  f="<<f<<endl ;    //输出变量 c、a、f 的值
    return 0;
}
```

//example2_02_method1: 两种符号常量定义的区别, 方法 1, 用宏定义

```
#include <iostream>
using namespace std;
int main()
{   int x=5;
```

```

#define T1 x+x //行 5, 用宏定义定义符号常量 T1
#define T2 T1-T1 //行 6, 用宏定义定义符号常量 T2
cout<<"T1="<<T1<<" T2="<<T2<<endl;
return 0 ;
}

```

//example2_02_method2.cpp: 两种符号常量定义的区别, 方法 2: 用 const 定义符号常量

```

#include <iostream>
using namespace std;
int main()
{
    int x=5;
    const int T1=x+x ; //行 5, 用 const 定义符号常量 T1
    const int T2=T1-T1; //行 6, 用 const 定义符号常量 T2
    cout<<"T1="<<T1<<" T2="<<T2<<endl;
    return 0 ;
}

```

//example2_03.cpp: 布尔型示例

```

#include <iostream>
using namespace std; //该指令涵盖了 C++ 的名字空间中所有的标识符
int main()
{
    bool f=1<2;
    cout<<f<<" "<<boolalpha<<f<<" "<<noboolalpha<<f<<endl;
    return 0;
}

```

//example2_04.cpp: 名字空间使用示例

```

#include <iostream>
using namespace std; //using 声明使用一个完整的名字空间 std, C++ 中提供的名字
//空间 std 涵盖了所有标准 C++ 的定义和声明
namespace one //定义一个名字空间 one, 里面有 1 个常量 M 和 1 个变量 inf
{
    const int M=200;
    int inf=10;
} //后面未加分号
namespace two //定义一个名字空间 two, 里面有 2 个变量: x 和 inf
{
    int x;
    int inf=-100 ;
} //后面未加分号
using namespace one ; //方法 1: using 声明使用一个完整的名字空间 one
int main()
{
    using two::x ; //方法 3: using 声明仅使用名字空间 two 中的内容 x
    x=-100 ; //直接访问, 相当于 two::x=-100;
    cout<<inf<<endl; //using 声明使用了整个 one, 其所有成员直接访问
    cout<<M<<endl;
}

```

```

    two::inf*=2;                //方法 2: 使用名字空间名::局部内容名操作未使用 using 声明的内容
    cout<<two::inf<<endl;      //同样是 two 中的内容, 但是访问方式不一样
    cout<<x<<endl;             //已用 using 声明了 two 中内容 x, 可以直接访问
    return 0;
}

```

//example2_05.cpp: 局部变量随用随定义及作用域问题示例。

```

#include <iostream>
#include <iomanip>
using namespace std;          //使用 C++的标准名字空间
int main()
{
    int arr[3]={0}, i=123;      //定义第一个局部变量 i, 作用域不含第二个 i 所在域
    cout<<"i="<<i<<endl;      //输出第一个 i 的值为 123

    {
        //该语句块内, 有同名的局部变量 i, 则第一个 i 无作用
        for (int i=0; i<2; i++) //定义第二个局部变量 i, 作用域仅限于该语句块
            arr[i]=(i+5)*10+3;
        cout<<"i="<<i<<endl;    //输出第二个局部变量 i 的值 2, 第一个 i 不起作用
        arr[2]++;
        cout<<"i="<<i<<endl;    //输出第二个局部变量 i 的值 2, 第一个 i 不起作用
    }
    //第二个局部变量 i 的作用域到此结束

    cout<<"i="<<i<<endl;        //输出的为第一个 i 的值 123
    int sum=0;                  //定义局部变量 sum, 随用随定义
    for (i=0; i<3; i++)         //第一个 i 在起作用
        sum+=arr[i];
    cout<<"i="<<i<<endl;        //输出的为第一个 i 的值 3
    for (i=0; i<3; i++)         //第一个 i 作为循环控制变量
        cout<<setw(4)<<arr[i];  //setw 控制格式, 后面按每个元素占 4 列输出
    cout<<endl;
    cout<<"i="<<i<<endl, sum="<<sum<<endl; //输出第一个 i 的值 3 以及 sum 的值 117
    return 0;
}

```

//example2_06.cpp: 域解析符使全局变量真正具有全局作用域

```

#include <iostream>
#include <iomanip>
using namespace std;          //使用 C++的标准名字空间
int sum=5000;                 //定义全局变量 sum
int main()
{
    int arr[3]={10, 20, 30};
}

```

```

{
    //一个小程序块的开始
    int i, sum=0;           //定义局部变量 sum
    for (i=0; i<3; i++)
        sum+=arr[i];       //求和, 结果存于局部变量 sum 中
    cout<<"sum="<<sum<<endl; //输出局部变量 sum 的值
    ::sum+=sum;             //通过域解析符在同名局部变量的作用域内对全局 sum
访问
    cout<<"全局 sum="<<::sum<<endl; //输出全局 sum 变量的值
}
//小程序块的结束
sum*=2;                    //这是全局 sum, 因为局部变量 sum 的作用域已结束
cout<<"sum="<<sum<<endl;      //输出全局 sum 变量的值
int sum=200;               //定义另一个局部变量 sum
cout<<"sum="<<sum<<endl;      //输出刚定义的局部变量 sum 的值
::sum+=sum;                //通过域解析符使全局 sum 在同名局部 sum 的作用域可
见
    cout<<"全局 sum="<<::sum<<endl; //输出全局 sum 变量的值
return 0;
}

```

//example2_07.cpp: 形式参数带默认参数值的函数定义及调用示例

```

#include <iostream>
using namespace std;
void Fun(int i, int j=5, int k=10) ; //原型声明中形参 j 和 k 分别指定了默认参数值 5 和 10
int main()
{
    Fun(20) ;                    //实际参数个数少于形式参数, 20 与 i 对应, 至少有一个
实参
    Fun(20, 30) ;                //形式参数 j 和 k 分别使用默认参数值 5 和 10
    Fun(20, 30, 40) ;            //形式参数 k 使用默认参数值 10
    return 0;                    //实际参数个数等于形参个数, 都不使用默认参数值
}
void Fun(int i, int j, int k)    //原型中已指定了默认参数值, 在定义的首部不能再指定
{
    cout<<i<<" "<<j<<" "<<k<<endl;
}

```

//example2_08_1.cpp: 用宏定义实现两数相乘

```

#include <iostream>
using namespace std;
#define Multiply(x,y) x*y //注意:此处 x 和 y 两边未加括号
int main()
{

```



```

    int a=Multiply(3+4,2+3); //展开后为:int a=3+4*2+3
    cout<<"a="<<a<<endl;
    return 0;
}

//example2_08_2.cpp:用内联函数实现两数相乘
#include <iostream>
using namespace std;
inline int Multiply(int x,int y)
{
    return x*y;
}
int main()
{
    int a=Multiply(3+4,2+3);
    cout<<"a="<<a<<endl;
    return 0;
}

//example2_09.cpp: 重载函数示例
#include <iostream>
using namespace std;
int square(int x)                                //重载函数的第 1 版本, int 型参数
{
    return x*x;
}
float square(float x )                          //重载函数的第 2 版本, float 型参数
{
    return x*x;
}
double square(double x=1.5 )                    //重载函数的第 3 版本, double 型参数
{
    return x*x;
}
int main()
{
    cout<<"square()="<<square()<<endl;          //调用第 3 版本函数,用默认值,结果为 2.25
    cout<<"square(10)="<<square(10)<<endl;      //调用第 1 版本函数, 结果为 100
    cout<<"square(2.5f)="<<square(2.5f)<<endl;  //调用第 2 版本函数, 结果为 6.25
    cout<<"square(1.1)="<<square(1.1)<<endl;    //调用第 3 版本函数, 结果为 1.21
    return 0;
}

//example2_10.cpp: 引用的声明及使用示例
#include <iostream>
using namespace std;
int x=5,y=10;
int &r=x;                                         //定义一个引用 r 作为变量 x 的别名
void print()                                    //定义一个专门用于输出的函数
{
    cout<<"x="<<x<<" y="<<y<<" r="<<r<<endl;    //输出 x、y、r 的值
    cout<<"Address of x "<<&x<<endl;            //输出变量 x 的内存地址
    cout<<"Address of y "<<&y<<endl;            //输出变量 y 的内存地址
}

```

```

        cout<<"Address of r "<<&r<<endl ;           //输出引用 r 的内存地址
    }
    int main()
    {
    print();           //第 1 次调用输出函数
        r=y ;         //相当于 x=y，将 y 的值赋给 x
                        //而不是 r 改变为变量 y 的别名
        y=100 ;        //对 y 重新赋值不会影响引用 r 的值
        print() ;      //再次调用输出函数
    x=200;             //对 x 重新赋值，r 随之改变，不会影响变量 y 的
    值
    print() ;          //再次调用输出函数
    return 0;
    }

//example2_11.cpp: 用引用作参数修改对应实际参数变量的值
#include <iostream>
using namespace std;
void swap(int &x, int &y)           //调用之初参数传递就使得引用参数成为
                                    //本次调用对应实际参数变量的别名
{
    int t=x;           //这里的引用 x 和 y 分别是两个实际参数变量
    x=y;               //的别名，因此这 3 条语句实际上实现了
    y=t;               //实际参数变量值的互换
}
int main()
{
    int a=3,b=5,c=10,d=20;
    cout<<"a"<<a<<" b"<<b<<endl ;    //输出交换前的 a、b 值
    swap(a,b);           //调用函数，参数传递相当于执行了
                        //int &x=a; int &y=b;使引用参数获得了初值
    cout<<"a"<<a<<" b"<<b<<endl;    //输出交换后的 a、b 值
    cout<<"c"<<c<<" d"<<d<<endl;    //输出交换前的 c、d 值
    swap(c,d) ;          //调用函数，参数传递相当于执行了
                        //int &x=c; int &y=d; 使引用参数获得了初值
    cout<<"c"<<c<<" d"<<d<<endl;    //输出交换后的 c、d 值
    return 0;
}

//example2_12.cpp: 引用参数前面加 const 修饰符保证对应实际参数变量不能被修改
#include <iostream>
using namespace std;
int Fun(const int &x, int &y, int z)           //对第 1 个实际参数变量保护，只能访问不能修
改
{
    // x++ ;           //此句若作为函数的语句，则报错，用 const 限制后只能
                        //访问 x，不能修改 x

```

```

    z++;
    y=x+y+z;
    return y;
}
//对值形式参数的修改不会影响对应的实际参数变量
//通过修改 y 改变第 2 个实际参数变量的值

int main()
{
    int a=1,b=2,c=3,d=0;
    cout<<"a="<<a<<" b="<<b<<" c="<<c<<" d="<<d<<endl;
    d=Fun(a,b,c);
    cout<<"a="<<a<<" b="<<b<<" c="<<c<<" d="<<d<<endl;
    return 0;
}

```

//example2_13.cpp: 引用返回函数的定义及三种调用示例

```

#include <iostream>
using namespace std;
int& Fun(const int &x,int &y,int z)    //返回引用的函数
{
    z++;
    y=x+y+z;
    return y;
}
//对值形式参数的修改不会影响对应实际参数变量
//通过修改 y 改变第 2 个实际参数变量的值
//返回的是引用参数，实际上是对应实参变量

int main()
{
    int a=1,b=2,c=3,d=0;
    cout<<"a="<<a<<" b="<<b<<" c="<<c<<" d="<<d<<endl;
    Fun(a,b,c);
    cout<<"a="<<a<<" b="<<b<<" c="<<c<<" d="<<d<<endl;
    d=Fun(a,b,c);
    cout<<"a="<<a<<" b="<<b<<" c="<<c<<" d="<<d<<endl;
    Fun(a,b,c)=20;
    cout<<"a="<<a<<" b="<<b<<" c="<<c<<" d="<<d<<endl;
    return 0;
}

```

//example2_14.cpp: void 类型指针用法示例

```

#include <iostream>
using namespace std;
int main( )
{
    void *vp;
    char c='A';
    short int x=97;
    vp=&c;
    cout<<*(char *)vp<<" "<<*(short int*)vp<<endl;
}
//char 类型的指针可以直接赋值给 void 类型的指针变量

```

```

    vp=&x;                //short int 类型的指针可以直接赋值给 void 类型的指针变量
                           //需要显式类型转换输出 void 类型指针所指向的内容
    cout<<*(short int *)vp<<" "<<*(char*)vp<<endl;
    return 0;
}

//example2_15.cpp: C++的异常处理过程和方法示例
#include <iostream>
using namespace std;
int divide(int x,int y)
{   if (y==0) throw y;      //如果分母为零，抛出异常
    return x/y;
}
int main()
{   int a=10,b=5,c=0;
    try                    //检查是否出现异常
    {   cout<<"a/b="<<divide(a,b)<<endl;
        cout<<"b/a="<<divide(b,a)<<endl;
        cout<<"a/c="<<divide(a,c)<<endl;
        cout<<"c/b="<<divide(c,b)<<endl;
    }
    catch(int)             //捕获异常并作出处理，即输出一条提示信息
    {   cout<<"except of divide zero"<<endl;
    }
    cout<<"calculate finished"<<endl;    //catch 块的后续语句
    return 0;
}

```

④：根据实际撰写内容自行拓展页面，作业内容尾部尽量不要留有空白