北京化工大学

Beijing University of Chemical Technology

面向对象程序设计 (CSE24312C)

课程代码

Cheng Yong

目录

创建日期：2020 年 8 月 9 日

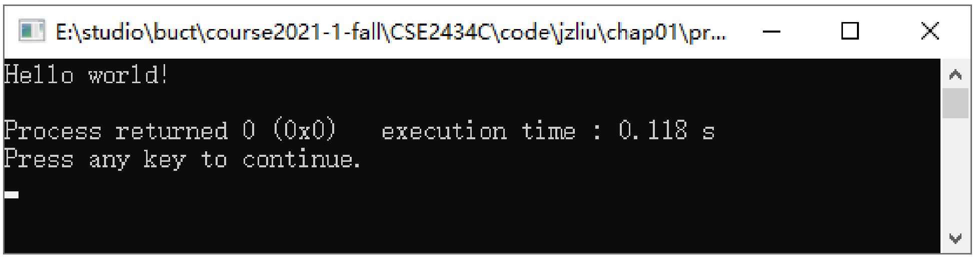
更新日期：2020 年 8 月 10 日

第 1 章　绪论

1.1　ex01p1-输出一行字符 “Hello, C++”

```
// prog0101(pp.1) 输出一行字符"Hello, C++"

#include <iostream> //包含头文件iostream

using namespace std; //使用命名空间std

int main( )
{
   cout<<"Hello, C++.\n";
   return 0;
}
```

运行结果:



E:\studio\buct\course2021-1-fall\CSE2434C\code\jzliu\chap01\pr...

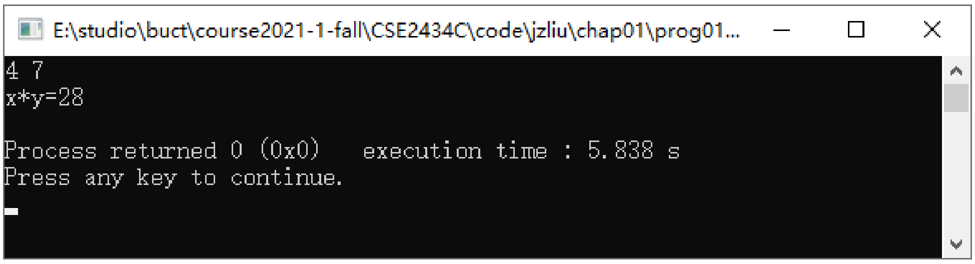
```
Hello world!

Process returned 0 (0x0)   execution time : 0.118 s
Press any key to continue.
```

图 1.1: 运行结果

1.2　ex02p2-输入两个数 x 和 y，然后求两数之积

```
#include <iostream> //预处理命令


using namespace std; //使用命名空间std


int main( ) //主函数首部
{
   //函数体开始
   int x,y,cj; //定义变量
   cin>>x>>y; //输入语句
   cj=x*y;//赋值语句
   cout<<"x*y="<<cj<<endl; //输出语句
   return 0; //如程序正常结束,向操作系统返回一个零值
} //函数结束
```

运行结果:


E:\studio\buct\course2021-1-fall\CSE2434C\code\jzliu\chap01\prog01...

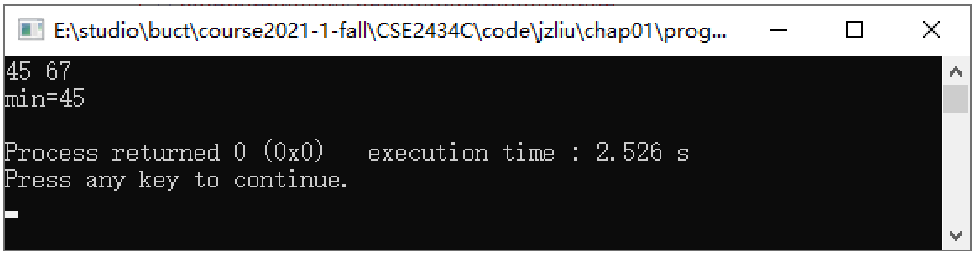
```
4 7
x*y=28

Process returned 0 (0x0)   execution time : 5.838 s
Press any key to continue.
```

图 1.2: 运行结果

1.3 ex03p3-输入两个数 x 和 y，求两者之间的最小者

```
#include <iostream> //预处理命令
using namespace std;
int min(int x,int y)//定义min函数,函数值为整型,形式参数x,y为整型
{
   //min函数体开始
   int z;//变量声明,定义本函数中用到的变量z为整型
   if(x>y) z=y;//if语句,如果x>y,则将y的值赋给z
   else z=x; //否则,将x的值赋给z
   return(z);//将z的值返回,通过min带回调用处
} //min函数结束

int main( ) //主函数
{
   //主函数体开始
   int a,b,m;//变量声明
   cin>>a>>b;//输入变量a和b的值
   m=min(a,b); //调用min函数,将得到的值赋给m
   cout<<"min="<<m<<endl;//输出大数m的值
   return 0; //如程序正常结束,向操作系统返回一个零值
} //主函数结束
```

运行结果:


E:\studio\buct\course2021-1-fall\CSE2434C\code\jzliu\chap01\prog...

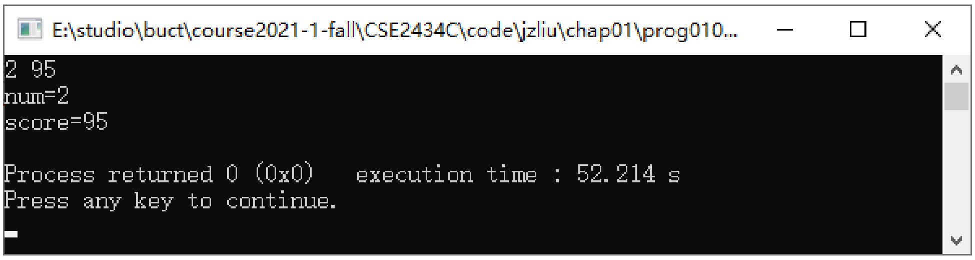
```
45 67
min=45

Process returned 0 (0x0)   execution time : 2.526 s
Press any key to continue.
```

图 1.3: 运行结果

1.4 ex04p3-包含类的 C++ 程序

```
#include <iostream>// 预处理命令

using namespace std;
class Student// 声明一个类,类名为Student
{
private: // 以下为类中的私有部分
    int num; // 私有变量num
    int score; // 私有变量score
public: // 以下为类中的公用部分
    void inputdata( ) // 定义公用函数inputdata
    {
        cin>>num; // 输入num的值
        cin>>score; // 输入score的值
    }
    void outputdata( ) // 定义公用函数outputdata
    {
        cout<<"num="<<num<<endl; // 输出num的值
        cout<<"score="<<score<<endl;//输出score的值
    };
}; // 类的声明结束


Student s1; //定义s1为Student类的变量,称为对象

int main( )// 主函数首部
{
    s1.inputdata( ); // 调用对象s1的inputdata函数
    s1.outputdata( ); // 调用对象s1的outputdata函数
    return 0;
}
```

运行结果:


E:\studio\buct\course2021-1-fall\CSE2434C\code\jzliu\chap01\prog010...

```
2 95
num=2
score=95

Process returned 0 (0x0)   execution time : 52.214 s
Press any key to continue.
```

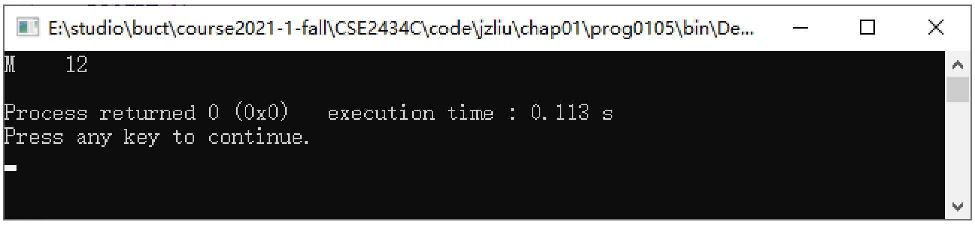
图 1.4: 运行结果

1.5 ex05p7-字符串常量的使用

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

```
int main( )
{
    char ch ='M';   //语句1
    int i =ch-'A';   //语句2
    cout<<ch <<" "<<i<<endl; //语句4
    return 0;
}
```

运行结果：



```
M    12

Process returned 0 (0x0)   execution time : 0.113 s
Press any key to continue.
```

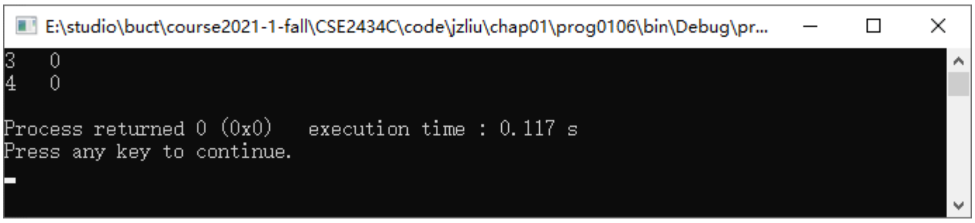
E:\studio\buct\course2021-1-fall\CSE2434C\code\jzliu\chap01\prog0105\bin\De...

图 1.5: 运行结果

1.6 ex06p11-逻辑运算符短路表达式

```
#include <iostream>

using namespace std;

int main()
{
    int a=3, b=0;
    !a && a+b && a++;
    cout<<a<<" "<<b<<endl;
    !a||a++||b++;
    cout<<a<<" "<<b<<endl;
    return 0;
}
```

运行结果：



```
3   0
4   0

Process returned 0 (0x0)   execution time : 0.117 s
Press any key to continue.
```

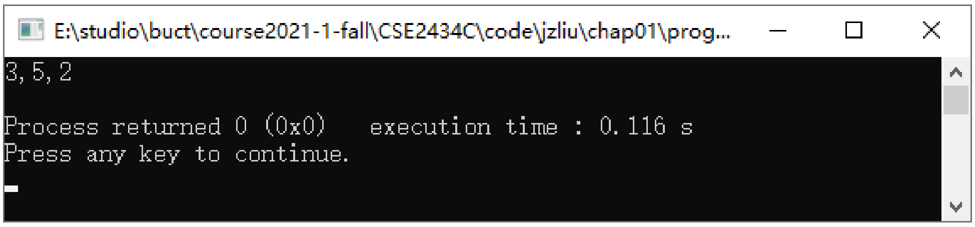
E:\studio\buct\course2021-1-fall\CSE2434C\code\jzliu\chap01\prog0106\bin\Debug\pr...

图 1.6: 运行结果

1.7 ex07p12-条件表达式的使用

```
#include <iostream>

using namespace std;

int main()
{
   int a,b,c;
   a=3; b=4;
   c=a>b? ++a:++b;
   cout<<a<<","<<b<<","<<c<<endl;
   c=a-b?a-3?b:b-a:a;
   cout<<a<<","<<b<<","<<c<<endl;
   return 0;
}
```

运行结果：


E:\studio\buct\course2021-1-fall\CSE2434C\code\jzliu\chap01\prog...

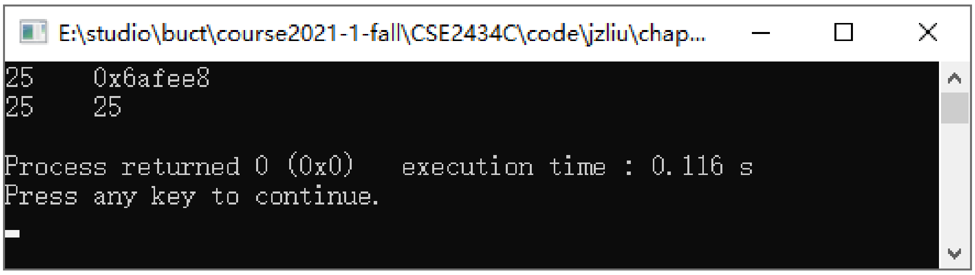
```
3,5,2

Process returned 0 (0x0)   execution time : 0.116 s
Press any key to continue.
```

图 1.7: 运行结果

1.8 ex08p13-引用和变量的关系

```
#include <iostream>
#include <iomanip>

using namespace std;

int main( )
{ int a=25;
     int &b=a; //声明b是a的引用
     cout<<a<<" "<<&b<<endl;
     cout<<a<<" "<<b<<endl;
     return 0;
}
```

运行结果：


E:\studio\buct\course2021-1-fall\CSE2434C\code\jzliu\chap...

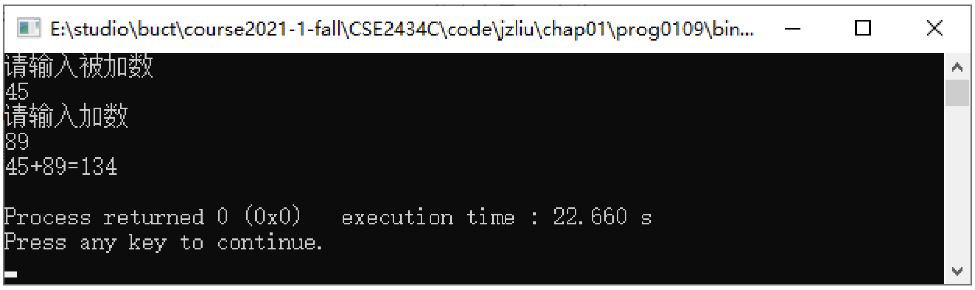
```
25      0x6afee8
25      25

Process returned 0 (0x0)   execution time : 0.116 s
Press any key to continue.
```

图 1.8: 运行结果

1.9 ex09p17-输出两数和的对象化 C++ 程序

```
#include <iostream>
using namespace std;
class sum
{
private: //私有数据成员定义区
   int m,n;
public:
   void he()
   {
      cout <<"请输入被加数"<<endl; //提示输入
      cin >>m; //输入变量m之值
      cout <<"请输入加数"<<endl; //提示输入
      cin >>n; //输入变量n之值
      cout <<m<<"+"<<n<<"="<<m+n<<endl; //输出变量m+n之值
   };
};

sum s;
int main()
{
   //主程序开始
   s.he() ;
   return 0;
}
```

运行结果：


E:\studio\buct\course2021-1-fall\CSE2434C\code\jzliu\chap01\prog0109\bin...

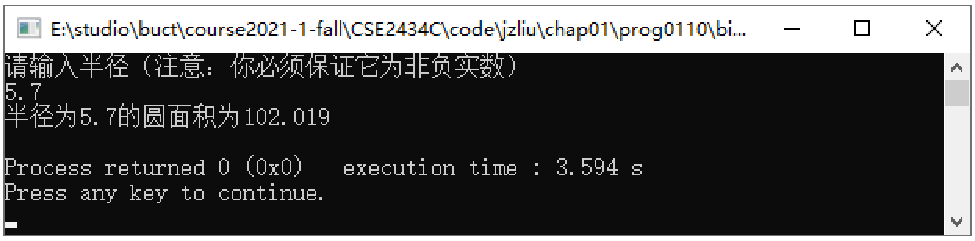
```
请输入被加数
45
请输入加数
89
45+89=134

Process returned 0 (0x0)   execution time : 22.660 s
Press any key to continue.
```

图 1.9: 运行结果

1.10 ex10p18-求圆面积的 C++ 程序

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

class cir
{
private: //私有数据成员定义区
    float r; //实型数据成员定义
public: //公有函数成员定义区
    void mj()
    {
        cout <<"请输入半径（注意：你必须保证它为非负实数）"<<endl; //提示输入
        cin>>r;  //输入半径
        cout<<"半径为"<<r<<"的圆面积为"<<3.14*r*r<<endl;
    };
};

cir c;

int main()
{
    //主程序开始
    c.mj();
    return 0;
}
```

运行结果：


E:\studio\buct\course2021-1-fall\CSE2434C\code\jzliu\chap01\prog0110\bi...

```
请输入半径（注意：你必须保证它为非负实数）
5.7
半径为5.7的圆面积为102.019

Process returned 0 (0x0)   execution time : 3.594 s
Press any key to continue.
```

图 1.10: 运行结果

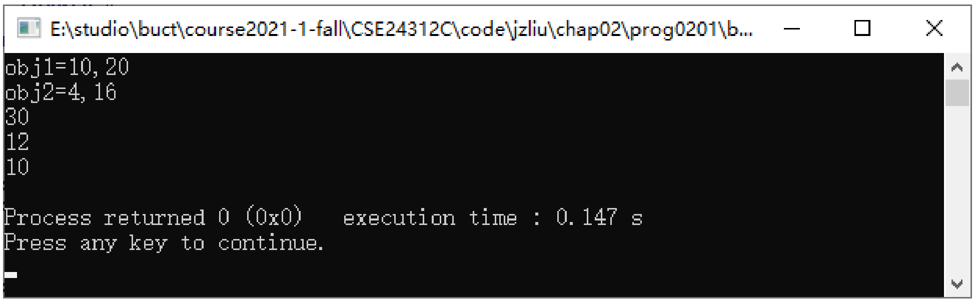
第 2 章　面向对象的程序设计方法概述

2.1　ex01p27-成员函数在类体外定义示例

```
#include <iostream>

using namespace std;
class Test
{
private:
   int X , Y ;
public:
   void setVal(int x) ;     //成员函数重载
   void setVal(int x , int y);
   int add(int x , int y);    //成员函数重载
   int add(int x);
   int add();
   int getX()
   {
      return X ;
   }
   int getY()
   {
      return Y ;
   }
};
void Test::setVal(int x)
{
   X=x;
   Y=x*x;
}
void Test::setVal(int x , int y)
{
   X=x;
   Y=y;
}
int Test::add(int x , int y)
{
   X=x;
   Y=y;
   return X+Y;
}
int Test::add(int x)
{
   X=Y=x;
   return X+Y;
}
int Test::add( )
```

```
{
   return X+Y;
}
int main()
{
   Test obj1,obj2;
   obj1.setVal(10,20);
   obj2.setVal(4);
   cout<<"obj1="<<obj1.getX( )<<','<<obj1.getY( )<<endl;
   cout<<"obj2="<<obj2.getX( )<<','<<obj2.getY( )<<endl;
   int i =obj1.add( );
   int j =obj2.add(3,9);
   int k =obj2.add(5);
   cout<<i <<endl<<j<<endl<<k<<endl;
   return 0;
}
```

运行结果:



E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap02\prog0201\b...

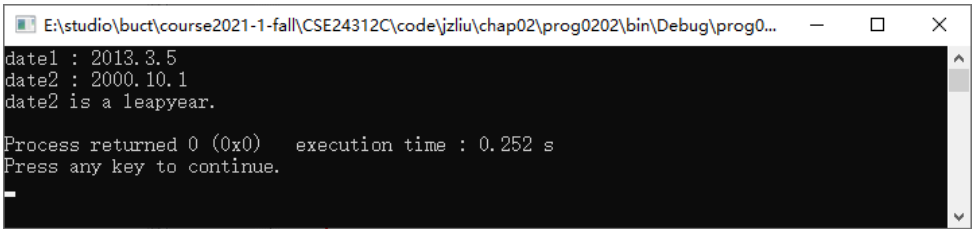
```
obj1=10,20
obj2=4,16
30
12
10

Process returned 0 (0x0)   execution time : 0.147 s
Press any key to continue.
```

图 2.1: 运行结果

2.2 ex02p29-类的定义 (日期和闰年判定)

```
#include <iostream>

using namespace std;

class Date
{
private:
   int year , month , day;
public:
   void SetDate(int y =2013, int m =3 , int n =5);
   int IsLeapYear( );
   void Print( )
   {
      cout <<year <<"."<<month<<"."<<day<<endl;
   }
};
void Date::SetDate( int y , int m , int d )
{
   year=y;
   month=m;
   day=d;
}
int Date::IsLeapYear( )
{
   return (year%4==0&&year%100!=0)||(year%400==0);
}
int main( )
{
   Date date1,date2;
   date1.SetDate( );     //默认参数值
   date2.SetDate(2000 , 10 , 1);
   cout<<"date1 : ";
   date1.Print( );
   cout <<"date2 : ";
   date2.Print( );
   if (date2.IsLeapYear( ) )
      cout <<"date2 is a leapyear."<<endl;
   else
      cout <<"date2 is a leapyear."<<endl;
   return 0;
}
```

运行结果：


E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap02\prog0202\bin\Debug\prog0...

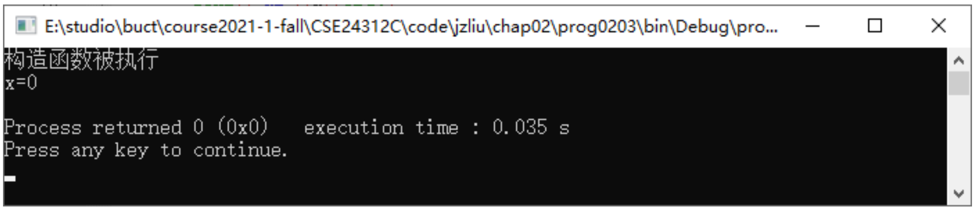
```
date1 : 2013.3.5
date2 : 2000.10.1
date2 is a leapyear.

Process returned 0 (0x0)   execution time : 0.252 s
Press any key to continue.
```

图 2.2: 运行结果

2.3 ex03p31-使用构造函数初始化对象 (构造函数不带参数)

```
#include <iostream>

using namespace std;

class Test
{
private:
   int x;
public:
    Test()
    {
        cout<<"构造函数被执行"<<endl;
        x=0;
    }
    void print()
    {
        cout<<"x="<<x<<endl;
    }
};

int main()
{
    Test obj1;
    obj1.print();
    return 0;
}
```

运行结果:



E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap02\prog0203\bin\Debug\pro...

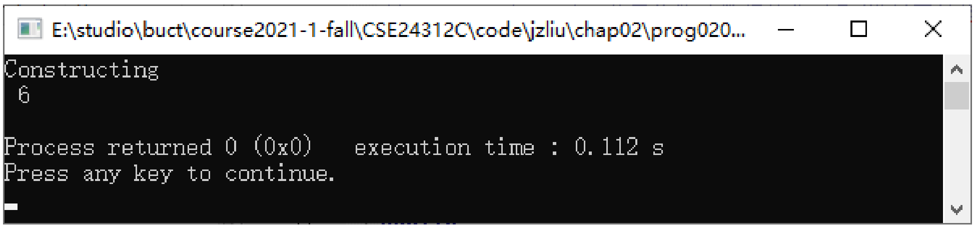
```
构造函数被执行
x=0

Process returned 0 (0x0)   execution time : 0.035 s
Press any key to continue.
```

图 2.3: 运行结果

2.4 ex04p31-使用构造函数初始化对象 (通过带参构造函数初始化)

```
#include <iostream>

using namespace std;

class example
{
private:
    int x;
public:
    example(int n)
    {
        x=n;
        cout<<"Constructing\n ";
    }

    int getX()
    {
        return x;
    }
};

int main()
{
    example m(6);
    cout<<m.getX()<<endl;
    return 0;
}
```

运行结果：



E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap02\prog020...

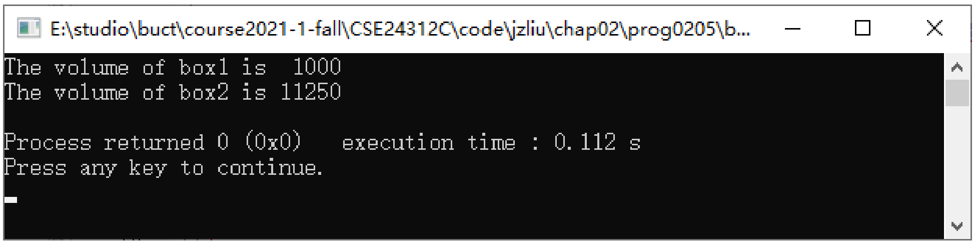
```
Constructing
 6

Process returned 0 (0x0)   execution time : 0.112 s
Press any key to continue.
```

图 2.4: 运行结果

2.5 ex05p32-构造函数的重载 (两个构造函数，一个有参数，一个无参数)

```
#include <iostream>

using namespace std;

class Box
{
public:
    Box( ); //声明一个无参的构造函数
    Box(int h,int w,int len):height(h),width(w),length(len) { }
//声明一个有参的构造函数，用参数的初始化表对数据成员初始化
    int volume( );
private:
    int height;
    int width;
    int length;
};
Box::Box( ) //定义一个无参的构造函数
{
    height=10;
    width=10;
    length=10;
}

int Box::volume( )
{
    return height*width*length;
}

int main( )
{
    Box box1; //建立对象box1,不指定实参
    cout<<"The volume of box1 is "<<box1.volume( )<<endl;
    Box box2(15,30,25); //建立对象box2,指定3个实参
    cout<<"The volume of box2 is "<<box2.volume( )<<endl;
    return 0;
}
```

运行结果:



E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap02\prog0205\b...

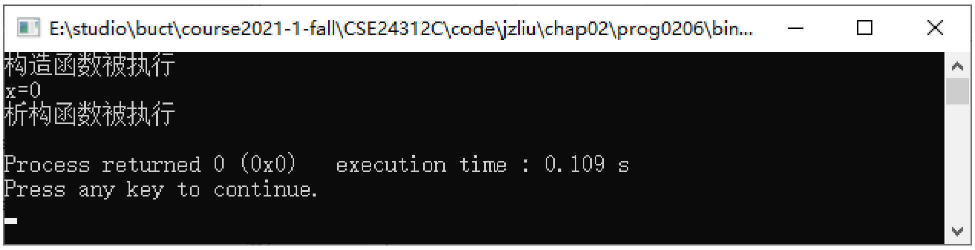
```
The volume of box1 is  1000
The volume of box2 is 11250

Process returned 0 (0x0)   execution time : 0.112 s
Press any key to continue.
```

图 2.5: 运行结果

2.6 ex06p35-包含构造函数和析构函数的 C++ 程序

```
#include <iostream>

using namespace std;

class Test
{
private:
    int x;
public:
    Test()
    {
        cout<<"构造函数被执行"<<endl;
        x=0;
    }
    ~Test()
    {
        cout<<"析构函数被执行"<<endl;
    }
    void print()
    {
        cout<<"x="<<x<<endl;
    }
};

int main()
{
    Test obj1;
    obj1.print();
    return 0;
}
```

运行结果:


E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap02\prog0206\bin...

```
构造函数被执行
x=0
析构函数被执行

Process returned 0 (0x0)   execution time : 0.109 s
Press any key to continue.
```

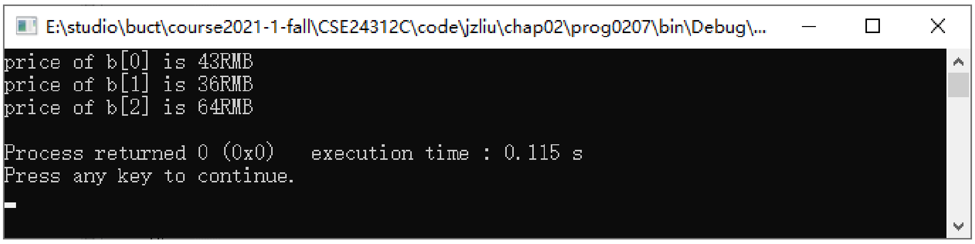
图 2.6: 运行结果

2.7 ex07p36-对象数组的使用

```
#include <iostream>

using namespace std;

class book
{
public:
    //声明有默认参数的构造函数，用参数初始化表对数据成员初始化
    book(int p=0,int w=0,int len=0): page(p),width(w),length(len) { }

    int price( );
private:
    int page;
    int width;
    int length;
};

int book::price( )
{
    return (page*width*length)/1000;
}
```

```
int main()
{
    book b[3]= { //定义对象数组
        book(100,24,18), //调用构造函数book，提供第1个元素的实参
        book(150,24,10), //调用构造函数book，提供第2个元素的实参
        book(300,18,12) //调用构造函数book，提供第3个元素的实参
    };
    cout<<"price of b[0] is "<< b[0].price( )<<"RMB"<<endl;
    cout<<"price of b[1] is "<< b[1].price( )<<"RMB"<<endl;
    cout<<"price of b[2] is "<< b[2].price( )<<"RMB"<<endl;
}
```

运行结果：



E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap02\prog0207\bin\Debug\...

```
price of b[0] is 43RMB
price of b[1] is 36RMB
price of b[2] is 64RMB

Process returned 0 (0x0)   execution time : 0.115 s
Press any key to continue.
```

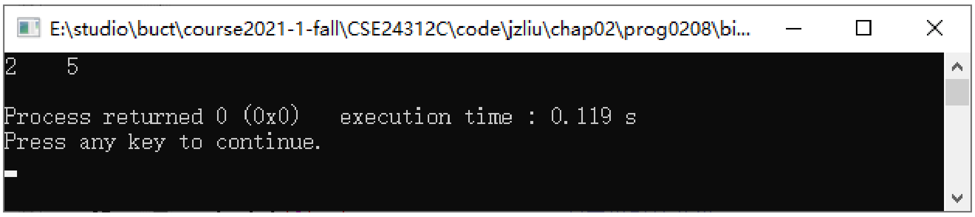
图 2.7: 运行结果

2.8 ex08p38-对象及成员的使用

```
#include <iostream>

using namespace std;

class A
{
    float x,y;
public:
    float m,n;
    void setXY(float a,float b)
    {
        x=a;
        y=b;
    }
    void print()
    {
        cout<<x<<" "<<y<<endl;
    }
};

int main()
```

```
{
    A a1,a2; //定义对象
    a1.m=10;
    a1.n=20;
    a1.setXY(2.0, 5.0);
    a1.print();
    return 0;
}
```

运行结果：



E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap02\prog0208\bi...

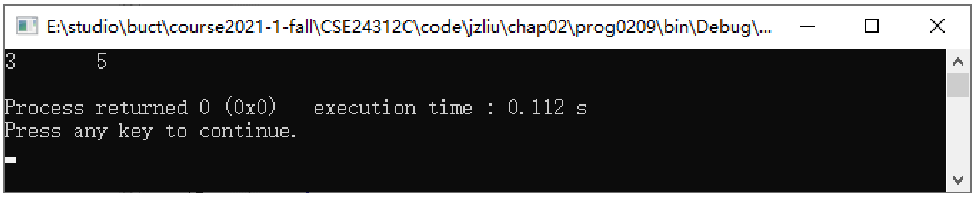
```
2    5

Process returned 0 (0x0)   execution time : 0.119 s
Press any key to continue.
```

图 2.8: 运行结果

2.9 ex09p38-使用引用访问私有数据成员

```
#include <iostream>

using namespace std;

class Test
{
private:
    int x,y;
public:
    void setXY(int a, int b)
    {
        x=a;
        y=b;
    }
    void getXY(int &px, int &py)
    {
        px=x; //提取x,y值
        py=y;
    }
    void Printxy()
    {
        cout<<"x="<<x<<'\t'<<"y="<<y<<endl;
    }
};
int main()
{
    Test p1,p2;
    p1.setXY(3,5);
    int a,b;
    p1.getXY(a, b); //将 a=x, b=y
    cout<<a<<'\t'<<b<<endl;
    return 0;
}
```

运行结果：



E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap02\prog0209\bin\Debug\...

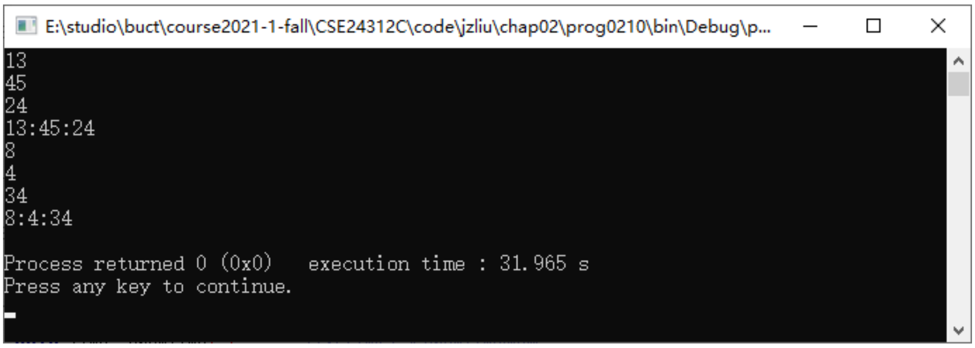
```
3       5

Process returned 0 (0x0)   execution time : 0.112 s
Press any key to continue.
```

图 2.9: 运行结果

2.10 ex10p38-用含成员函数的类来处理

```
#include <iostream>

using namespace std;

class Time
{
private: //数据成员为私有
    int hour;
    int minute;
    int sec;
public:
    void setTime(); //公用成员函数
    void showTime(); //公用成员函数
};

int main()
{
    Time t1; //定义对象t1
    t1.setTime( ); //调用对象t1的成员函数setTime，向t1的数据成员输入数据
    t1.showTime( ); //调用对象t1的成员函数showTime，输出t1的数据成员的值
    Time t2; //定义对象t2
    t2.setTime( ); //调用对象t2的成员函数setTime，向t2的数据成员输入数据
    t2.showTime( ); //调用对象t2的成员函数showTime，输出t2的数据成员的值
    return 0;
}

void Time::setTime( ) //在类外定义setTime函数
{
    cin>>hour;
    cin>>minute;
    cin>>sec;
}

void Time::showTime( ) //在类外定义showTime函数
{
    cout<<hour<<":"<<minute<<":"<<sec<<endl;
}
```

运行结果：



E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap02\prog0210\bin\Debug\p...

```
13
45
24
13:45:24
8
4
34
8:4:34

Process returned 0 (0x0)   execution time : 31.965 s
Press any key to continue.
```

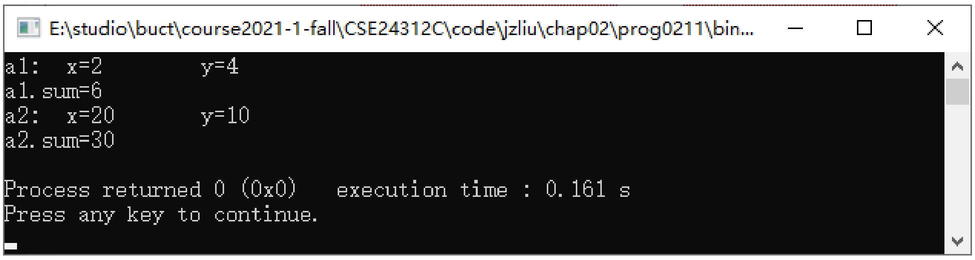
图 2.10: 运行结果

2.11 ex11p39-带缺省参数的成员函数

```
#include <iostream>

using namespace std;

class A
{
    float x,y;
public:
    float Sum(void)
    {
        return x+y;
    }
    void Set(float a,float b=10.0)
    {
        x=a;
        y=b;
    }
    void print(void)
    {
        cout<<"x="<<x<<'\t'<<"y="<<y<<endl;
    }
};
```

```
int main(void)
{
    A a1,a2;
    a1.Set (2.0,4.0);
    cout<<"a1:  ";
    a1.print ();
    cout<<"a1.sum="<<a1.Sum ()<<endl;
    a2.Set(20.0);
    cout<<"a2:  ";
    a2.print ();
    cout<<"a2.sum="<<a2.Sum ()<<endl;
    return 0;
}
```

运行结果：



E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap02\prog0211\bin...

```
a1:  x=2        y=4
a1.sum=6
a2:  x=20       y=10
a2.sum=30

Process returned 0 (0x0)   execution time : 0.161 s
Press any key to continue.
```

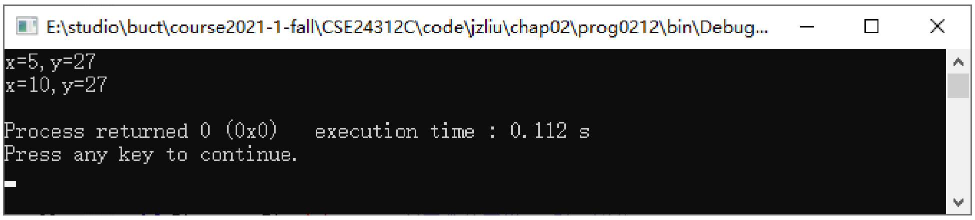
图 2.11: 运行结果

2.12 ex12p40-静态数据成员的使用

```
#include <iostream> //包含头文件iostream

using namespace std; //使用命名空间std

class Sample
{
private:
    int x;
    static int y; // y为静态成员，实现多个对象之间数据的共享
public:
    Sample(int a);
    void print();
};
Sample:: Sample(int a)
{
    x=a;
    y ++;
}
void Sample::print()
```

```
{
    cout<<"x="<<x<<",y="<<y<<endl;
}
int Sample::y=25;

int main()
{
    Sample s1(5);
    Sample s2(10);
    s1.print();
    s2.print();
    return 0;
}
```

运行结果：



E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap02\prog0212\bin\Debug...

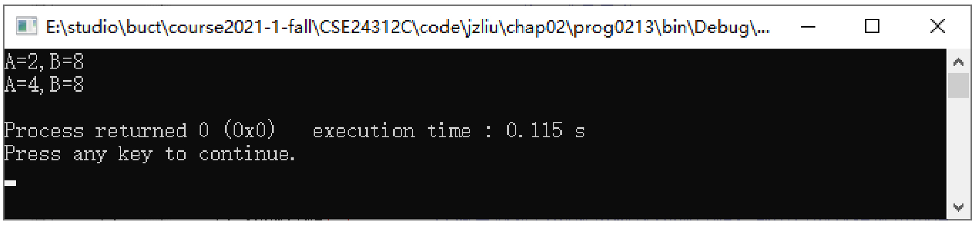
```
x=5,y=27
x=10,y=27

Process returned 0 (0x0)   execution time : 0.112 s
Press any key to continue.
```

图 2.12: 运行结果

2.13 ex13p42-静态成员函数的应用

```
#include <iostream>

using namespace std;

class Tc
{
private:
    int A;
    static int B;//静态数据成员
public:
    Tc(int a)
    {
        A=a;
        B+=a;
    }
    static void display(Tc c)//Tc的对象为形参
    {
        cout<<"A="<<c.A<<",B="<<B<<endl;
    }
};
int Tc::B=2;
int main()
{
    Tc a(2),b(4);
    Tc::display (a);
    Tc::display (b);
    return 0;

}
```

运行结果：



E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap02\prog0213\bin\Debug\...

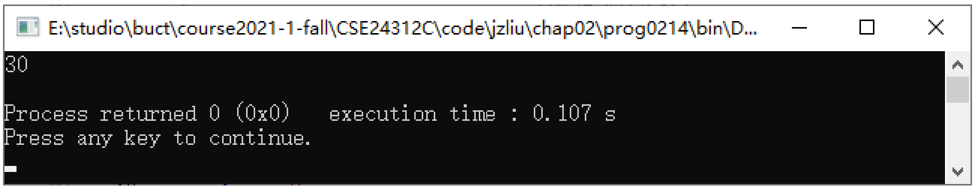
```
A=2,B=8
A=4,B=8

Process returned 0 (0x0)   execution time : 0.115 s
Press any key to continue.
```

图 2.13: 运行结果

2.14 ex14p43-友元函数的使用

```
#include <iostream>

using namespace std; //使用命名空间std

class Sample
{
private:
    int n;
public:
    Sample(int i)
    {
        n=i;
    }
    friend int add(Sample &s1,Sample &s2);
};

int add(Sample &s1,Sample &s2)
{
    return s1.n+s2.n;
}

int main()
{
    Sample s1(10),s2(20);
    cout<<add(s1,s2)<<endl;
    return 0;
}
```

运行结果：



E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap02\prog0214\bin\D...

```
30

Process returned 0 (0x0)   execution time : 0.107 s
Press any key to continue.
```

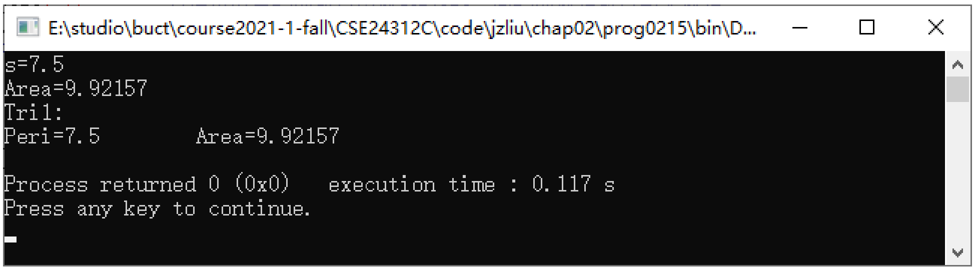
图 2.14: 运行结果

2.15 ex15p44-类的应用 (三角形类)

```
#include <iostream>
#include <math.h>

using namespace std;

class Triangle
{
private:
    float a,b,c; //三边为私有成员数据
public:
    void Setabc(float x, float y, float z);//置三边的值
    void Getabc(float &x, float &y, float &z);//取三边的值
    float Perimeter();//计算三角形的周长
    float Area();//计算三角形的面积
    void Print();//打印相关信息
};
void Triangle::Setabc (float x,float y,float z)
{
    a =x; //置三边的值
    b=y;
    c=z;
}
void Triangle::Getabc (float &x,float &y,float &z) //取三边的值
{
    x=a;
    y=b;
    z=c;
}
float Triangle::Perimeter ()
{
    return (a+b+c)/2; //计算三角形的周长
}
float Triangle::Area () //计算三角形的面积
```

```
{
   float area, p;
   p= Perimeter();
   area=sqrt((p-a)*(p-b)*(p-c)*p);
   return area;
}
void Triangle::Print() //打印相关信息
{
   cout<<"Peri="<<Perimeter()<<'\t'<<"Area="<<Area()<<endl;
}

int main()
{
   Triangle Tri1; //定义三角形类的一个实例（对象）
   Tri1.Setabc (4,5,6); //为三边置初值
   float x,y,z;
   Tri1.Getabc (x,y,z); //将三边的值为x,y,z赋值
   cout<<x<<'\t'<<y<<'\t'<<z<<endl;
   cout<<"s="<<Tri1.Perimeter ()<<endl;//求三角形的周长
   cout<<"Area="<<Tri1.Area ()<<endl;//求三角形的面积
   cout<<"Tri1:"<<endl;
   Tri1.Print();//打印有关信息
   return 0;
}
```

运行结果:


E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap02\prog0215\bin\D...

```
s=7.5
Area=9.92157
Tri1:
Peri=7.5        Area=9.92157

Process returned 0 (0x0)   execution time : 0.117 s
Press any key to continue.
```

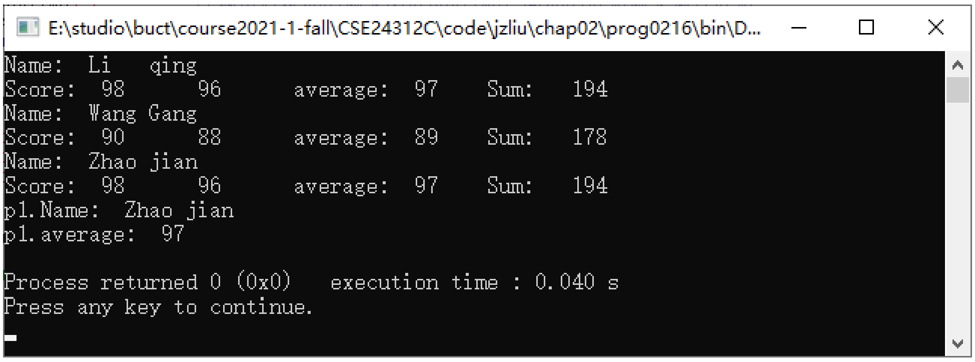
图 2.15: 运行结果

2.16 ex16p45-类的应用 (学生类)

```
#include <iostream>
#include <cstring>

using namespace std;

class Stu
{
   char Name[20]; //学生姓名
   float Chinese;  //语文成绩
```

```cpp
    float Math;  //数学成绩
public:
    float Average(void); //计算平均成绩
    float Sum(void); //计算总分
    void Show(void); //打印信息
    void SetStudent(char Name[20],float,float);//为对象置姓名、成绩
    void SetName(char Name[20]); //为对象置姓名
    string GetName(void); //取得学生姓名
};
float Stu::Average(void)
{
    return (Chinese+Math)/2; //平均成绩
}
float Stu::Sum(void)
{
    return Chinese+Math; //总分
}
void Stu::Show(void) //打印信息
{
    cout<<"Name: "<<Name<<endl<<"Score: "<<Chinese<<'\t'<<
        Math<<'\t'<<"average: "<<Average()<<'\t'<<"Sum: "<<Sum()<<endl;
}
void Stu::SetStudent(char *name,float chinese,float math)
{
    strcpy(Name,name); //置姓名
    Chinese=chinese; //置语文成绩
    Math=math;  //置数学成绩
}
void Stu::SetName(char N[20])
{
    strcpy(Name,N);
};
string Stu::GetName(void)
{
    return Name; //返回姓名
}

int main(void)
{
    Stu p1,p2;
    p1.SetStudent("Li qing",98,96);//对象置初值
    p2.SetStudent("Wang Gang",90,88); //对象置初值
    p1.Show();//打印信息
    p2.Show();//打印信息
    p1.SetName ("Zhao jian");//重新置p1对象的名字
    p1.Show ();
    cout<<"p1.Name: "<<p1.GetName ()<<endl;//打印对象的名字
    cout<<"p1.average: "<<p1.Average ()<<endl;//打印对象的成绩
    return 0;
}
```

运行结果：


E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap02\prog0216\bin\D...

```
Name:  Li   qing
Score:  98      96      average:  97    Sum:   194
Name:  Wang Gang
Score:  90      88      average:  89    Sum:   178
Name:  Zhao jian
Score:  98      96      average:  97    Sum:   194
p1.Name:  Zhao jian
p1.average:  97

Process returned 0 (0x0)   execution time : 0.040 s
Press any key to continue.
```

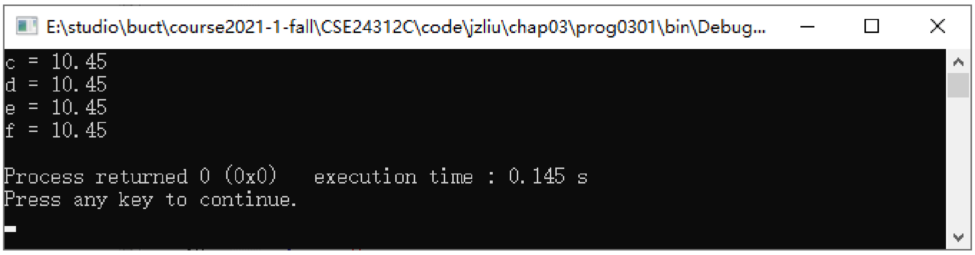
图 2.16: 运行结果

第 3 章　重载与类型转换

3.1　ex01p47-函数重载

```
#include <iostream>

using namespace std;

float i_add(float x, float y)
{
    return x+y;
}
float p_add(float *p, float *q)
{
    return *p+*q;
}

float add(float x, float y)
{
    return x+y ;
}
float add(float *p, float *q)
{
    return *p+*q;
}

int main()
{
    float a = 4.67;
    float b = 5.78;
    float c = i_add(a, b);
    float d = p_add(&a, &b);
    cout<<"c = " << c <<endl;
    cout<<"d = " << d <<endl;

    // 函数重载
    float e = add(a, b);
    float f = add(&a, &b);
    cout<<"e = " << e <<endl;
    cout<<"f = " << f <<endl;

    return 0;
}
```

运行结果：



E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap03\prog0301\bin\Debug...

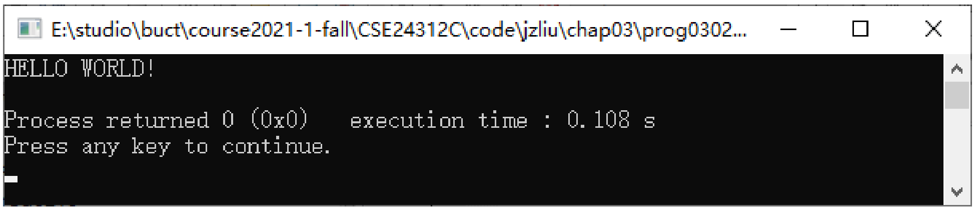
```
c = 10.45
d = 10.45
e = 10.45
f = 10.45

Process returned 0 (0x0)   execution time : 0.145 s
Press any key to continue.
```

图 3.1: 运行结果

3.2 ex02p50-演示缺省拷贝构造函数存在的问题 (解释深拷贝)，添加了 strupr 函数【有问题：程序可运行】

```
#include <iostream>
#include <cstring>

using namespace std;

class Student
{
public:
    char * name;
    Student()
    {
        name=new char[100];
    }
    ~Student( )
    {
        delete [ ] name;
    }
};

/**
 * strupr不是标准C库函数，应该是VC自己扩充的。
 */
char *strupr(char *str)
{
   char *ptr = str;

   while (*ptr != '\0')
   {
      if (islower(*ptr))
         *ptr = toupper(*ptr);
      ptr++;
   }

   return str;
}

Student upperCase(Student s)
{
   s.name=strupr(s.name);  //将串中的小写字母转换为大写字母
   return s;
}

int main()
{
   Student s1;
   strcpy(s1.name, "Hello world!");
   cout<<upperCase(s1).name<<endl;
   return 0;
}
```

运行结果：



E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap03\prog0302...

```
HELLO WORLD!

Process returned 0 (0x0)   execution time : 0.108 s
Press any key to continue.
```

图 3.2: 运行结果

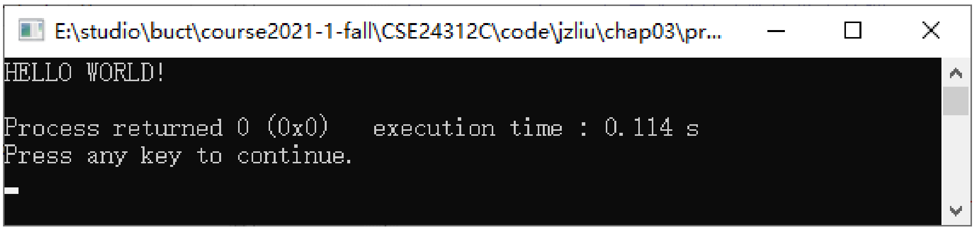
3.3 ex03p51-使用重载拷贝函数来解决浅拷贝

```
#include <iostream>
#include <cstring>
```

```
using namespace std;

class Student
{
  public:
  char * name;
  Student( )
  {
    name=new char[100];
  }
  Student(Student & s)  //拷贝构造函数
  {
    name=new char[100];
    strcpy(name, s.name); //深拷贝
  }
  ~Student( )
  {
    delete [] name;
  }
};


/**
 * strupr不是标准C库函数，应该是VC自己扩充的。
 */
char *strupr(char *str)
{
    char *ptr = str;

    while (*ptr != '\0')
    {
        if (islower(*ptr))
            *ptr = toupper(*ptr);
        ptr++;
    }

    return str;
}


Student upperCase(Student s)
{
  s.name=strupr(s.name);  //将串中的小写字母转换为大写字母
  return s;
}

int main()
```

```
{
  Student s1;
  strcpy(s1.name, "Hello world!");
  cout<<upperCase(s1).name<<endl;
  return 0;
}
```

运行结果：



```
HELLO WORLD!

Process returned 0 (0x0)   execution time : 0.114 s
Press any key to continue.
```

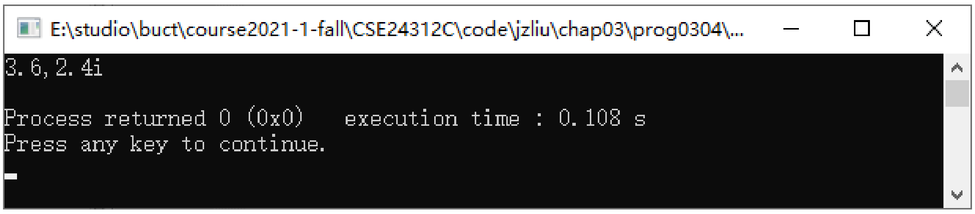
E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap03\pr...

图 3.3: 运行结果

3.4 ex04p53-实现复数的加法运算

```
#include <iostream>

using namespace std;

class Complex
{
  float real;
  float image;

public:
  Complex(float r, float i) //带两个参数的构造函数
  {
    real=r;
    image=i;
  }
  Complex add(const Complex &c)  //复数的加法运算
  {
    return(Complex(real+c.real, image+c.image));
  }
  void show( )  //显示复数的实部和虚部
  {
    cout<<real<<','<<image<<'i'<<endl;
  }
};

int main( )
{
  Complex c1(1.2, 3.5);
  Complex c2(2.4, −1.1);
  Complex c3=c1.add(c2);
  c3.show( );
  return 0;
}
```

运行结果：



E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap03\prog0304\...

```
3.6,2.4i

Process returned 0 (0x0)   execution time : 0.108 s
Press any key to continue.
```

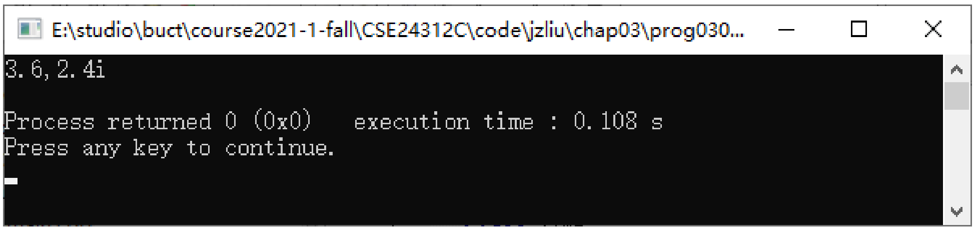
图 3.4: 运行结果

3.5 ex05p56-使用成员函数重载方法实现复数加法运算

```
#include <iostream>

using namespace std;

class Complex
{
private:
   float real;
   float image;
public:
   Complex(float r, float i) // 带两个参数的构造函数
   {
      real=r;
      image=i;
   }
   Complex operator + (const Complex &c)  // 复数的加法运算
   {
      return(Complex(real+c.real, image+c.image));
   }
   void show( )  //显示复数的实部和虚部
   {
      cout<<real<<','<<image<<'i'<<endl;
   }
};

int main( )
{
   Complex c1(1.2, 3.5);
   Complex c2(2.4, −1.1);
```

```
    Complex c3=c1+c2;
    c3.show( );
    return 0;
}
```

运行结果：



E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap03\prog030...

```
3.6,2.4i

Process returned 0 (0x0)   execution time : 0.108 s
Press any key to continue.
```

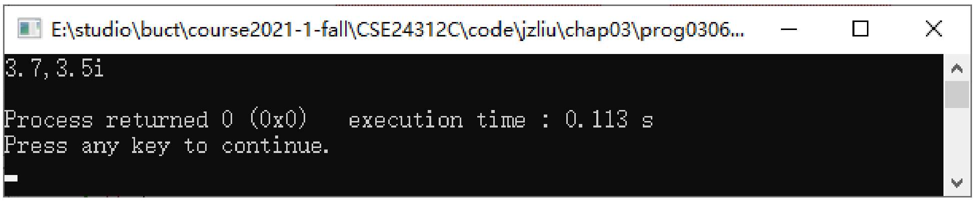
图 3.5: 运行结果

3.6 ex06p56-使用成员函数重载方法实现复数加法运算（重载了构造函数）

```
#include <iostream>

using namespace std;

class Complex
{
  float real;
  float image;
public:
  Complex(float r) //带一个参数的构造函数
  {
    real=r;
    image=0;
  }
Complex(float r, float i) //带两个参数的构造函数
  {
    real=r;
    image=i;
  }
  Complex operator + (const Complex &c)  //复数的加法运算
  {
    return(Complex(real+c.real, image+c.image));
  }
  void show( )  //显示复数的实部和虚部
  {
    cout<<real<<','<<image<<'i'<<endl;
  }
};
```

```
int main( )
{
  Complex c1(1.2, 3.5);
  Complex c2(2.4, -1.1);
  Complex c3=c1+2.5; // 这里将2.5转换为实部为2.5虚部为0的复数
  c3.show( );
  return 0;
}
```

运行结果:



E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap03\prog0306...

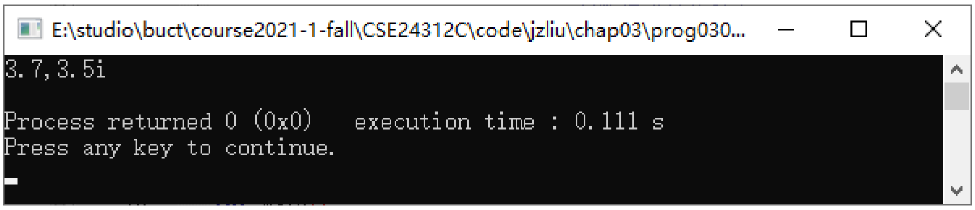
```
3.7,3.5i

Process returned 0 (0x0)   execution time : 0.113 s
Press any key to continue.
```

图 3.6: 运行结果

3.7 ex07p58-用友元函数重载来实现复数的加法运算

```
#include <iostream>

using namespace std;

class Complex
{
    float real;
    float image;
public:
    Complex(float r) //带一个参数的构造函数
    {
        real=r;
        image=0;
    }
    Complex(float r, float i) //带两个参数的构造函数
    {
        real=r;
        image=i;
    }
    //复数的加法运算
    friend Complex operator + (const Complex &c1, const Complex &c2)
    {
        return(Complex(c1.real+c2.real, c1.image+c2.image));
    }
    void show( )  //显示复数的实部和虚部
    {
        cout<<real<<','<<image<<'i'<<endl;
    }
};

int main( )
{
    Complex c1(1.2, 3.5);
    Complex c2(2.4, −1.1);
    Complex c3=2.5+c1;
    c3.show( );
    return 0;
}
```

运行结果：



E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap03\prog030...

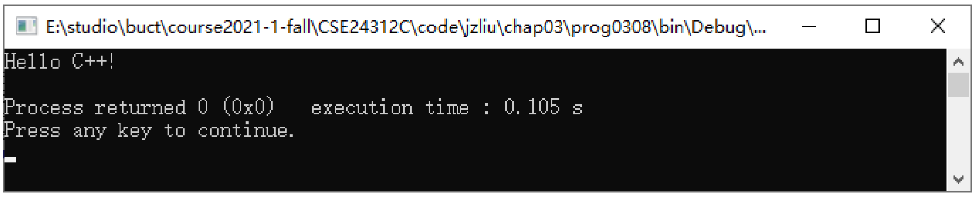
```
3.7,3.5i

Process returned 0 (0x0)   execution time : 0.111 s
Press any key to continue.
```

图 3.7: 运行结果

3.8 ex08p61-设计一个字符串类 (存在指针悬挂和堆的二次释放问题)【有问题：程序可运行】

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


class String
{
    char *p;
    int size;
public:
    String(char *q) //带参数的构造函数
    {
        size=strlen(q)+1;
        p=new char[size];
        strcpy(p,q);
    }
    void show( )
    {
        cout<<p<<endl;
    }
    ~String( )
    {
        delete[ ] p;
    }
};

int main( )
{
    String s1("Hello world!");
    String s2("Hello C++!");
    s1=s2;
    s1.show( );
    return 0;
}
```

运行结果:



E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap03\prog0308\bin\Debug\...

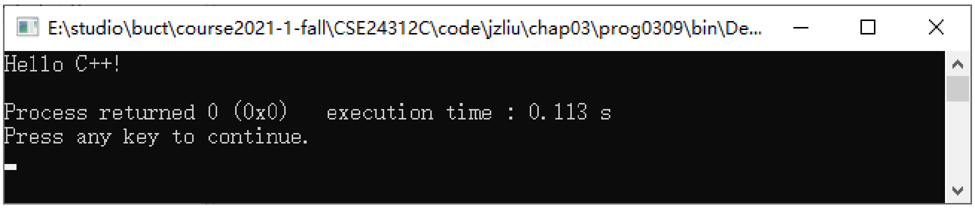
```
Hello C++!

Process returned 0 (0x0)   execution time : 0.105 s
Press any key to continue.
```

图 3.8: 运行结果

3.9 ex09p63-设计一个字符串类，通过重载运算符的方法解决指针悬挂问题

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

class String
{
    char *p;
    int size;
public:
    String(char *q) //带参数的构造函数
    {
        size=strlen(q)+1;
        p=new char[size];
        strcpy(p,q);
    }
    void operator =(String &s) //重载赋值运算符函数
    {
        delete[ ] p;
        size=s.size;
        p=new char[size];
        strcpy(p,s.p);   //深拷贝
    }
    void show( )
    {
        cout<<p<<endl;
    }
    ~String( )
    {
        delete[ ] p;
    }
};

int main( )
{
    String s1("Hello world!");
    String s2("Hello C++!");
    s1=s2;
    s2.show( );
    return 0;
}
```

运行结果：


E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap03\prog0309\bin\De...

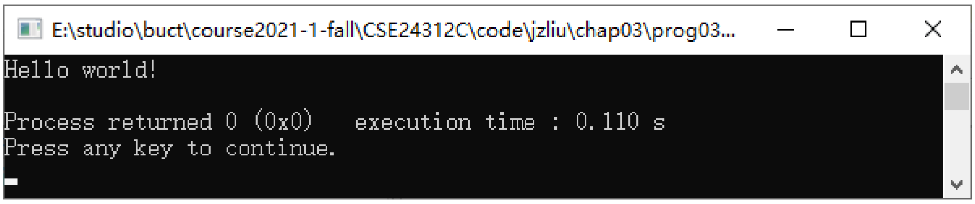
```
Hello C++!

Process returned 0 (0x0)   execution time : 0.113 s
Press any key to continue.
```

图 3.9: 运行结果

3.10 ex10p64-设计一个字符串类，实现其赋值运算符的右结合性 (定义返回为引用类型)

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

class String
{
  char *p;
  int size;
public:
  String(char *q) // 带参数的构造函数
  {
    size=strlen(q)+1;
    p=new char[size];
    strcpy(p,q);
  }
  String& operator=(String &s)  // 重载赋值运算符函数
  {
    delete[ ] p;
    size=s.size;
    p=new char[size];
    strcpy(p, s.p);   // 深拷贝
    return *this;
  }
void show( )
  {
    cout<<p<<endl;
  }
  ~String( )
  {
    delete[ ] p;
  }
};

int main( )
{
  String s1("Hello world!");
  String s2("Hello China!");
  String s3("Hello C++!");
  s3=s2=s1;
  s3.show( );
  return 0;
}
```

运行结果:


E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap03\prog03...

```
Hello world!

Process returned 0 (0x0)   execution time : 0.110 s
Press any key to continue.
```

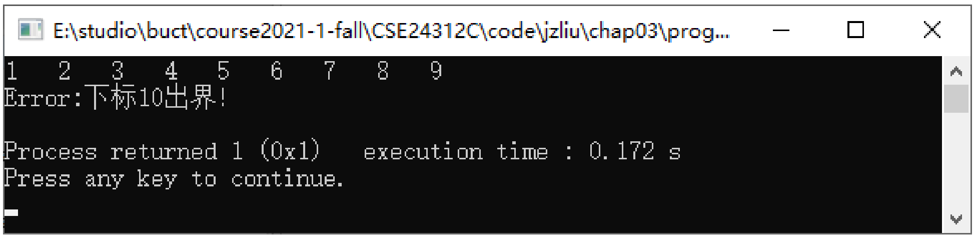
图 3.10: 运行结果

3.11 ex11p67-定义分配堆的类，重载下标运算符

```
#include <iostream>
#include "memory.h"
#include "stdlib.h"
```

```
using namespace std;

class CArray
{
    int len;
    float *arp;
public:
    CArray(int n=0);
    ~CArray( )
    {
        if (arp) delete[ ]arp;
    }
    int GetLen( )
    {
        return len;
    }
    void SetLen(int l)
    {
        if(l>0)
        {
            if(arp) delete[ ] arp;
            arp=new float[l];
            memset(arp, 0, sizeof(float)*l);  //堆中字节全部初始化为0
            len=l;
        }
    }
    float & operator[ ] (int index);  //定义重载的下标运算符，注意返回类型是引用
};

CArray::CArray(int n)
{
    if(n>0)
    {
        arp=new float[n];
        memset(arp, 0, sizeof(float)*n);  //arp中的元素全部初始化为0
        len=n;
    }
    else
    {
        len=0;
        arp=0;
    }
}

float& CArray::operator[ ](int index) //重载下标运算符的实现
{
    if(index>=len || index<0) //如果参数index超出规定的范围，则输出越界信息
    {
        cout<<"\nError:下标"<<index<<"出界!"<<'\n';
        exit(1); //程序非正常退出
    }
    return arp[index]; //如果不越界，则返回相应的数据
}

int main( )
{
    CArray m1(10),m2(3);
    int i;
    for(i=0; i<10; i++) m1[i]=i;
    for(i=1; i<11; i++)
        cout<<m1[i]<<" ";
    cout<<endl;
    m2[2]=26;
    cout<<"m2[2]="<<m2[2]<<'\n';
    return 0;
}
```

运行结果：


E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap03\prog...

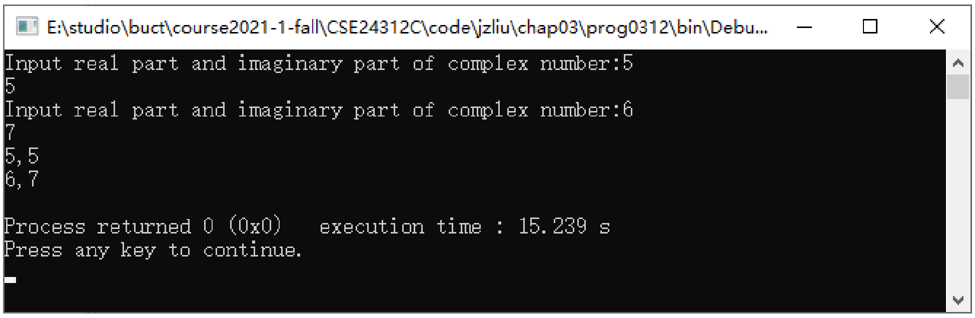
```
1   2   3   4   5   6   7   8   9
Error:下标10出界!

Process returned 1 (0x1)   execution time : 0.172 s
Press any key to continue.
```

图 3.11: 运行结果

3.12 ex12p69-设计复数类，重载 « 和 » 运算符

```
#include <iostream>

using namespace std;

class Complex
{
  float real;
  float image;
  friend ostream & operator <<(ostream& output, Complex& c)
  {
    output<<c.real<<','<<c.image<<endl;
    return output;
  }
  friend istream & operator >>(istream& input, Complex& c)
  {
    cout<<"Input real part and imaginary part of complex number:";
    input>>c.real>>c.image;
    return input;
  }
};

int main( )
{
  Complex c1,c2;
  cin>>c1>>c2;
  cout<<c1<<c2;
  return 0;
}
```

运行结果：


E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap03\prog0312\bin\Debu...

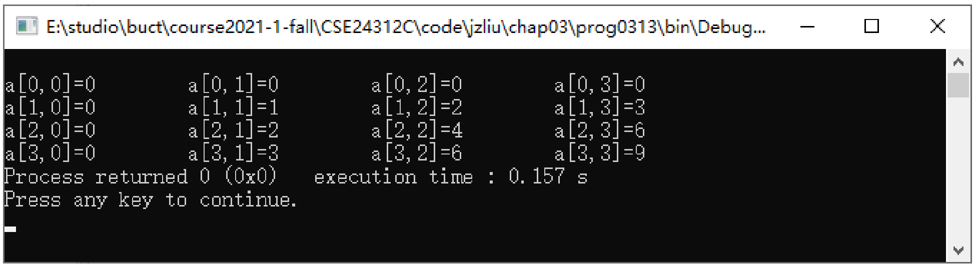
```
Input real part and imaginary part of complex number:5
5
Input real part and imaginary part of complex number:6
7
5,5
6,7

Process returned 0 (0x0)   execution time : 15.239 s
Press any key to continue.
```

图 3.12: 运行结果

3.13 ex13p70-重载函数调用运算符，实现二维数组的下标合法性检查

```
#include <iostream>
#include "memory.h"
#include "stdlib.h"

using namespace std;


const int cor1=4;
const int cor2=4;

class CArray
{
    float arr[cor1][cor2];
public:
    CArray( )
    {
        memset(arr, 0, sizeof(float)*cor1*cor2);  //arr中的元素全部初始化为0
    }
    void operator( )(int i, int j, float f);  //声明函数调用运算符重载函数
    float GetElem(int i, int j);
};

void CArray::operator( )(int i, int j, float f) //函数调用运算符重载函数的实现
{
    if(i>=0 && i<cor1 && j>=0 && j<cor2)
        arr[i][j]=f;
    else
    {
        cout<<"下标越界!"<<endl;
        exit(1); //程序非正常退出
    }
}

float CArray::GetElem(int i,int j)
{
    if(i<0 || j<0 || i>=cor1 || j>=cor2)
    {
        cout<<"下标越界!"<<endl;
        exit(1); //程序非正常退出
    }
    return arr[i][j];
}

int main( )
{
    CArray a;
    int i, j;
    for(i=0; i<4; i++)
        for(j=0; j<4; j++)
            a(i, j, i*j);  //使用重载的函数调用运算符

    for(i=0; i<4; i++)
    {
        cout<<endl;
        for(j=0; j<4; j++)
            cout<<"a["<<i<<","<<j<<"]="<<a.GetElem(i,j)<<'\t';
    }
    return 0;
}
```

运行结果:


E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap03\prog0313\bin\Debug...

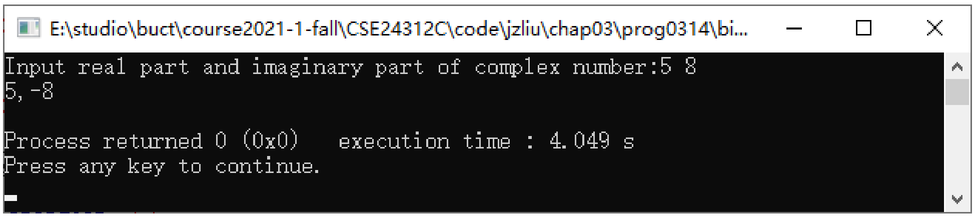
```
a[0,0]=0        a[0,1]=0        a[0,2]=0        a[0,3]=0
a[1,0]=0        a[1,1]=1        a[1,2]=2        a[1,3]=3
a[2,0]=0        a[2,1]=2        a[2,2]=4        a[2,3]=6
a[3,0]=0        a[3,1]=3        a[3,2]=6        a[3,3]=9
Process returned 0 (0x0)   execution time : 0.157 s
Press any key to continue.
```

图 3.13: 运行结果

3.14 ex14p71-用成员函数的方法重载 运算符，使之具有求共轭复数的功能

```
#include <iostream>

using namespace std;


class Complex
{
   float real;
   float image;
public:
   Complex( )
   {
      ; //不带参数的构造函数
   }
   Complex(float r, float i)  //带两个参数的构造函数
   {
      real=r;
      image=i;
   }
   Complex operator ~( )
   {
      return(Complex(real, −image));
   }
   friend ostream & operator <<(ostream& output, Complex& c)
   {
      output<<c.real<<','<<c.image<<endl;
      return output;
   }
   friend istream & operator >>(istream& input, Complex& c)
   {
      cout<<"Input real part and imaginary part of complex number:";
      input>>c.real>>c.image;
      return input;
   }
};

int main( )
{
    Complex c1, c2;
    cin>>c1;
    c2=~c1;
    cout<<c2;
    return 0;
}
```

运行结果：



```
Input real part and imaginary part of complex number:5 8
5,-8

Process returned 0 (0x0)   execution time : 4.049 s
Press any key to continue.
```

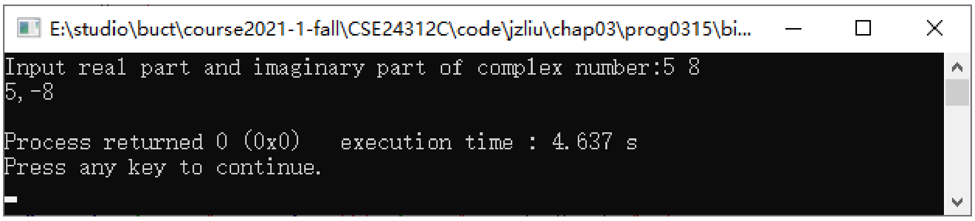
E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap03\prog0314\bi...

图 3.14: 运行结果

3.15 ex15p72-用友元函数的方法重载 运算符，使之具有求共轭复数的功能

```
#include <iostream>

using namespace std;

class Complex
{
    float real;
    float image;
public:
    Complex( )
    {
        ; //不带参数的构造函数
    }
    Complex(float r, float i)  //带两个参数的构造函数
    {
        real=r;
        image=i;
    }
    friend Complex operator ~(Complex &c)
    {
        return(Complex(c.real, −c.image));
    }
    friend ostream & operator <<(ostream& output, Complex& c)
    {
        output<<c.real<<','<<c.image<<endl;
        return output;
    }
    friend istream & operator >>(istream& input, Complex& c)
    {
        cout<<"Input real part and imaginary part of complex number:";
        input>>c.real>>c.image;
        return input;
    }
};

int main( )
{
    Complex c1, c2;
    cin>>c1;
    c2=~c1;
    cout<<c2;
    return 0;
}
```

运行结果：


E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap03\prog0315\bi...

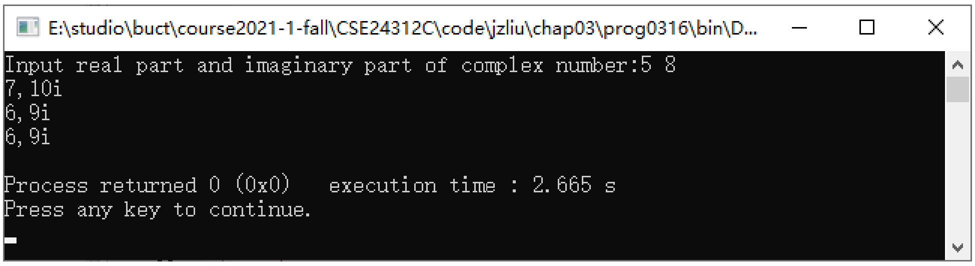
```
Input real part and imaginary part of complex number:5 8
5,-8

Process returned 0 (0x0)   execution time : 4.637 s
Press any key to continue.
```

图 3.15: 运行结果

3.16 ex16p74-用成员函数的方法重载自增运算符 ++，分别实现 ++ 的前缀和后缀功能

```
#include <iostream>

using namespace std;
class Complex
{
  float real;
  float image;
  public:
    Complex( ){;}     //不带参数的构造函数
  Complex(float r, float i)  //带两个参数的构造函数
  {
    real=r;
    image=i;
  }
  Complex operator++( )  //重载为前缀运算符
  {
    real++;
    image++;
    return(*this);
  }
  Complex operator++(int x) //重载为后缀运算符
  {
    Complex c(real,image);
    real++;
    image++;
    return(c);
  }
  friend ostream & operator <<(ostream& output, Complex& c)
  {
    output<<c.real<<','<<c.image<<'i';
    return output;
  }
  friend istream & operator >>(istream& input, Complex& c)
  {
    cout<<"Input real part and imaginary part of complex number:";
    input>>c.real>>c.image;
    return input;
  }
};

int main( )
{
  Complex c1, c2, c3;
  cin>>c1;
  c2=++c1;
  c3=c1++;
  cout<<c1<<endl;
  cout<<c2<<endl;
  cout<<c3<<endl;
  return 0;
}
```

运行结果：



E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap03\prog0316\bin\D...

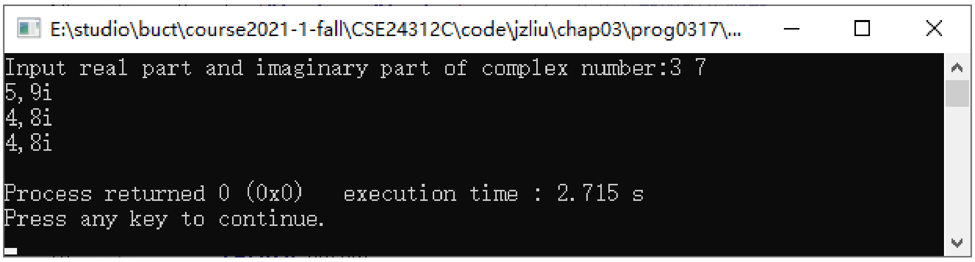
```
Input real part and imaginary part of complex number:5 8
7,10i
6,9i
6,9i

Process returned 0 (0x0)   execution time : 2.665 s
Press any key to continue.
```

图 3.16: 运行结果

3.17 ex17p75-用友元函数的方法重载自增运算符 ++，分别实现 ++ 的前缀和后缀功能

```
#include <iostream>

using namespace std;


class Complex
{
    float real;
    float image;
public:
    Complex( )
    {
        ; //不带参数的构造函数
    }
    Complex(float r, float i)  //带两个参数的构造函数
    {
        real=r;
        image=i;
    }
    friend Complex operator++(Complex& c)  //重载为前缀运算符
    {
        c.real++;
        c.image++;
        return(c);
    }
    friend Complex operator++(Complex& c, int x) //重载为后缀运算符
    {
        Complex t(c.real, c.image);
        c.real++;
        c.image++;
        return(t);
    }
    friend ostream & operator <<(ostream& output, Complex& c)
    {
        output<<c.real<<','<<c.image<<'i';
        return output;
    }
    friend istream & operator >>(istream& input, Complex& c)
    {
        cout<<"Input real part and imaginary part of complex number:";
        input>>c.real>>c.image;
        return input;
    }
};

int main( )
{
    Complex c1, c2, c3;
    cin>>c1;
    c2=++c1;
    c3=c1++;
    cout<<c1<<endl;
    cout<<c2<<endl;
    cout<<c3<<endl;
    return 0;
}
```

运行结果：



E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap03\prog0317\...

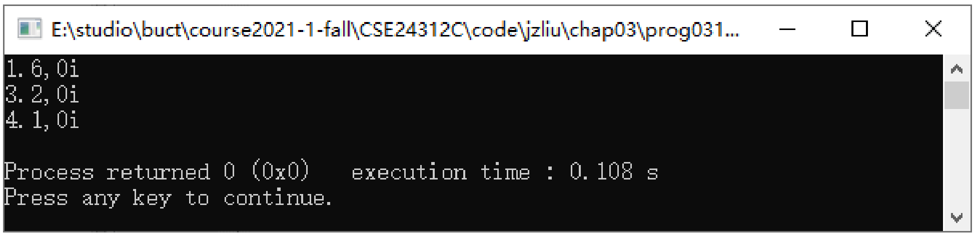
```
Input real part and imaginary part of complex number:3 7
5,9i
4,8i
4,8i

Process returned 0 (0x0)   execution time : 2.715 s
Press any key to continue.
```

图 3.17: 运行结果

3.18 ex18p77-针对复数类设计并使用转换构造函数

```
#include <iostream>

using namespace std;

class Complex
{
    float real;
    float image;
public:
    Complex(float r, float i=0) //第二个参数有缺省值，故可作为转换构造函数
    {
        real=r;
        image=i;
    }
//复数的加法运算
    friend Complex operator + (const Complex &c1, const Complex &c2)
    {
        return(Complex(c1.real+c2.real, c1.image+c2.image));
    }
    void show( )  //显示复数的实部和虚部
    {
        cout<<real<<','<<image<<'i'<<endl;
    }
};

int main( )
{
    Complex c1(1.6);
    Complex c2=3.2;  //完成了从实型到复数类型的转换
    Complex c3=2.5+c1; //完成了从实型到复数类型的转换
    c1.show( );
    c2.show( );
    c3.show( );
    return 0;
}
```

运行结果:



E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap03\prog031...

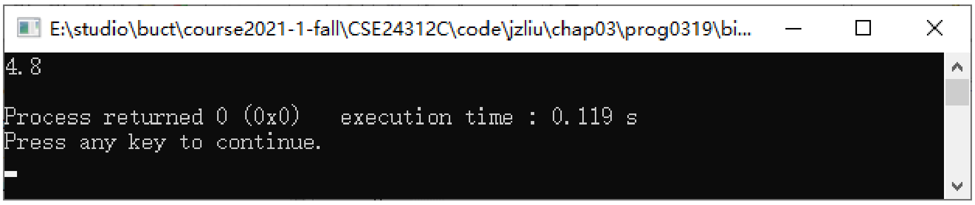
```
1.6,0i
3.2,0i
4.1,0i

Process returned 0 (0x0)   execution time : 0.108 s
Press any key to continue.
```

图 3.18: 运行结果

3.19 ex19p79-采用类型转换函数的方法实现复数类型向实数类型转换

```
#include <iostream>

using namespace std;

class Complex
{
  float real;
  float image;
public:
  Complex(float r, float i=0) //第二个参数有缺省值，故可作为转换构造函数
  {
    real=r;
    image=i;
  }
  operator float( ) //类型转换函数
  {
    return real;
  }
  //复数的加法运算
  friend Complex operator + (const Complex &c1, const Complex &c2)
  {
    return(Complex(c1.real+c2.real, c1.image+c2.image));
  }
  void show( )  //显示复数的实部和虚部
  {
    cout<<real<<','<<image<<'i'<<endl;
  }
};

int main( )
{
  Complex c1(1.6, 4.5);
  float f=3.2+float(c1);  //调用类型转换函数
  cout<<f<<endl;
  return 0;
}
```

运行结果：



E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap03\prog0319\bi...

```
4.8

Process returned 0 (0x0)   execution time : 0.119 s
Press any key to continue.
```

图 3.19: 运行结果

3.20 ex20p80-重载运算符应用 (矩阵例子)【有问题：程序中取矩阵元素地方有错误】

```
#include <iostream>
#include "stdlib.h"

using namespace std;
```

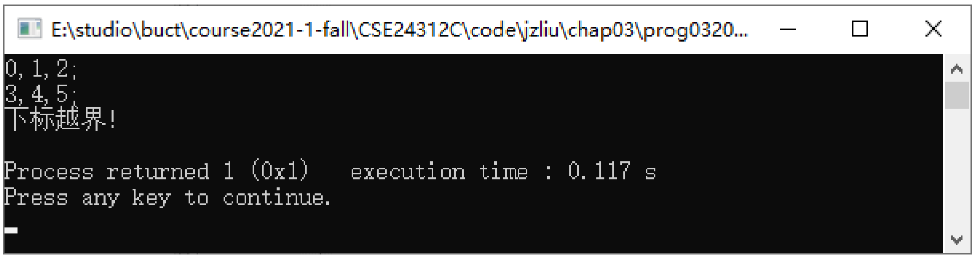
```
class Matrix
{
private:
    int rows;  // 行数
    int columns; // 列数
    double ∗pm; // 用来保存矩阵元素的一维结构
public:
    Matrix(int rows, int columns, double x=0);
    Matrix(Matrix & mat);  //拷贝构造函数，思考为什么要设置拷贝构造函数。

    ~Matrix( )
    {
        delete [] pm;
    }

    int getRows( )
    {
        return rows;
    }

    int getColumns( )
    {
        return columns;
    }

    //运算符重载
    Matrix& operator =(Matrix &mat); //重载赋值运算符=
    Matrix& operator ~( );    //重载运算符~，让其具有转置的功能
    double& operator( )(int i, int j); //重载函数运算符( )，让其具有按行列取元素功能
    //思考此处为什么不选择重载运算符[ ]?
    //重载<<
    friend ostream & operator <<(ostream& output, Matrix& mat);
    //重载+、−、∗
    friend Matrix& operator +(Matrix mat1, Matrix mat2);
    friend Matrix& operator −(Matrix mat1, Matrix mat2);
    friend Matrix& operator ∗(Matrix mat1, Matrix mat2);
    friend Matrix& operator ∗(Matrix mat, double k);
    friend Matrix& operator ∗(double k, Matrix mat);
};

Matrix::Matrix(int rows, int columns, double x)  //普通构造函数
{
    int i,j;
    if(rows>0 && columns>0)
    {
        this−>rows=rows;
        this−>columns=columns;
        pm=new double[rows∗columns]; //分配矩阵空间
        for(i=0; i<rows; i++)    //初始化矩阵中元素
            for(j=0; j<columns; j++)
                (*this)(i, j)=x;   //等价于*(pm+i*columns+j)=x;
    }
    else
    {
        rows=0;
        columns=0;
        pm=0;
    }
}

Matrix::Matrix(Matrix & mat)  //拷贝构造函数
{
    int i,j;
    rows=mat.getRows( );
    columns=mat.getColumns( );
    pm=new double[rows*columns];
    for(i=0; i<rows; i++)
        for(j=0; j<columns; j++)
            (*this)(i, j)=mat(i, j);  //等价于*(pm+i*columns+j)=mat(i,j);
}

Matrix& Matrix::operator=(Matrix &mat)  //重载赋值运算符函数
{
    int i,j;
    if(rows==mat.getRows( ) && columns==mat.getColumns( ))
    {
        for(i=0; i<rows; i++)    //初始化矩阵中元素
            for(j=0; j<columns; j++)
                (*this)(i, j)=mat(i, j);  //等价于*(pm+i*columns+j)=x;
        return *this;
    }
    else
    {
        cout<<"Error:维数不匹配！";
        exit(1);
    }
}

Matrix& Matrix::operator ~ ( ) //重载运算符~
{
    int i,j;
    Matrix matrix(columns, rows);
    for(i=0; i<rows; i++)
        for(j=0; j<columns; j++)
            matrix(j, i)=(*this)(i, j);
    return matrix;
```

```
}

double & Matrix::operator( )(int i, int j)  //重载函数运算符( )
{
    if(i>=0 && i<rows && j>=0 && j<columns)
        return *(pm+i*columns+j);  //注意，此处是否能等价为(*this)(i, j)
    else
    {
        cout<<"下标越界！"<<endl;
        exit(1); //程序非正常退出
    }
}

ostream & operator <<(ostream& output, Matrix& mat) //重载输出运算符<<
{
    int i,j;
    int rows=mat.getRows( );
    int columns=mat.getColumns( );
    for(i=0; i<rows; i++)
    {
        for(j=0; j<columns; j++)
            output<<mat(i,j)<<',';
        output<<'\b'<<';'<<endl;
    }
    return output;
}

Matrix& operator +(Matrix mat1, Matrix mat2)  //重载加法运算符+
{
    if(mat1.getRows( )==mat2.getRows( ) && mat1.getColumns( )==mat2.getColumns( ))
    {
        int i, j;
        int rows=mat1.getRows( );
        int columns=mat1.getColumns( );
        Matrix mat(rows, columns);
        for(i=0; i<rows; i++)
            for(j=0; j<columns; j++)
                mat(i,j)=mat1(i, j)+mat2(i, j);
        return mat;
    }
    else
    {
        cout<<"Error:维数不匹配！ ";
        exit(1);
    }
}

Matrix& operator –(Matrix mat1, Matrix mat2)  //重载减法运算符–
```

```
{
    if(mat1.getRows( )==mat2.getRows( ) && mat1.getColumns( )==mat2.getColumns( ))
    {
        int i, j;
        int rows=mat1.getRows( );
        int columns=mat1.getColumns( );
        Matrix mat(rows, columns);
        for(i=0; i<rows; i++)
            for(j=0; j<columns; j++)
                mat(i,j)=mat1(i, j)−mat2(i, j);
        return mat;
    }
    else
    {
        cout<<"Error:维数不匹配！";
        exit(1);
    }
}

Matrix& operator ∗(Matrix mat1, Matrix mat2) //重载乘法运算符∗，实现两个矩阵相乘
{
    if(mat1.getColumns( )==mat2.getRows( ))
    {
        int i,j,k;
        int rows=mat1.getRows( );
        int columns=mat2.getColumns( );
        int mid=mat1.getColumns( );
        Matrix mat(rows, columns);
        for(i=0; i<rows; i++)
            for(j=0; j<columns; j++)
                for(k=0; k<mid; k++)
                    mat(i, j)=mat(i, j)+mat1(i, k)∗mat2(k, j);
        return mat;
    }
    else
    {
        cout<<"Error:维数不匹配！";
        exit(1);
    }
}

Matrix& operator ∗(Matrix mat, double k) //重载乘法运算符∗，实现矩阵与实数相乘
{
    int i,j;
    int rows=mat.getRows( );
    int columns=mat.getColumns( );
    Matrix matrix(rows, columns);
    for(i=0; i<rows; i++)
```

```cpp
        for(j=0; j<columns; j++)
            matrix(i, j)=mat(i, j)*k;
    return matrix;
}

Matrix& operator *(double k, Matrix mat) //重载乘法运算符*，实现实数与矩阵相乘
{
    Matrix matrix(mat.getRows( ),mat.getColumns( ));
    matrix=mat*k;
    return matrix;
}

int main( )
{
    Matrix mat1(2, 3);
    Matrix mat2(2, 3);
    Matrix mat3(2, 2);
    mat1(0, 0)=0;
    mat1(0, 1)=1;
    mat1(0, 2)=2;
    mat1(1, 0)=3;
    mat1(1, 1)=4;
    mat1(1, 2)=5;
    cout<<mat1;
    mat2 = mat1*2;
    cout<<mat2;

    // mat2=mat1*2-mat1;
    // mat3=mat1*(~mat2);

    return 0;
}
```

运行结果：



E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap03\prog0320...

```
0, 1, 2;
3, 4, 5;
下标越界!

Process returned 1 (0x1)   execution time : 0.117 s
Press any key to continue.
```

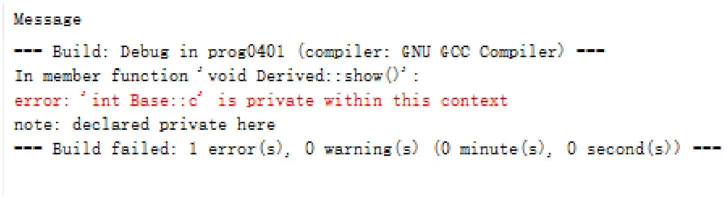
图 3.20: 运行结果

第 4 章　继承与派生

4.1　ex01p90-公有继承对基类成员访问性的影响

```cpp
#include <iostream>

using namespace std;

class Base
{
public:
    Base( )
    {
        a=1;
        b=2;
        c=3;
    }
    int a;
protected:
    int b;
private:
    int c; // c是Base类的私有成员
};

class Derived: public Base
{
public:
    void show( )
    {
        cout<<a<<endl;  //a的访问属性是public
        cout<<b<<endl;  //b的访问属性是protected
        cout<<c<<endl;  //错误，c不可访问
    }
};

int main( )
{
    Derived d;
    d.show( );
    return 0;
}
```

运行结果：



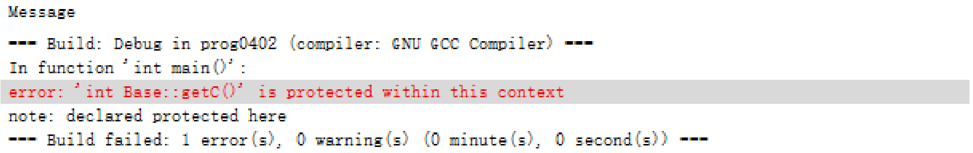
```
Message
--- Build: Debug in prog0401 (compiler: GNU GCC Compiler) ---
In member function 'void Derived::show()':
error: 'int Base::c' is private within this context
note: declared private here
--- Build failed: 1 error(s), 0 warning(s) (0 minute(s), 0 second(s)) ---
```

图 4.1: 运行结果

4.2 ex02p91-通过函数接口来访问私有成员

```
#include <iostream>

using namespace std;

class Base
{
  public:
  Base( )
  {
    a=1;
    b=2;
    c=3;
  }
  int a;
  protected:
  int b;
  int getC( ) //通过对外接口访问私有成员
  {
    return c;
  }
  private:
  int c;
};

class Derived: public Base
{
  public:
  void show( )
  {
    cout<<a<<endl;
    cout<<b<<endl;
    cout<<getC( )<<endl;
  }
};

int main( )
{
  Derived d;
  d.show( );
  cout<<d.getC( );  //错误，函数getC( )的访问属性是protected，不能被外界访问
  return 0;
}
```

运行结果：



```
Message
--- Build: Debug in prog0402 (compiler: GNU GCC Compiler) ---
In function 'int main()':
error: 'int Base::getC()' is protected within this context
note: declared protected here
--- Build failed: 1 error(s), 0 warning(s) (0 minute(s), 0 second(s)) ---
```

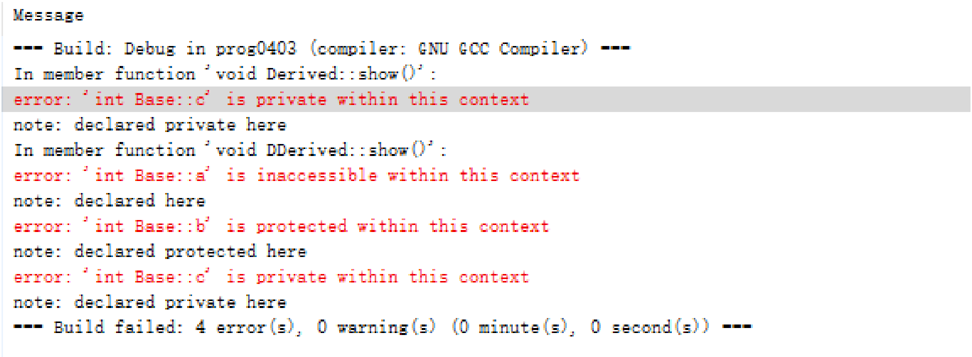
图 4.2: 运行结果

4.3 ex03p92-私有继承对基类成员访问性的影响

```
#include <iostream>

using namespace std;

class Base
{
public:
   Base( )
   {
      a=1;
      b=2;
      c=3;
   }
   int a;
protected:
   int b;
private:
   int c;
};

class Derived: private Base
{
public:
   void show( )
   {
      cout<<a<<endl;
      cout<<b<<endl;
      cout<<c<<endl; //错误，c不可访问
   }
};
```

```
class DDerived: private Derived
{
public:
    void show( )
    {
        cout<<a<<endl; //错误，a不可访问
        cout<<b<<endl; //错误，b不可访问
        cout<<c<<endl; //错误，c不可访问
    }
};

int main( )
{
    Derived de;
    DDerived dde;
    de.show( );
    dde.show( );
    return 0;
}
```

运行结果：



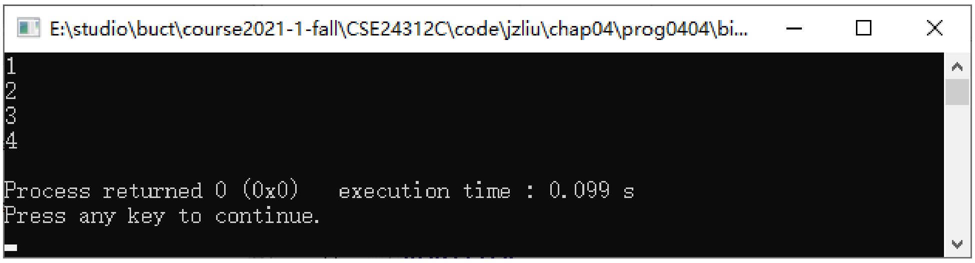
```
Message
--- Build: Debug in prog0403 (compiler: GNU GCC Compiler) ---
In member function 'void Derived::show()':
error: 'int Base::c' is private within this context
note: declared private here
In member function 'void DDerived::show()':
error: 'int Base::a' is inaccessible within this context
note: declared here
error: 'int Base::b' is protected within this context
note: declared protected here
error: 'int Base::c' is private within this context
note: declared private here
--- Build failed: 4 error(s), 0 warning(s) (0 minute(s), 0 second(s)) ---
```

图 4.3: 运行结果

4.4 ex04p93-使用指针访问不可访问的成员

```
#include <iostream>

using namespace std;

class Base
{
public:
    Base( )
    {
        a=1;
        b=2;
        c=3;
    }
    int a;
protected:
    int b;
private:
    int c;
};

class Derived: private Base
{
public:
    Derived( ):Base( ) //派生类构造函数调用基类的构造函数，详见4.4小节
    {
        d=4;
    }
private:
    int d;
};

int main( )
{
    Derived de;
    int *p=(int *)(&de);
    cout<<*p<<endl;  //输出Derived的私有成员a
    cout<<*(p+1) <<endl;  //输出Derived的私有成员b
    cout<<*(p+2) <<endl;  //输出Derived的不可直接访问成员c
    cout<<*(p+3) <<endl;  //输出Derived新增的私有成员d
    return 0;
}
```

运行结果：


E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap04\prog0404\bi...

```
1
2
3
4

Process returned 0 (0x0)   execution time : 0.099 s
Press any key to continue.
```

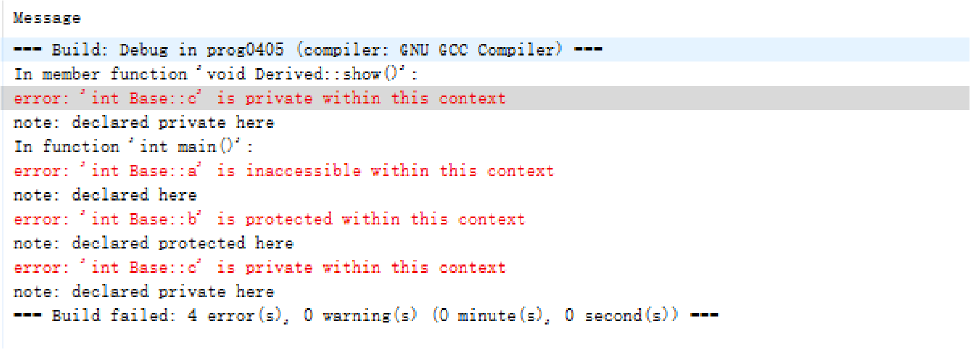
图 4.4: 运行结果

4.5 ex05p95-保护继承对基类成员访问性的影响

```
#include <iostream>
```

```
using namespace std;

class Base
{
  public:
  Base( )
  {
    a=1;
    b=2;
    c=3;
  }
  int a;
  protected:
  int b;
  private:
  int c;
};

class Derived: protected Base
{
  public:
    void show( )
    {
      cout<<a<<endl;
      cout<<b<<endl;
      cout<<c<<endl;  //错误，不能直接访问
    }
};

int main( )
{
  Derived de;
  de.show( );

  cout<<de.a<<endl;  //错误，不能访问Derived的保护成员
  cout<<de.b<<endl;  //错误，不能访问Derived的保护成员
  cout<<de.c<<endl;  //错误，不能直接访问
  return 0;
}
```

运行结果：



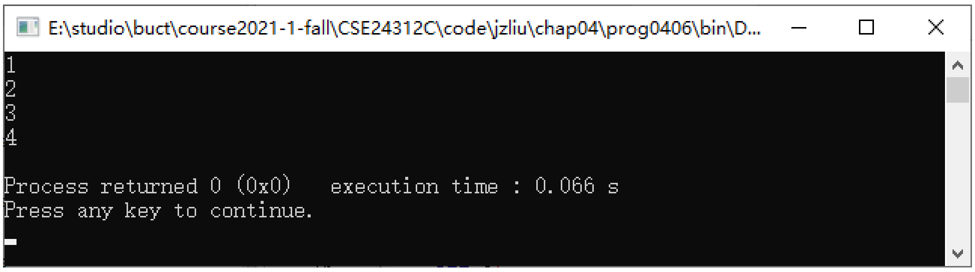
```
Message
--- Build: Debug in prog0405 (compiler: GNU GCC Compiler) ---
In member function 'void Derived::show()':
error: 'int Base::c' is private within this context
note: declared private here
In function 'int main()':
error: 'int Base::a' is inaccessible within this context
note: declared here
error: 'int Base::b' is protected within this context
note: declared protected here
error: 'int Base::c' is private within this context
note: declared private here
--- Build failed: 4 error(s), 0 warning(s) (0 minute(s), 0 second(s)) ---
```

图 4.5: 运行结果

4.6 ex06p96-派生类访问重名的基类成员

```
#include <iostream>

using namespace std;


class Base
{
  public:
  int a;
  int b;
  Base( )
  {
    a=1;
    b=2;
  }
};

class Derived: public Base
{
  public:
  int b; //与基类成员重名
  int c;
  Derived( ):Base( )
  {
    b=3;
    c=4;
  }
    void show( )
    {
      cout<<a<<endl;
      cout<<Base::b<<endl;  //输出从Base中继承来的成员
      cout<<b<<endl;    //输出Derived中新增的成员
      cout<<c<<endl;
    }
};

int main( )
{
  Derived de;
  de.show( );
  return 0;
}
```

运行结果:



E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap04\prog0406\bin\D...

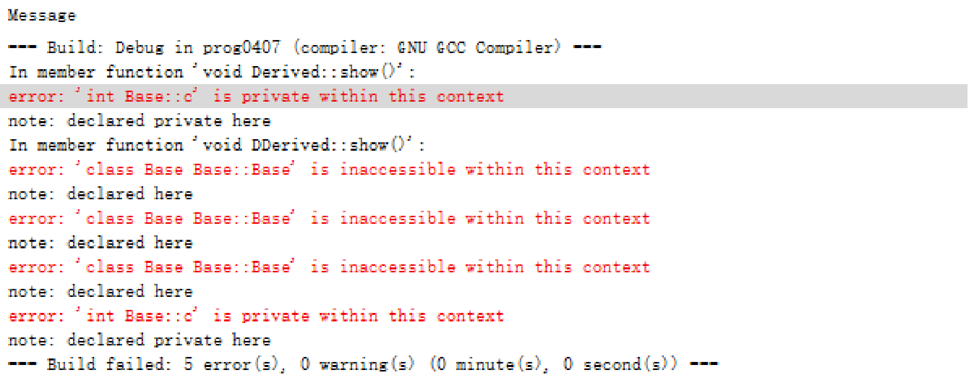
```
1
2
3
4

Process returned 0 (0x0)   execution time : 0.066 s
Press any key to continue.
```

图 4.6: 运行结果

4.7 ex07p97-派生类访问基类的静态成员【有问题：不可访问私有成员】

```
#include <iostream>

using namespace std;


class Base
{
public:
    static int a;
protected:
    static int b;
private:
    static int c;
};
int Base::a=1;  //静态成员初始化
int Base::b=2;  //静态成员初始化
int Base::c=3;  //静态成员初始化

class Derived: private Base
{
public:
    void show( )
    {
        cout<<a<<endl;  //成员a处于可访问状态下
        cout<<b<<endl;  //成员b处于可访问状态下
        cout<<Base::c<<endl; //成员c处于不可访问状态下，必须指定作用域
    }
};

class DDerived: private Derived
{
public:
    void show( )
    {
        cout<<Base::a<<endl;  //成员a处于不可访问状态下，必须指定作用域
        cout<<Base::b<<endl;  //成员b处于不可访问状态下，必须指定作用域
        cout<<Base::c<<endl;  //成员c处于不可访问状态下，必须指定作用域
    }
};

int main( )
{
    Derived de;
    DDerived dde;
    de.show( );
    dde.show( );
    return 0;
}
```

运行结果:



```
Message
--- Build: Debug in prog0407 (compiler: GNU GCC Compiler) ---
In member function 'void Derived::show()':
error: 'int Base::c' is private within this context
note: declared private here
In member function 'void DDerived::show()':
error: 'class Base Base::Base' is inaccessible within this context
note: declared here
error: 'class Base Base::Base' is inaccessible within this context
note: declared here
error: 'class Base Base::Base' is inaccessible within this context
note: declared here
error: 'int Base::c' is private within this context
note: declared private here
--- Build failed: 5 error(s), 0 warning(s) (0 minute(s), 0 second(s)) ---
```

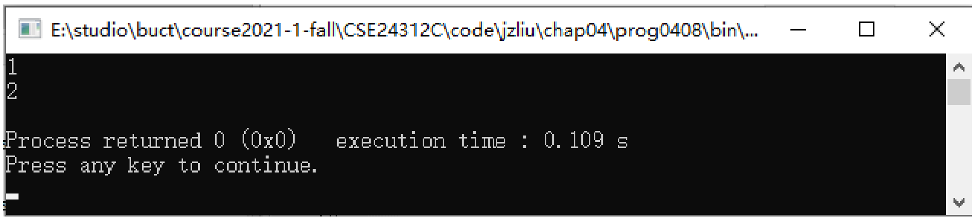
图 4.7: 运行结果

4.8 ex08p99-用访问声明的方法使得派生类中成员的访问属性得到调整

```
#include <iostream>
```

```
using namespace std;


class Base
{
  public:
  Base( )
  {
    a=1;
    b=2;
    c=3;
  }
  int a;
  protected:
  int b;
  private:
  int c;
};

class Derived: private Base
{
  public:
  Base::a;  //调整a的访问属性为公有属性
  protected:
  Base::b;  //调整b的访问属性为保护属性
};

class DDerived: private Derived
{
  public:
  void show( )
  {
    cout<<b<<endl; //b是DDerived的私有成员
  }
};

int main( )
{
  Derived de;
  DDerived dde;
  cout<<de.a<<endl;  //a是Derived的公有成员，可以直接访问
  dde.show( );
  return 0;
}
```

运行结果：


E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap04\prog0408\bin\...

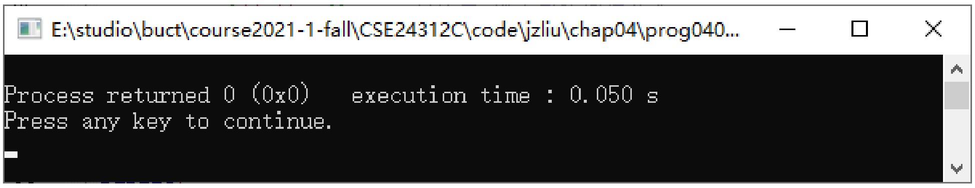
```
1
2

Process returned 0 (0x0)   execution time : 0.109 s
Press any key to continue.
```

图 4.8: 运行结果

4.9 ex09p103-基类中有不带参数的构造函数的情况

```cpp
#include <iostream>

using namespace std;

class Base
{
public:
    int a;
    int b;
    Base( )
    {
        a=1;
        b=2;
    }
};

class Derived: public Base
{
public:
    int c;
    Derived( )
    {
        c=3;
    }
    void show( )
    {
        cout<<a<<","<<b<<","<<c<<endl;
    }
};

int main( )
{
    Derived de;
    de.show( );
    return 0;
}
```

运行结果：



E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap04\prog040...

```
Process returned 0 (0x0)   execution time : 0.050 s
Press any key to continue.
```

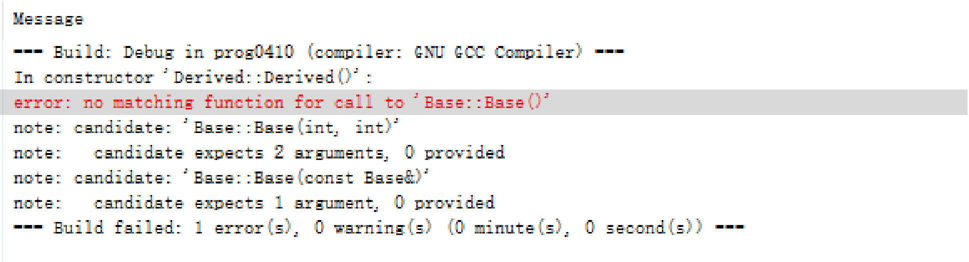
图 4.9: 运行结果

4.10 ex10p104-基类中有带参数的构造函数的情况

```cpp
#include <iostream>

using namespace std;

class Base
{
public:
    int a;
    int b;
    Base(int x, int y)
    {
        a=x;
        b=y;
    }
};

class Derived: public Base
{
public:
    int c;
    Derived( ) //错误，派生类会去调用基类的中不带参数的构造函数
    {
        c=3;
    }
    Derived(int x, int y, int z): Base(x, y) //正确，派生类显式调用基类中的构造函数
    {
        a=x;
        b=y;
        c=z;
    }
    void show( )
    {
        cout<<a<<","<<b<<","<<c<<endl;
    }
};
```

```
int main( )
{
   Derived de;
   de.show( );
   return 0;
}
```

运行结果：



```
Message
--- Build: Debug in prog0410 (compiler: GNU GCC Compiler) ---
In constructor 'Derived::Derived()':
error: no matching function for call to 'Base::Base()'
note: candidate: 'Base::Base(int, int)'
note:   candidate expects 2 arguments, 0 provided
note: candidate: 'Base::Base(const Base&)'
note:   candidate expects 1 argument, 0 provided
--- Build failed: 1 error(s), 0 warning(s) (0 minute(s), 0 second(s)) ---
```

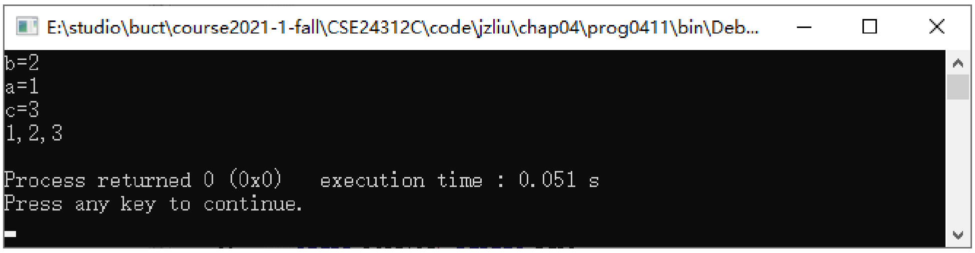
图 4.10: 运行结果

4.11 ex11p105-多重继承时派生类定义构造函数的情况

```
#include <iostream>

using namespace std;

class Base1
{
public:
   int a;
   Base1(int x)
   {
      a=x;
      cout<<"a="<<a<<endl;
   }
};

class Base2
{
public:
   int b;
   Base2(int y)
   {
      b=y;
      cout<<"b="<<b<<endl;
   }
};

class Derived: public Base2, Base1  //这里的摆放顺序有意义
```

```
{
public:
   int c;
   Derived(int x, int y, int z): Base1(x), Base2(y)  //这里的摆放顺序不重要
   {
      c=z;
      cout<<"c="<<c<<endl;
   }
   void show( )
   {
      cout<<a<<","<<b<<","<<c<<endl;
   }
};

int main( )
{
   Derived de(1, 2, 3);
   de.show( );
   return 0;
}
```

运行结果:



E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap04\prog0411\bin\Deb...

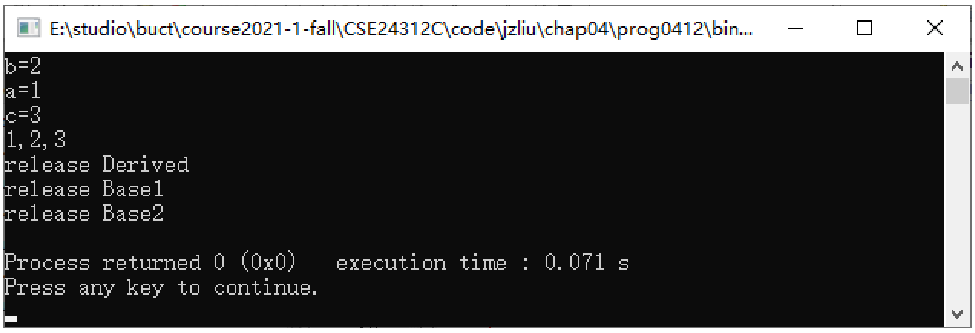
```
b=2
a=1
c=3
1,2,3

Process returned 0 (0x0)   execution time : 0.051 s
Press any key to continue.
```

图 4.11: 运行结果

4.12 ex12p107-派生类在调用析构函数时的调用顺序

```
#include <iostream>

using namespace std;

class Base1
{
  public:
  int a;
  Base1(int x)
  {
    a=x;
    cout<<"a="<<a<<endl;
  }
  ~Base1( )
  {
    cout<<"release Base1"<<endl;
  }
};

class Base2
{
  public:
  int b;
  Base2(int y)
  {
    b=y;
    cout<<"b="<<b<<endl;
  }
  ~Base2( )
  {
    cout<<"release Base2"<<endl;
  }
};

class Derived: public Base2, Base1  //这里的摆放顺序有意义
{
  public:
  int c;
  Derived(int x, int y, int z): Base1(x), Base2(y)  //这里的摆放顺序不重要
  {
    c=z;
    cout<<"c="<<c<<endl;
  }
  void show( )
  {
    cout<<a<<","<<b<<","<<c<<endl;
  }
  ~Derived( )
  {
    cout<<"release Derived"<<endl;
  }
};

int main( )
{
  Derived de(1, 2, 3);
  de.show( );
  return 0;
}
```

运行结果:



E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap04\prog0412\bin...

```
b=2
a=1
c=3
1,2,3
release Derived
release Base1
release Base2

Process returned 0 (0x0)   execution time : 0.071 s
Press any key to continue.
```

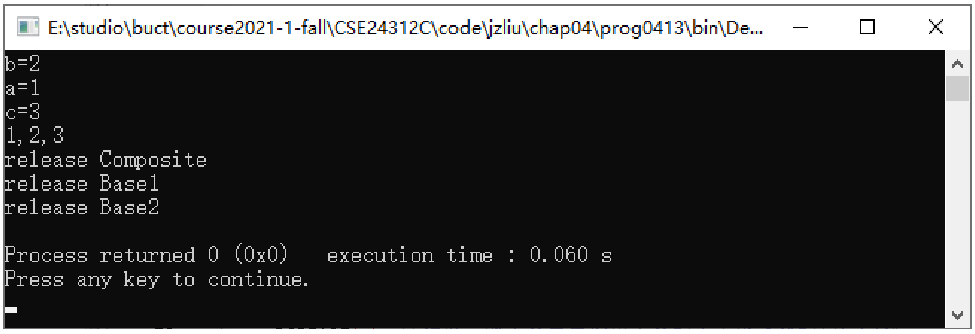
图 4.12: 运行结果

4.13 ex13p108-具有无参数构造函数的对象成员的构造和析构过程

```
#include <iostream>

using namespace std;


class Base1
{
  public:
  int a;
  Base1( )
  {
    a=1;
    cout<<"a="<<a<<endl;
  }
  ~Base1( )
  {
    cout<<"release Base1"<<endl;
  }
};

class Base2
{
  public:
  int b;
  Base2( )
  {
    b=2;
    cout<<"b="<<b<<endl;
  }
  ~Base2( )
  {
    cout<<"release Base2"<<endl;
  }
};
```

```
class Composite
{
  public:
  int c;
  Base2 b2;  //这里的摆放顺序有意义
  Base1 b1;  //这里的摆放顺序有意义
  Composite( )
  {
    c=3;
    cout<<"c="<<c<<endl;
  }
  void show( )
  {
    cout<<b1.a<<","<<b2.b<<","<<c<<endl;
  }
  ~Composite( )
  {
    cout<<"release Composite"<<endl;
  }
};

int main( )
{
  Composite co;
  co.show( );
  return 0;
}
```

运行结果：


E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap04\prog0413\bin\De...

```
b=2
a=1
c=3
1,2,3
release Composite
release Base1
release Base2

Process returned 0 (0x0)   execution time : 0.060 s
Press any key to continue.
```

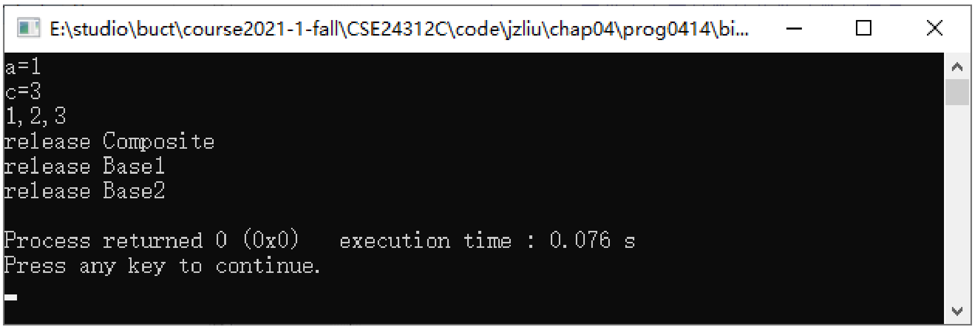
图 4.13: 运行结果

4.14 ex14p110-没有无参数构造函数的对象成员的构造和析构过程

```
#include <iostream>

using namespace std;
```

```cpp
class Base1
{
public:
    int a;
    Base1(int x)
    {
        a=x;
        cout<<"a="<<a<<endl;
    }
    ~Base1( )
    {
        cout<<"release Base1"<<endl;
    }
};

class Base2
{
public:
    int b;
    Base2(int y)
    {
        b=y;
        cout<<"b="<<b<<endl;
    }
    ~Base2( )
    {
        cout<<"release Base2"<<endl;
    }
};

class Composite
{
public:
    int c;
    Base2 b2;  //这里的摆放顺序有意义
    Base1 b1;  //这里的摆放顺序有意义
    Composite(int x ,int y, int z): b1(x), b2(y)  //这里的摆放顺序不重要
    {
        c=z;
        cout<<"c="<<c<<endl;
    }
    void show( )
    {
        cout<<b1.a<<","<<b2.b<<","<<c<<endl;
    }
    ~Composite( )
    {
        cout<<"release Composite"<<endl;
    }
};

int main( )
{
    Composite co(1, 2, 3);
    co.show( );
    return 0;
}
```

运行结果：


E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap04\prog0414\bi...

```
a=1
c=3
1,2,3
release Composite
release Base1
release Base2

Process returned 0 (0x0)   execution time : 0.076 s
Press any key to continue.
```

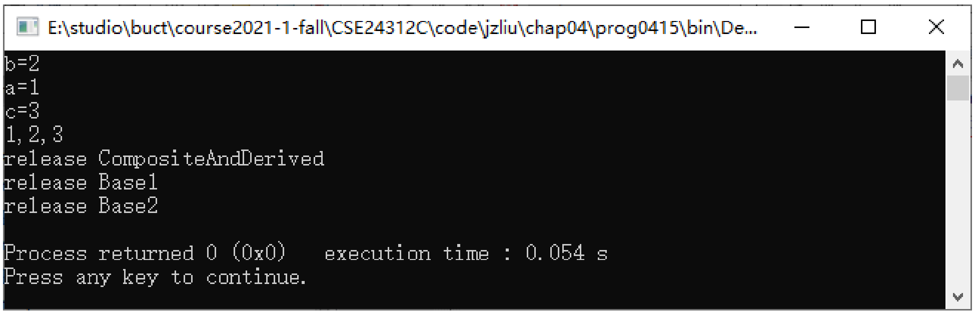
图 4.14: 运行结果

4.15 ex15p112-包含了基类成员初始化和对象成员初始化的派生类构造函数

```
#include <iostream>

using namespace std;

class Base1
{
public:
    int a;
    Base1(int x)
    {
        a=x;
        cout<<"a="<<a<<endl;
    }
    ~Base1( )
    {
        cout<<"release Base1"<<endl;
    }
};
```

```
class Base2
{
public:
   int b;
   Base2(int y)
   {
      b=y;
      cout<<"b="<<b<<endl;
   }
   ~Base2( )
   {
      cout<<"release Base2"<<endl;
   }
};

class CompositeAndDerived: public Base2
{
public:
   int c;
   Base1 b1;
   CompositeAndDerived(int x ,int y, int z): b1(x), Base2(y)  //这里的摆放顺序不重要
   {
      c=z;
      cout<<"c="<<c<<endl;
   }
   void show( )
   {
      cout<<b1.a<<","<<b<<","<<c<<endl;
   }
   ~CompositeAndDerived( )
   {
      cout<<"release CompositeAndDerived"<<endl;
   }
};

int main( )
{
   CompositeAndDerived cd(1, 2, 3);
   cd.show( );
   return 0;
}
```

运行结果:


E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap04\prog0415\bin\De...

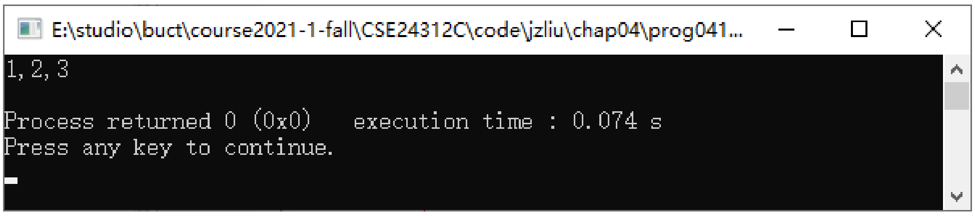
```
b=2
a=1
c=3
1,2,3
release CompositeAndDerived
release Base1
release Base2

Process returned 0 (0x0)   execution time : 0.054 s
Press any key to continue.
```

图 4.15: 运行结果

4.16 ex16p114-实现一个多重继承的例子

```
#include <iostream>

using namespace std;

class Base1
{
public:
    int a;
    Base1( )
    {
        a=1;
    }
};

class Base2
{
public:
    int b;
    Base2( )
    {
        b=2;
    }
};

class Derived: public Base1, Base2
{
public:
    int c;
    Derived( )
    {
        c=3;
    }
    void show( )
    {
        cout<<a<<","<<b<<","<<c<<endl;
    }
};

int main( )
{
    Derived de;
    de.show( );
    return 0;
}
```

运行结果：



E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap04\prog041...

```
1,2,3

Process returned 0 (0x0)   execution time : 0.074 s
Press any key to continue.
```

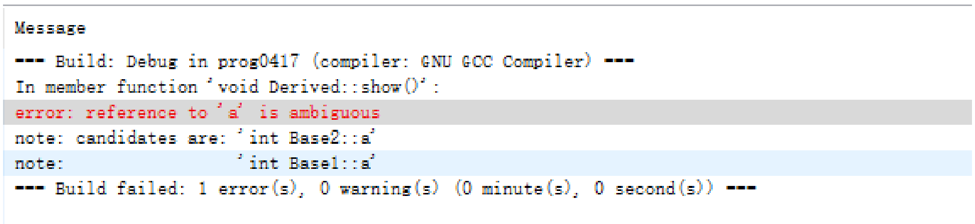
图 4.16: 运行结果

4.17 ex17p115-多重继承中的二义性

```
#include <iostream>

using namespace std;

class Base1
{
  public:
  int a;
  Base1( )
  {
    a=1;  //Base1类中定义了一个名字是a的数据成员
  }
};

class Base2
{
  public:
  int a;
  Base2( )
  {
    a=2;  //Base2类中也定义了一个名字是a的数据成员
  }
};
```

```
class Derived: public Base1, Base2
{
  public:
  int c;
  Derived( )
  {
    c=3;
  }
  void show( )
  {
    cout<<a<<endl;    //错误，a存在二义性
    cout<<Base1::a<<endl;  //正确，消除了a的二义性
cout<<Base2::a<<endl;  //正确，消除了a的二义性
cout<<c<<endl;
  }
};

int main( )
{
  Derived de;
  de.show( );
  return 0;
}
```

运行结果：



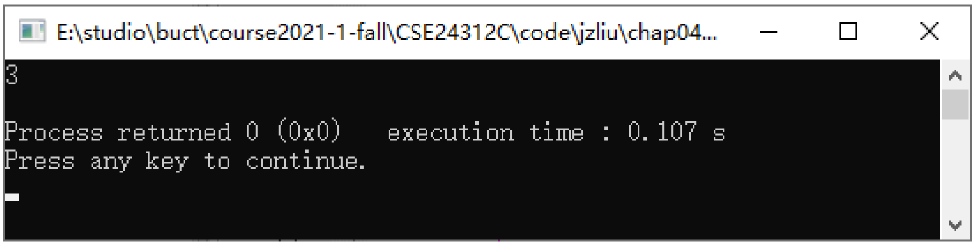
```
Message
=== Build: Debug in prog0417 (compiler: GNU GCC Compiler) ===
In member function 'void Derived::show()':
error: reference to 'a' is ambiguous
note: candidates are: 'int Base2::a'
note:                 'int Base1::a'
=== Build failed: 1 error(s), 0 warning(s) (0 minute(s), 0 second(s)) ===
```

图 4.17: 运行结果

4.18 ex18p116-派生类同名成员覆盖基类中的二义性成员

```
#include <iostream>

using namespace std;

class Base1
{
  public:
  int a;
  Base1( )
  {
    a=1;
  }
};

class Base2
{
  public:
  int a;
  Base2( )
  {
    a=2;
  }
};

class Derived: public Base1, Base2
{
  public:
  int a;
  Derived( )
  {
    a=3;
  }
  void show( )
  {
    cout<<a<<endl; //正确，这里的a是Derived自己新增的成员
  }
};

int main( )
{
  Derived de;
  de.show( );
  return 0;
}
```

运行结果：


E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap04...

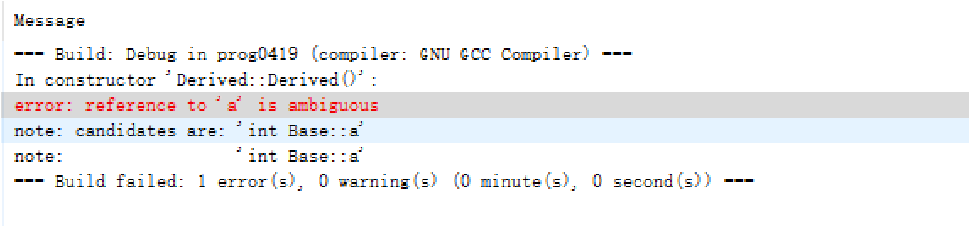
```
3

Process returned 0 (0x0)   execution time : 0.107 s
Press any key to continue.
```

图 4.18: 运行结果

4.19 ex19p117-多重继承来自同一个基类的派生类

```
#include <iostream>

using namespace std;

class Base
{
  public:
  int a;
  Base( )
  {
    a=1;
    cout<<"Base.a="<<a<<endl;
  }
};

class Base1: public Base
{
  public:
  Base1( )
  {
    a=a+1;
    cout<<"Base1.a="<<a<<endl;
  }
};

class Base2: public Base
{
  public:
  Base2( )
  {
    a=a+2;
    cout<<"Base2.a="<<a<<endl;
  }
};

class Derived: public Base1, Base2
{
  public:
  Derived( )
  {
cout<<"a="<<a<<endl;      //错误，a具有二义性
cout<<"Base1::a="<<Base1::a<<endl;  //正确，消除a的二义性
    cout<<"Base2::a="<<Base2::a<<endl;   //正确，消除a的二义性
  }
};

int main( )
{
  Derived de;
  return 0;
}
```

运行结果：



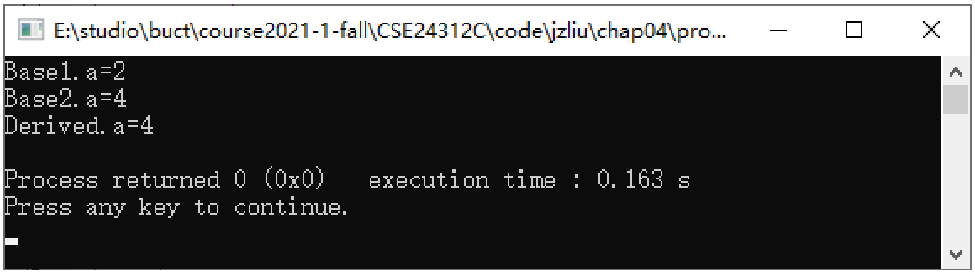
```
Message
--- Build: Debug in prog0419 (compiler: GNU GCC Compiler) ---
In constructor 'Derived::Derived()':
error: reference to 'a' is ambiguous
note: candidates are: 'int Base::a'
note:                 'int Base::a'
--- Build failed: 1 error(s), 0 warning(s) (0 minute(s), 0 second(s)) ---
```

图 4.19: 运行结果

4.20 ex20p119-实现虚基类

```
#include <iostream>

using namespace std;

class Base
{
  public:
  int a;
  Base( )
  {
    a=1;
    cout<<"Base.a="<<a<<endl;
  }
};

class Base1: virtual public Base
{
  public:
  Base1( )
  {
    a=a+1;
    cout<<"Base1.a="<<a<<endl;
  }
};

class Base2: virtual public Base
{
  public:
  Base2( )
  {
    a=a+2;
    cout<<"Base2.a="<<a<<endl;
  }
};

class Derived: public Base1, Base2
{
  public:
  Derived( )
  {
    cout<<"Derived.a="<<a<<endl;
  }
};

int main( )
{
  Derived de;
  return 0;
}
```

运行结果:



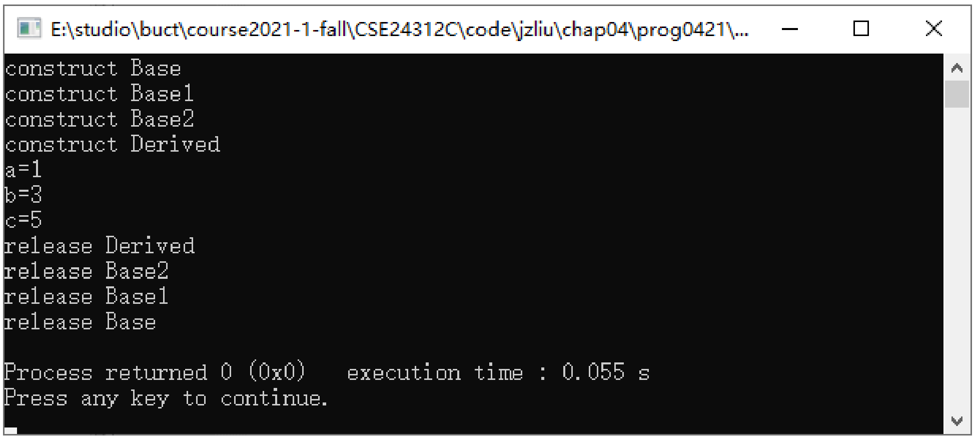
```
Base1.a=2
Base2.a=4
Derived.a=4

Process returned 0 (0x0)   execution time : 0.163 s
Press any key to continue.
```

图 4.20: 运行结果

4.21 ex21p121-带有参构造函数的虚基类的初始化和析构过程

```
#include <iostream>

using namespace std;

class Base
{
public:
    int a;
    Base(int x)
    {
        a=x;
        cout<<"construct Base"<<endl;
    }
    ~Base( )
    {
        cout<<"release Base"<<endl;
    }
};

class Base1: virtual public Base
{
public:
    int b;
    Base1(int x, int y): Base(x)
    {
        b=y;
        cout<<"construct Base1"<<endl;
    }
    ~Base1( )
    {
        cout<<"release Base1"<<endl;
    }
};

class Base2: virtual public Base
{
public:
    int c;
    Base2(int x, int y): Base(x)
    {
        c=y;
        cout<<"construct Base2"<<endl;
    }
    ~Base2( )
    {
        cout<<"release Base2"<<endl;
    }
};

class Derived: public Base1, Base2
{
public:
    Derived(int i1, int i2, int i3, int i4, int i5): Base(i1), Base1(i2, i3), Base2(i4, i5)
    {
        cout<<"construct Derived"<<endl;
    }
    void show( )
    {
        cout<<"a="<<a<<endl;
        cout<<"b="<<b<<endl;
        cout<<"c="<<c<<endl;
    }
    ~Derived( )
    {
        cout<<"release Derived"<<endl;
    }
};

int main( )
{
    Derived de(1,2,3,4,5);
    de.show( );
    return 0;
}
```

运行结果：



E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap04\prog0421\...

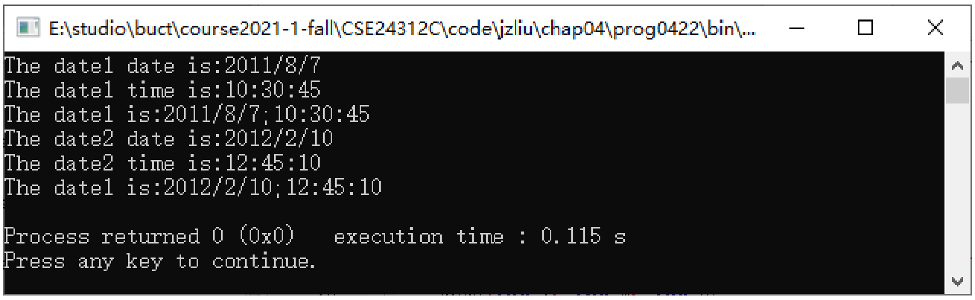
```
construct Base
construct Base1
construct Base2
construct Derived
a=1
b=3
c=5
release Derived
release Base2
release Base1
release Base

Process returned 0 (0x0)   execution time : 0.055 s
Press any key to continue.
```

图 4.21: 运行结果

4.22 ex22p123-应用实例 (用多重继承方式表达日期和时间关系)

```
#include <iostream>
#include "stdio.h"

using namespace std;

class Date
{
public:
    Date( ) { }  //思考，如果不设置这个空的构造函数行不行？

    Date(int y, int m, int d)
    {
        SetDate(y, m, d);
    }

    void SetDate(int y, int m, int d)
    {
        Year = y;
        Month = m;
        Day = d;
    }

    char* GetStringDate(char* DateStr)
    {
        sprintf(DateStr, "%d/%d/%d", Year, Month, Day);
        return DateStr;
    }

protected:
    int Year, Month, Day;
};

class Time
{
public:
    Time( ) { }  //思考，如果不设置这个空的构造函数行不行？

    Time(int h, int m, int s)
    {
        SetTime(h, m, s);
    }

    void SetTime(int h, int m, int s)
    {
        Hours = h;
        Minutes = m;
        Seconds = s;
    }

    char* GetStringTime(char* TimeStr)
    {
        sprintf(TimeStr, "%d:%d:%d", Hours, Minutes, Seconds);
        return TimeStr;
    }

protected:
    int Hours, Minutes, Seconds;
};

class TimeDate:public Date, public Time
{
public:
    TimeDate( ) { }  //思考，如果不设置这个空的构造函数行不行？
    TimeDate(int y, int mo, int d, int h, int mi, int s): Date(y, mo, d), Time(h, mi, s) {
        }

    char* GetStringDT(char* DTstr)
    {
        sprintf(DTstr, "%d/%d/%d;%d:%d:%d", Year,Month,Day,Hours,Minutes,Seconds);
        return DTstr;
    }
};

int main( )
{
    TimeDate date1, date2(2012, 2, 10, 12, 45, 10);
    char DemoStr[80];
    date1.SetDate(2011, 8, 7);
    date1.SetTime(10, 30, 45);
    cout<<"The date1 date is:"<<date1.GetStringDate(DemoStr)<<endl;
    cout<<"The date1 time is:"<<date1.GetStringTime(DemoStr)<<endl;
    cout<<"The date1 is:"<<date1.GetStringDT(DemoStr)<<endl;
    cout<<"The date2 date is:"<<date2.GetStringDate(DemoStr)<<endl;
    cout<<"The date2 time is:"<<date2.GetStringTime(DemoStr)<<endl;
    cout<<"The date1 is:"<<date2.GetStringDT(DemoStr)<<endl;
    return 0;
}
```

运行结果：


E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap04\prog0422\bin\...

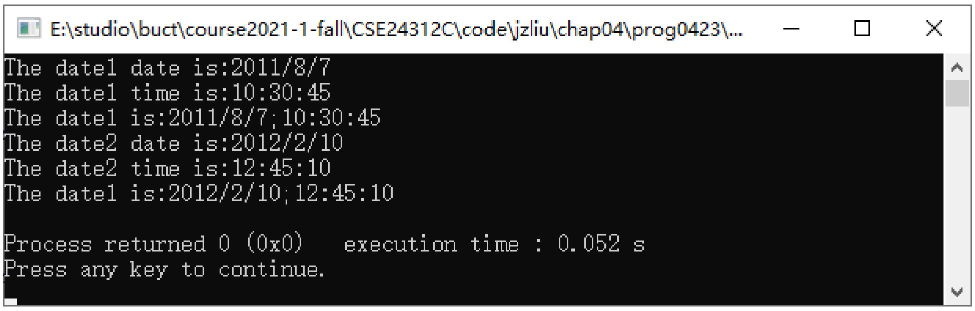
```
The date1 date is:2011/8/7
The date1 time is:10:30:45
The date1 is:2011/8/7;10:30:45
The date2 date is:2012/2/10
The date2 time is:12:45:10
The date1 is:2012/2/10;12:45:10

Process returned 0 (0x0)   execution time : 0.115 s
Press any key to continue.
```

图 4.22: 运行结果

4.23 ex23p125-应用实例 (用组合方式表达日期和时间关系)

```
#include <iostream>
#include "stdio.h"

using namespace std;

class Date
{
public:
    Date( ) { }  //思考，如果不设置这个空的构造函数行不行？

    Date(int y, int m, int d)
    {
        SetDate(y, m, d);
    }

    void SetDate(int y, int m, int d)
    {
        Year = y;
        Month = m;
        Day = d;
    }

    char* GetStringDate(char* DateStr)
    {
        sprintf(DateStr, "%d/%d/%d", Year, Month, Day);
        return DateStr;
    }

protected:
    int Year, Month, Day;
};

class Time
{
public:
    Time( ) { }  //思考，如果不设置这个空的构造函数行不行？

    Time(int h, int m, int s)
    {
        SetTime(h, m, s);
    }

    void SetTime(int h, int m, int s)
    {
        Hours = h;
        Minutes = m;
        Seconds = s;
    }

    char* GetStringTime(char* TimeStr)
    {
        sprintf(TimeStr, "%d:%d:%d", Hours, Minutes, Seconds);
        return TimeStr;
    }

protected:
    int Hours, Minutes, Seconds;
};

class TimeDate
{
public:
    TimeDate( ) { }  //思考，如果不设置这个空的构造函数行不行？

    TimeDate(int y, int mo, int d, int h, int mi, int s): date(y, mo, d), time(h, mi, s) {
        }

    char* GetStringDT(char* DTstr)
    {
        char strD[80], strT[80];
        sprintf(DTstr, "%s;%s", date.GetStringDate(strD), time.GetStringTime(strT));
        return DTstr;
    }

    void SetDate(int y, int m, int d)
    {
        date.SetDate(y, m, d);
    }

    void SetTime(int h, int m, int s)
    {
        time.SetTime(h, m, s);
    }

    char* GetStringDate(char* DateStr)
    {
        date.GetStringDate(DateStr);
        return DateStr;
    }

    char* GetStringTime(char* TimeStr)
    {
        time.GetStringTime(TimeStr);
        return TimeStr;
    }

protected:
    Date date;
    Time time;
};

int main( )
{
    TimeDate date1, date2(2012, 2, 10, 12, 45, 10);
    char DemoStr[80];
    date1.SetDate(2011, 8, 7);
    date1.SetTime(10, 30, 45);
    cout<<"The date1 date is:"<<date1.GetStringDate(DemoStr)<<endl;
    cout<<"The date1 time is:"<<date1.GetStringTime(DemoStr)<<endl;
    cout<<"The date1 is:"<<date1.GetStringDT(DemoStr)<<endl;
    cout<<"The date2 date is:"<<date2.GetStringDate(DemoStr)<<endl;
    cout<<"The date2 time is:"<<date2.GetStringTime(DemoStr)<<endl;
    cout<<"The date1 is:"<<date2.GetStringDT(DemoStr)<<endl;
    return 0;
}
```

运行结果：


E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap04\prog0423\...

```
The date1 date is:2011/8/7
The date1 time is:10:30:45
The date1 is:2011/8/7;10:30:45
The date2 date is:2012/2/10
The date2 time is:12:45:10
The date1 is:2012/2/10;12:45:10

Process returned 0 (0x0)   execution time : 0.052 s
Press any key to continue.
```

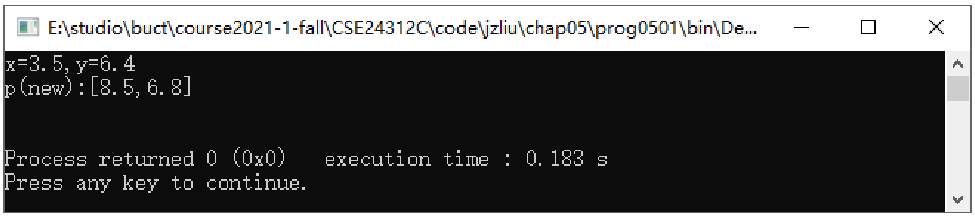
图 4.23: 运行结果

第 5 章　多态性

5.1　ex01p130-一个多态性的例子 (点、圆、柱面)

```
#include<iostream>

using namespace std;

//声明类Point
class Point
{
public:
    Point(float x=0,float y=0);//有默认参数的构造函数
    void setPoint(float ,float);//设置坐标值
    float getX( )const
    {
        return x; //读x坐标
    }
    float getY( )const
    {
        return y; //读y坐标
    }
    friend ostream & operator <<(ostream &,const Point &);//重载运算符"<<"
protected ://受保护成员
    float x, y;
};

//下面定义Point类的成员函数
Point::Point(float a,float b) //Point的构造函数
{
    x=a; //对x,y初始化
    y=b;
}

void Point::setPoint(float a,float b) //设置x和y的坐标值
{
    x=a; //为x,y赋新值
    y=b;
}

ostream & operator <<(ostream &output, const Point &p)
{
    //重载运算符"<<"，使之能输出点的坐标
    output<<"["<<p.x<<","<<p.y<<"]"<<endl;
    return output;
}

int main( )
{
    Point p(3.5,6.4);//建立Point类对象p
    cout<<"x="<<p.getX( )<<",y="<<p.getY( )<<endl;//输出p的坐标值
    p.setPoint(8.5,6.8);//重新设置p的坐标值
    cout<<"p(new):"<<p<<endl;//用重载运算符"<<"输出p点坐标
    return 0;
}
```

运行结果：



E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap05\prog0501\bin\De...

```
x=3.5,y=6.4
p(new):[8.5,6.8]


Process returned 0 (0x0)   execution time : 0.183 s
Press any key to continue.
```

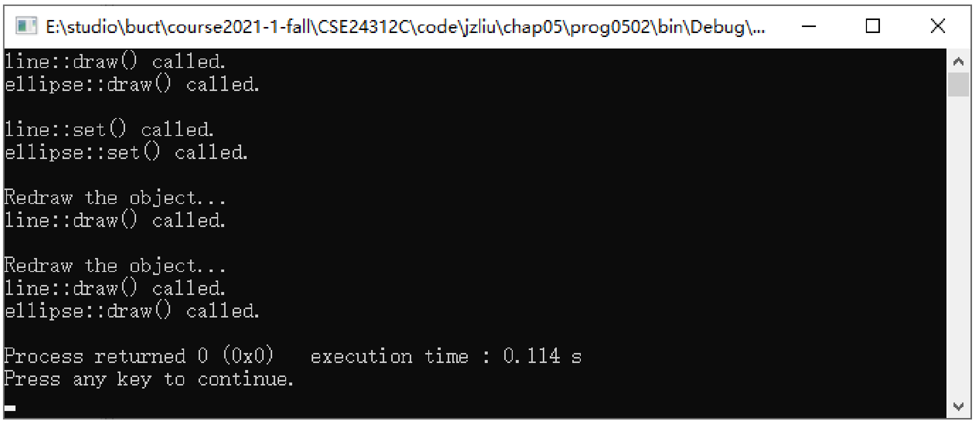
图 5.1: 运行结果

5.2 ex02p137-纯虚函数的使用方法

```
#include<iostream>
```

```
using namespace std;

class point
{
public:
    point(int i=0, int j=0)
    {
        x0=i;
        y0=j;
    }
    virtual void set() = 0;
    virtual void draw() = 0;
protected:
    int x0, y0;
};
class line : public point
{
public:
    line(int i=0, int j=0, int m=0, int n=0):point(i, j)
    {
        x1=m;
        y1=n;
    }
    void set()
    {
        cout<<"line::set() called.\n";
    }
    void draw()
    {
        cout<<"line::draw() called.\n";
    }
protected:
    int x1, y1;
};
class ellipse : public point
{
public:
    ellipse(int i=0, int j=0, int p=0, int q=0):point(i, j)
    {
        x2=p;
        y2=q;
    }
    void set()
    {
        cout<<"ellipse::set() called.\n";
    }
    void draw()
    {
        cout<<"ellipse::draw() called.\n";
    }
protected:
    int x2, y2;
};
void drawobj(point *p)
{
    p->draw();
}
void setobj(point *p)
{
    p->set();
}
int main()
{
    line *lineobj = new line;
    ellipse *elliobj = new ellipse;
    drawobj(lineobj);
    drawobj(elliobj);
    cout<<endl;
    setobj(lineobj);
    setobj(elliobj);
    cout<<"\nRedraw the object...\n";
    drawobj(lineobj);
    cout<<"\nRedraw the object...\n";
    drawobj(lineobj);
    drawobj(elliobj);
    return 0;
}
```

运行结果：



E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap05\prog0502\bin\Debug\...

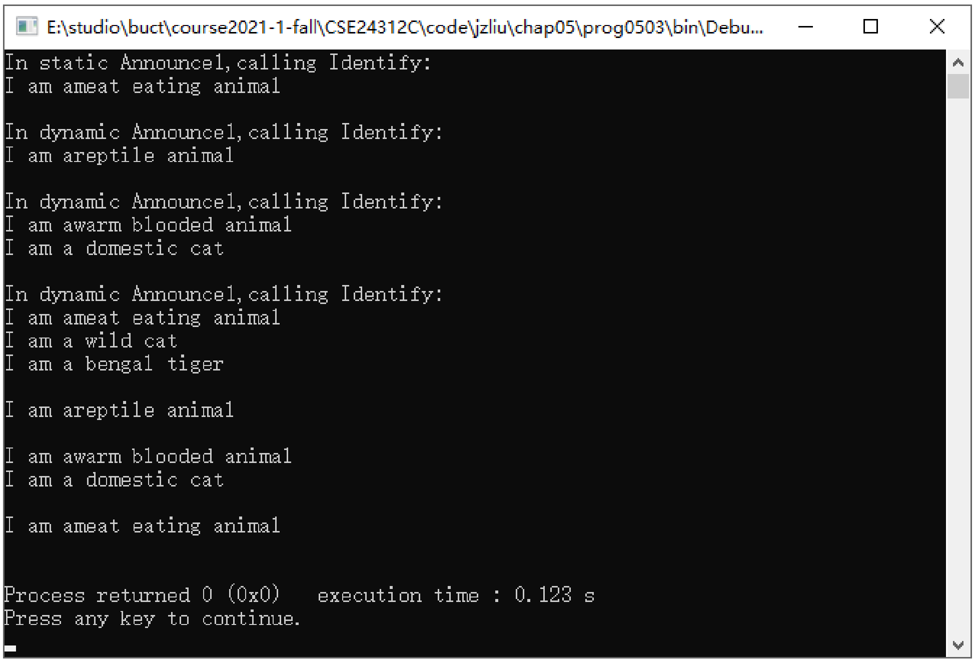
```
line::draw() called.
ellipse::draw() called.

line::set() called.
ellipse::set() called.

Redraw the object...
line::draw() called.

Redraw the object...
line::draw() called.
ellipse::draw() called.

Process returned 0 (0x0)   execution time : 0.114 s
Press any key to continue.
```

图 5.2: 运行结果

5.3 ex03p139-虚函数多态性应用例子 (动物猫科动物老虎)

```
#include<iostream>
#include<string.h>

using namespace std;


class Animal
{
private:
    char animalName[20]; //动物的名字字符串
public:
    Animal(char nmap[]) //动物类的构造函数
    {
        strcpy(animalName,nmap); //把名字字符串传递给特定字符串
    }
//Identity函数用于输出该字符串来标识调用该函数的对象
    virtual void Identify(void)
    {
        cout<<"I am a" << animalName <<" animal" << endl;
    }
};
//下面是猫科类的声明:
class Cat:public Animal
{
private:
    char catName[20];
public:
    Cat(char nmc[],char nma[]):Animal(nma)
    {
        strcpy(catName,nmc);
    }
    virtual void Identify(void)
    {
        Animal::Identify();
        cout<<"I am a "<<catName<<" cat"<<endl;
    }
};
//接下来是老虎类的声明:
class Tiger:public Cat
{
private:
    char tigerName[20];
public:
    Tiger(char nmt[],char nmc[],char nma[]):Cat(nmc,nma)
    {
        strcpy(tigerName,nmt);
    }
    virtual void Identify(void)
    {
        Cat::Identify();
        cout<<"I am a "<<tigerName<<" tiger"<<endl;
    }
};
//下面这段程序中 ， 使用Announce1和Announce2两个函数表示了静态和动态关联，同时还使用了两种不同的参数传递方式。
//Announce1通过传值方式传递了一个动物类的对象。
void Announce1(Animal a)
{
//这是一个静态关联的例子
//执行动物类对象的Identify成员函数
    cout<<"In static Announce1,calling Identify:"<<endl;
    a.Identify();
    cout<<endl;
}
//Annouce2通过传地址方式传递了一个动物类的对象。
void Announce2(Animal *pa)
{
//这是一个静态关联的例子
    cout<<"In dynamic Announce1,calling Identify:"<<endl;
    pa->Identify();
    cout<<endl;
}

int main()
{
    Animal A("reptile"),*p;
    Cat C("domestic","warm blooded");
    Tiger T("bengal","wild","meat eating");
//静态关联，Announce1有一个值参数
//T是Tiger类对象，函数调用了动物类的Identity成员函数
    Announce1(T); //静态关联，调用动物类成员函数
//多态性的例子，参数是指针
//Announce2采用动态关联，去执行指针所指向的对象的Identity成员函数
    Announce2(&A); //动态关联；调用动物类成员函数
    Announce2(&C);//动态关联；调用猫科类成员函数
    Announce2(&T);//动态关联；调用老虎类成员函数
//执行动物类成员函数
    A.Identify(); //静态关联
    cout<<endl;
//动态关联；调用猫科类的成员函数
    p=&C;
    p->Identify();
    cout<<endl;
    A=T; //用老虎类对象赋值给动物类对象
    A.Identify(); //调用老虎类的Identity成员函数
    cout<<endl;
    return 0;
}
```

运行结果:


E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap05\prog0503\bin\Debu...

```
In static Announce1,calling Identify:
I am ameat eating animal

In dynamic Announce1,calling Identify:
I am areptile animal

In dynamic Announce1,calling Identify:
I am awarm blooded animal
I am a domestic cat

In dynamic Announce1,calling Identify:
I am ameat eating animal
I am a wild cat
I am a bengal tiger

I am areptile animal

I am awarm blooded animal
I am a domestic cat

I am ameat eating animal


Process returned 0 (0x0)   execution time : 0.123 s
Press any key to continue.
```

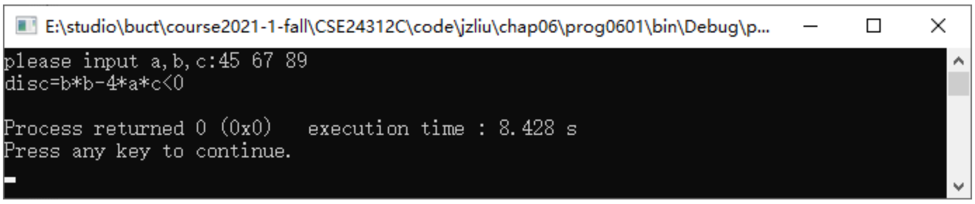
图 5.3: 运行结果

第 6 章　输入输出流

6.1　ex01p149-一元二次方程求解

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

int main( )
{
    float a,b,c,disc;
    cout<<"please input a,b,c:";
    cin>>a>>b>>c;
    if (a==0)
        cerr<<"a is equal to zero,error!"<<endl;
//将有关出错信息插入cerr流,在屏幕输出
    else if ((disc=b∗b−4∗a∗c)<0)
        cerr<<"disc=b∗b−4∗a∗c<0"<<endl;
    else
    {
        cout<<"x1="<<(−b+sqrt(disc))/(2∗a)<<endl;
        cout<<"x2="<<(−b−sqrt(disc))/(2∗a)<<endl;
    }
    return 0;
}
```

运行结果：


E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap06\prog0601\bin\Debug\p...

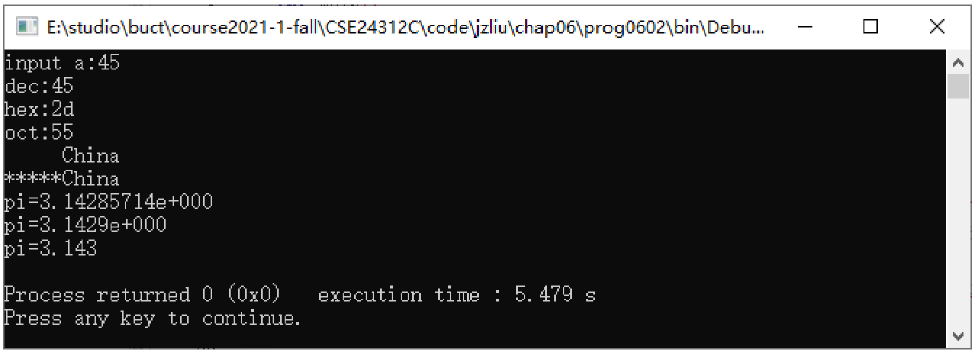
```
please input a,b,c:45 67 89
disc=b*b-4*a*c<0

Process returned 0 (0x0)   execution time : 8.428 s
Press any key to continue.
```

图 6.1: 运行结果

6.2 ex02p150-用控制符控制输出格式

```
#include <iostream>
#include <iomanip>//不要忘记包含此头文件

using namespace std;

int main()
{
    int a;
    cout<<"input a:";
    cin>>a;
    cout<<"dec:"<<dec<<a<<endl; //以十进制形式输出整数a
    cout<<"hex:"<<hex<<a<<endl; //以十六进制形式输出整数a
    cout<<"oct:"<<setbase(8)<<a<<endl; //以八进制形式输出整数a
    char ∗pt="China";//pt指向字符串"China"
    cout<<setw(10)<<pt<<endl; //指定域宽为10,输出字符串
    cout<<setfill('∗')<<setw(10)<<pt<<endl; //指定域宽10,输出字符串,空白处以'∗'填充
    double pi=22.0/7.0; //计算pi值
    cout<<setiosflags(ios::scientific)<<setprecision(8);//按指数形式输出,8位小数
    cout<<"pi="<<pi<<endl; //输出pi值
    cout<<"pi="<<setprecision(4)<<pi<<endl; //改为4位小数
    cout<<"pi="<<setiosflags(ios::fixed)<<pi<<endl;//改为小数形式输出
    return 0;
}
```

运行结果：



```
input a:45
dec:45
hex:2d
oct:55
     China
*****China
pi=3.14285714e+000
pi=3.1429e+000
pi=3.143

Process returned 0 (0x0)   execution time : 5.479 s
Press any key to continue.
```

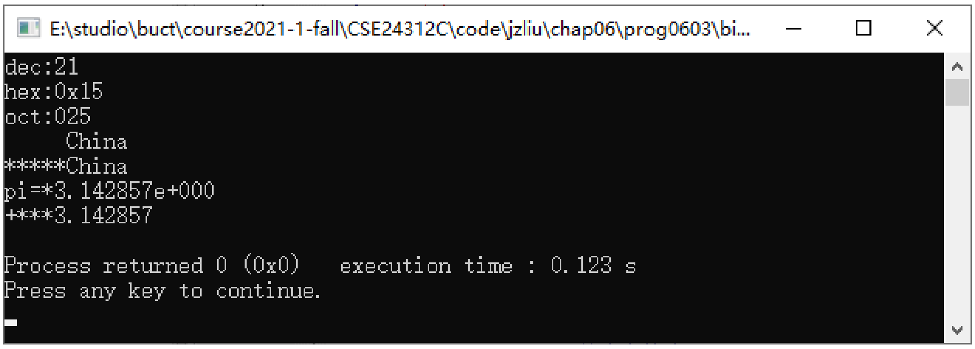
E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap06\prog0602\bin\Debu...

图 6.2: 运行结果

6.3 ex03p151-用流对象的成员函数控制输出格式

```
#include <iostream>


using namespace std;

int main( )
{
    int a=21;
    cout.setf(ios::showbase);
    cout<<"dec:"<<a<<endl; //默认以十进制形式输出a
    cout.unsetf(ios::dec); //终止十进制的格式设置
    cout.setf(ios::hex); //设置以十六进制输出的状态
    cout<<"hex:"<<a<<endl; //以十六进制形式输出a
    cout.unsetf(ios::hex);
    cout.setf(ios::oct); //设置以八进制输出的状态
    cout<<"oct:"<<a<<endl; //以八进制形式输出a
    cout.unsetf(ios::oct); //终止八进制的格式设置
    char *pt="China";//pt指向字符串"China"
    cout.width(10); //指定域宽为10
    cout<<pt<<endl; //输出字符串
    cout.width(10); //指定域宽为10
    cout.fill('*'); //指定空白处以'*'填充
    cout<<pt<<endl; //输出字符串
    double pi=22.0/7.0; //定义pi并赋初始值
    cout.setf(ios::scientific); //指定用科学记数法输出
    cout<<"pi="; //输出"pi="
    cout.width(14); //指定域宽
    cout<<pi<<endl; //输出pi值
    cout.unsetf(ios::scientific); //终止科学记数法状态
    cout.setf(ios::fixed); //指定用定点形式输出
    cout.width(12); //指定域宽为12
    cout.setf(ios::showpos); //正数输出"+"号
    cout.setf(ios::internal); //数符出现在左侧
    cout.precision(6); //保留6位小数
    cout<<pi<<endl; //输出pi,注意数符"+"的位置
    return 0;
}
```

运行结果:



E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap06\prog0603\bi...

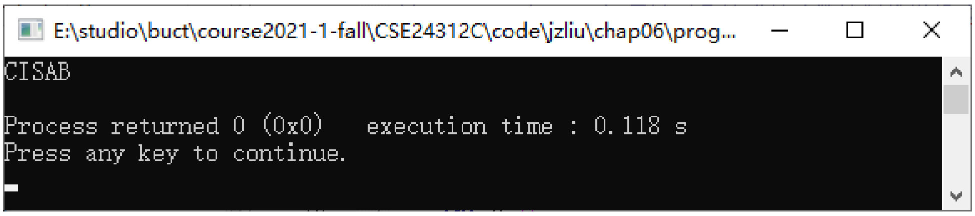
```
dec:21
hex:0x15
oct:025
         China
*****China
pi=*3.142857e+000
+***3.142857

Process returned 0 (0x0)   execution time : 0.123 s
Press any key to continue.
```

图 6.3: 运行结果

6.4 ex04p153-将字符串反向输出

```
#include <iostream>

using namespace std;

int main( )
{
    char *a="BASIC";//字符指针指向'B'
    for(int i=4; i>=0; i--)
        cout.put(*(a+i));
    cout.put('\n');
    return 0;
}
```

运行结果:



E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap06\prog...

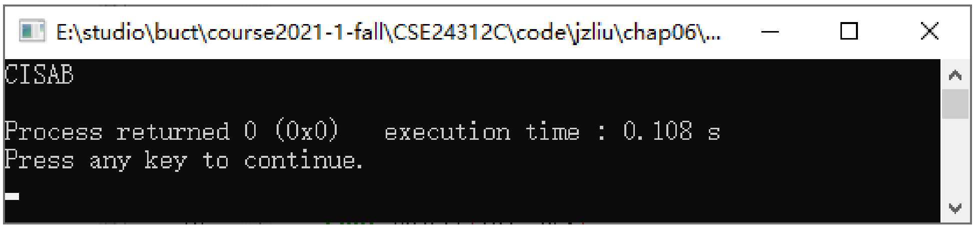
```
CISAB

Process returned 0 (0x0)   execution time : 0.118 s
Press any key to continue.
```

图 6.4: 运行结果

6.5 ex05p153-用 putchar 函数将字符串反向输出

```
#include <stdio.h>
#include <iostream>

using namespace std;

int main( )
{
    char *a="BASIC";//字符指针指向'B'
    for(int i=4; i>=0; i--)
        putchar(*(a+i));
    putchar('\n');
    return 0;
}
```

运行结果:



```
CISAB

Process returned 0 (0x0)   execution time : 0.108 s
Press any key to continue.
```

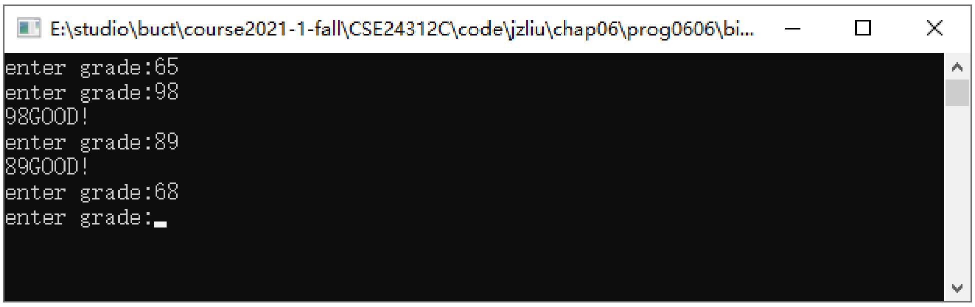
E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap06\...

图 6.5: 运行结果

6.6 ex06p153-通过测试 cin 的真假值，判断流对象是否处于正常状态

```
#include <iostream>

using namespace std;

int main( )
{
    float grade;
    cout<<"enter grade:";
    while(cin>>grade)//能从cin流读取数据
    {
        if(grade>=85)
            cout<<grade<<"GOOD!"<<endl;
        if(grade<60)
            cout<<grade<<"fail!"<<endl;
        cout<<"enter grade:";
    }
    cout<<"The end."<<endl;
    return 0;
}
```

运行结果：


E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap06\prog0606\bi...

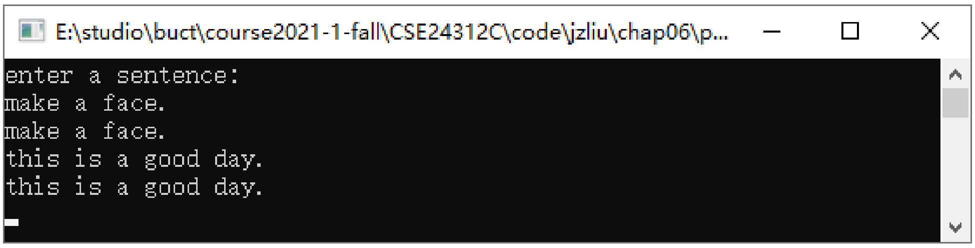
```
enter grade:65
enter grade:98
98GOOD!
enter grade:89
89GOOD!
enter grade:68
enter grade:_
```

图 6.6: 运行结果

6.7 ex07p155-用 get 函数读入字符

```
#include <stdio.h>
#include <iostream>

using namespace std;

int main( )
{
   int c;
   cout<<"enter a sentence:"<<endl;
   while((c=cin.get())!=EOF)
      cout.put(c);
   return 0;
}
```

运行结果：


E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap06\p...

```
enter a sentence:
make a face.
make a face.
this is a good day.
this is a good day.
_
```

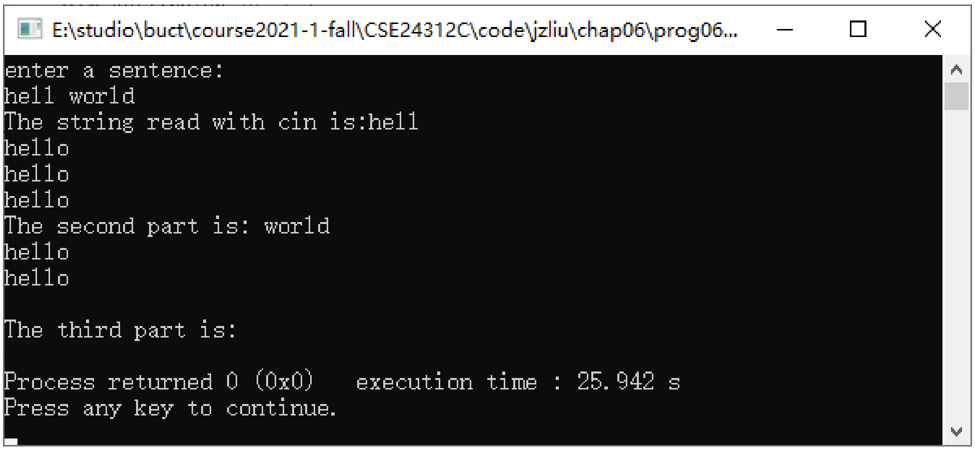
图 6.7: 运行结果

6.8 ex08p156-用 getline 函数读入一行字符

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

int main( )
```

```
{
    char ch[20];
    cout<<"enter a sentence:"<<endl;
    cin>>ch;
    cout<<"The string read with cin is:"<<ch<<endl;
    cin.getline(ch,20,'/');//¶Á¸öÖ·û»òÓö'/'½áÊø
    cout<<"The second part is:"<<ch<<endl;
    cin.getline(ch,20);
    cout<<"The third part is:"<<ch<<endl;
    return 0;
}
```

运行结果：



E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap06\prog06...

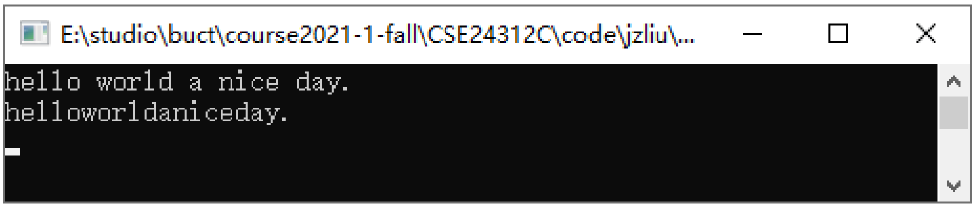
```
enter a sentence:
hell world
The string read with cin is:hell
hello
hello
hello
The second part is: world
hello
hello

The third part is:

Process returned 0 (0x0)   execution time : 25.942 s
Press any key to continue.
```

图 6.8: 运行结果

6.9 ex09p157-逐个读入一行字符，将其中的非空格字符输出

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

int main( )
{
    char c;
    while(!cin.eof( )) //eof( )为假表示未遇到文件结束符
        if((c=cin.get( ))!=' ')
            cout.put(c);
    return 0;
}
```

运行结果：



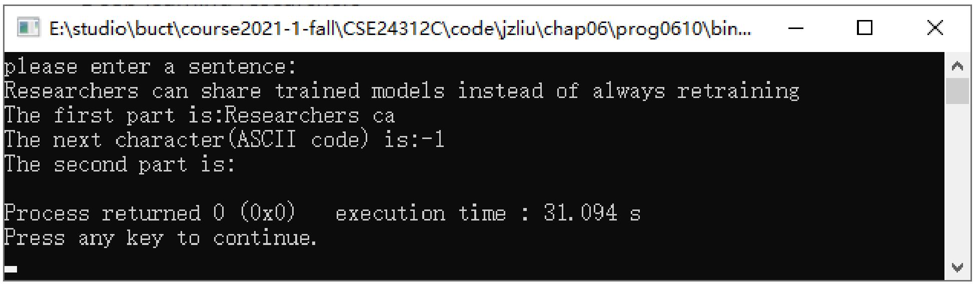
```
hello world a nice day.
helloworldaniceday.
```

图 6.9: 运行结果

6.10 ex10p157-peek 函数和 putback 函数的用法

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

int main( )
{
   char c[20];
   int ch;
   cout<<"please enter a sentence:"<<endl;
   cin.getline(c,15,'/');
   cout<<"The first part is:"<<c<<endl;
   ch=cin.peek( );//观看当前字符
   cout<<"The next character(ASCII code) is:"<<ch<<endl;
   cin.putback(c[0]);
//将'I'插入到指针所指处
   cin.getline(c,15,'/');
   cout<<"The second part is:"<<c<<endl;
   return 0;
}
```

运行结果：



E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap06\prog0610\bin...

```
please enter a sentence:
Researchers can share trained models instead of always retraining
The first part is:Researchers ca
The next character(ASCII code) is:-1
The second part is:

Process returned 0 (0x0)   execution time : 31.094 s
Press any key to continue.
```

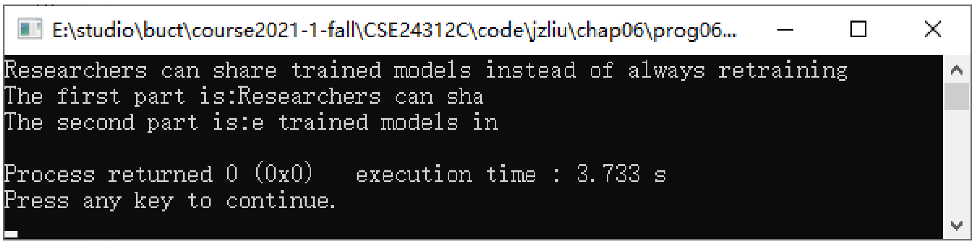
图 6.10: 运行结果

6.11 ex11p158-用 ignore 函数跳过输入流中的字符

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

```
int main( )
{
   char ch[20];
   cin.get(ch,20,'/');
   cout<<"The first part is:"<<ch<<endl;
   cin.ignore(); // 跳过输入流中的一个字符
   cin.get(ch,20,'/');
   cout<<"The second part is:"<<ch<<endl;
   return 0;
}
```

运行结果：


E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap06\prog06...

```
Researchers can share trained models instead of always retraining
The first part is:Researchers can sha
The second part is:e trained models in

Process returned 0 (0x0)   execution time : 3.733 s
Press any key to continue.
```

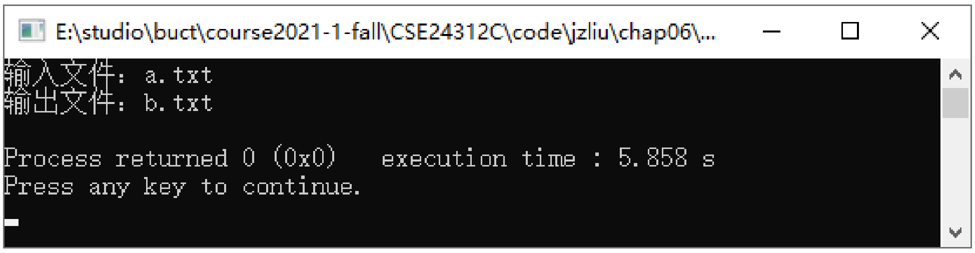
图 6.11: 运行结果

6.12 ex12p161-应用实例 (拷贝文件)

```
#include <iostream>
#include <fstream>
#include "stdlib.h"

using namespace std;

void CopyFile(const char *from, const char *to)
{
   cin.clear();
   ifstream inf(from);
   if(!inf)
   {
      cerr<<"输入文件打开失败！"<<endl;
      exit(1);
   }
   ofstream outf(to,ios_base::trunc);
   if(!outf)
   {
      cerr<<"输出文件打开失败！"<<endl;
      exit(3);
   }
   while(!inf.eof())
      outf.put(inf.get());
```

```
    inf.close();
    outf.close();
}
int main()
{
    char fin[30],fout[30];
    cout<<"输入文件: ";
    cin>>fin;
    cout<<"输出文件: ";
    cin>>fout;
    CopyFile(fin,fout);
    return 0;
}
```

运行结果:


E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap06\...

```
输入文件: a.txt
输出文件: b.txt

Process returned 0 (0x0)   execution time : 5.858 s
Press any key to continue.
```

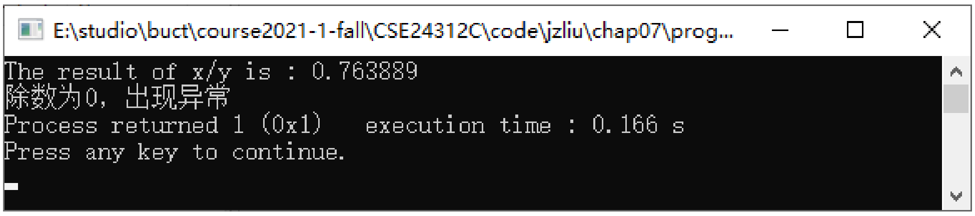
图 6.12: 运行结果

第 7 章　容错与异常处理

7.1　ex01p164-实现当除数为 0 时，停止运行并给出错误信息

```
#include <iostream>
#include <stdlib.h>

using namespace std;

double div(double x, double y) //定义函数
{
    if(y==0) //除数为0
    {
        cout<<"除数为0, 出现异常"; //输出错误信息
        exit(1); //异常退出程序
    }
    return x/y; //返回两个数的商
}

int main()
{
    double result;
    result=div(5.5, 7.2); // 调用函数
    cout<<"The result of x/y is : "<<result<<endl; //输出正确结果
    result=div(8.5,(double)0); //除数为0发生异常
    cout<<"The result of x/y is : "<<result<<endl; //不执行该语句
    return 0;
}
```

运行结果：


E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap07\prog...

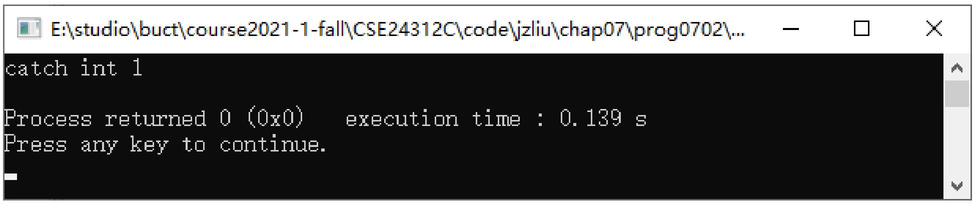
```
The result of x/y is : 0.763889
除数为0，出现异常
Process returned 1 (0x1)   execution time : 0.166 s
Press any key to continue.
```

图 7.1: 运行结果

7.2 ex02p166-throw, try,catch 语句的简单用法

```
#include<iostream>
using namespace std;
int main()
{
    try
    {
        throw 1;
    }
    catch(int id)
    {
        cout<<"catch int "<<id<<endl ;
    }
    return 0 ;
}
```

运行结果：


E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap07\prog0702\...

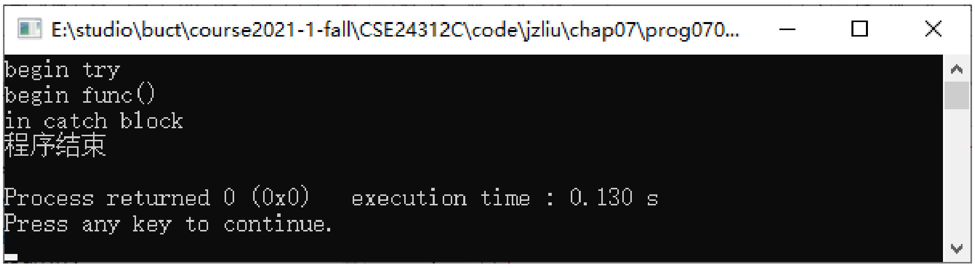
```
catch int 1

Process returned 0 (0x0)   execution time : 0.139 s
Press any key to continue.
```

图 7.2: 运行结果

7.3 ex03p164-throw 异常必须在 try 语句块中执行

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

void func()
{
   cout<<"begin func()"<<endl ;
   throw 1;//抛出一个异常
   cout<<"end func()"<<endl ;
}
int main()
{
   try
   {
      cout<<"begin try"<<endl ;
      func() ;
      cout<<"end try"<<endl ;
   }
   catch(int) //注意, catch 的类型和 throw 的类型一致
   {
      cout<<"in catch block"<<endl ;
   }
   cout<< "程序结束"<<endl;
   return 0;
}
```

运行结果:


E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap07\prog070...

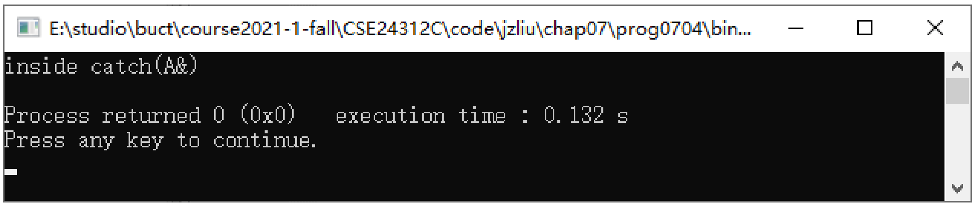
```
begin try
begin func()
in catch block
程序结束

Process returned 0 (0x0)   execution time : 0.130 s
Press any key to continue.
```

图 7.3: 运行结果

7.4 ex04p167-类对象的异常匹配处理

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


class A {};
class B
{
public:
   B(const A&) {} //以A来拷贝构造函数B
};
void f()
{
   throw A(); //抛出A类型的异常，即使没有构造，但是可以使用
}
//系统自己生成的默认的构造函数。
int main()
{
   try
   {
      f();
   }
   catch(B&) //强调过了catch语句最好用的是引用传递而不是值传递。
   {
      cout<<"inside catch(B&)"<<endl;
   }
   catch(A&)
   {
      cout<<"inside catch(A&)"<<endl;
   }
   return 0;
}
```

运行结果：



E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap07\prog0704\bin...

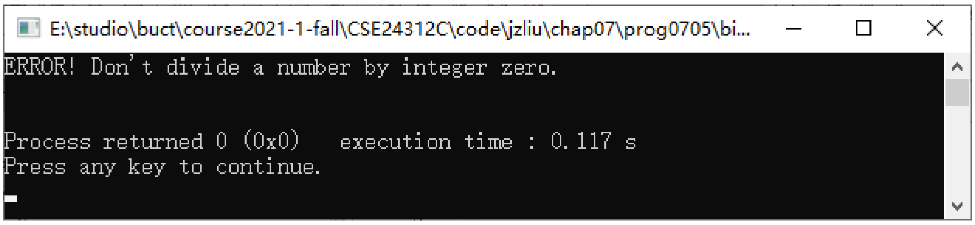
```
inside catch(A&)

Process returned 0 (0x0)   execution time : 0.132 s
Press any key to continue.
```

图 7.4: 运行结果

7.5 ex05p169-编写一个自己的异常处理【对原程序进行了改写】

```
#include<exception>
#include<iostream>

using namespace std;

class MyException:public exception
{
public:
   const char* what()const throw()
   {
      return "ERROR! Don't divide a number by integer zero.\n";
   }
};

int main()
{
    try
    {
        throw MyException ();
    }
    catch(MyException &e)
    {
        cout<<e.what()<<endl;
    }
    return 0;
}
```

运行结果：



```
ERROR! Don't divide a number by integer zero.

Process returned 0 (0x0)   execution time : 0.117 s
Press any key to continue.
```

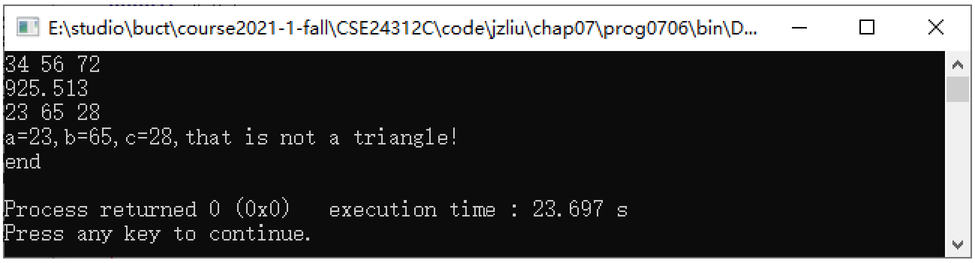
E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap07\prog0705\bi...

图 7.5: 运行结果

7.6 ex06p170-给出三角形的三边 a,b,c，求三角形的面积

```
#include <iostream>
#include <cmath>

using namespace std;

int main( )
{
    double triangle(double,double,double);
    double a,b,c;
    cin>>a>>b>>c;
    try//在try块中包含要检查的函数
    {
        while(a>0 && b>0 && c>0)
        {
            cout<<triangle(a,b,c)<<endl;
            cin>>a>>b>>c;
        }
    }
    catch(double) //用catch捕捉异常信息并作相应处理
    {
        cout<<"a="<<a<<",b="<<b<<",c="<<c<<",that is not a triangle!"<<endl;
    }
    cout<<"end"<<endl;
}

double triangle(double a,double b,double c) //计算三角形的面积的函数
{
    double s=(a+b+c)/2;
    if (a+b<=c||b+c<=a||c+a<=b) throw a; //当不符合三角形条件抛出异常信息
    return sqrt(s∗(s−a)∗(s−b)∗(s−c));
    return 0;
}
```

运行结果：



```
34 56 72
925.513
23 65 28
a=23,b=65,c=28,that is not a triangle!
end

Process returned 0 (0x0)   execution time : 23.697 s
Press any key to continue.
```

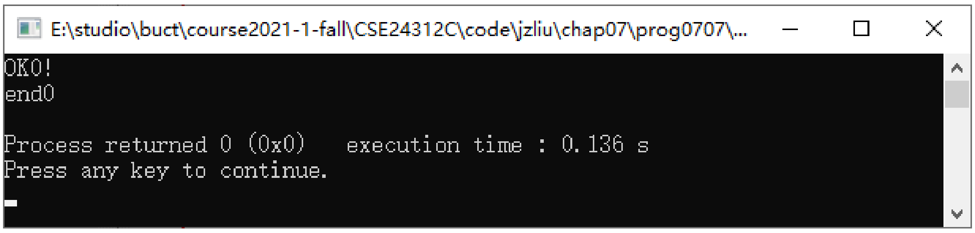
E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap07\prog0706\bin\D...

图 7.6: 运行结果

7.7 ex07p.171-在函数嵌套情况下检测异常处理

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

int main( )
{
    void f1( );
    try
    {
        f1( ); //调用f1( )
    }
    catch(double)
    {
        cout<<"OK0!"<<endl;
    }
    cout<<"end0"<<endl;
    return 0;
}
void f1( )
{
    void f2( );
    try
    {
        f2( ); //调用f2( )
    }
    catch(char)
    {
        cout<<"OK1!";
    }
    cout<<"end1"<<endl;
}

void f2( )
{
    void f3( );
    try
    {
        f3( ); //调用f3( )
    }
    catch(int)
    {
        cout<<"Ok2!"<<endl;
    }
    cout<<"end2"<<endl;
}
void f3( )
{
    double a=0;
    try
    {
        throw a; //抛出double类型异常信息
    }
    catch(float)
    {
        cout<<"OK3!"<<endl;
    }
    cout<<"end3"<<endl;
}
```

运行结果：



E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap07\prog0707\...

```
OK0!
end0

Process returned 0 (0x0)   execution time : 0.136 s
Press any key to continue.
```

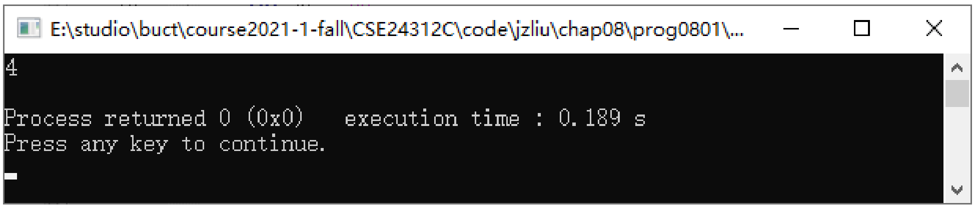
图 7.7: 运行结果

第 8 章　模版

8.1　ex01p177-求任意两个具有相同类型的数中较小值的函数模板

```
#include <iostream>

using namespace std;

template <class T>
T minvalue(T x, T y)
{
    return x>y?y:x;
}

int main() {
    int a1 = 4;
    int a2 = 56;
    cout << minvalue(a1, a2) << endl;
    return 0;
}
```

运行结果：


E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap08\prog0801\...

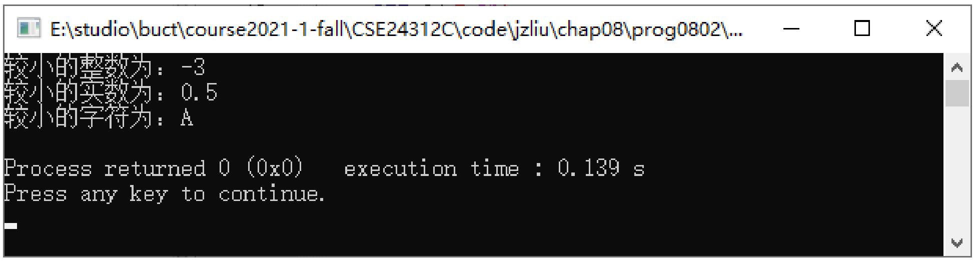
```
4

Process returned 0 (0x0)   execution time : 0.189 s
Press any key to continue.
```

图 8.1: 运行结果

8.2　ex02p177-函数模板的使用

```
#include <iostream>

using namespace std;

template <class T>
T minval(T x, T y)
{
    return x>y?y:x;
}
int main()
{
    int a1=-3,b1=1;
    double a2=0.5, b2=2.3;
    char a3='A', b3='d';
    cout<<"较小的整数为："<<minval(a1,b1)<<endl;
    cout<<"较小的实数为："<<minval(a2,b2)<<endl;
    cout<<"较小的字符为："<<minval(a3,b3)<<endl;
    return 0;
}
```

运行结果：



E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap08\prog0802\...

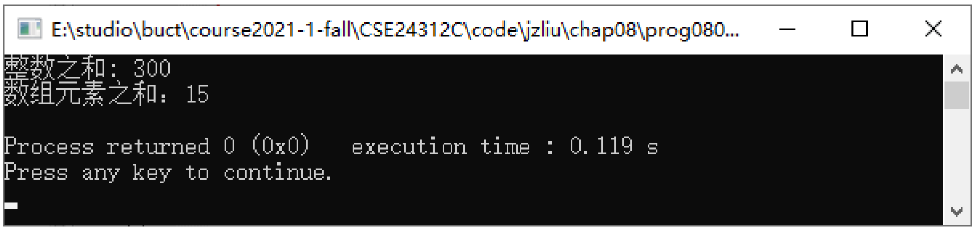
```
较小的整数为：-3
较小的实数为：0.5
较小的字符为：A

Process returned 0 (0x0)   execution time : 0.139 s
Press any key to continue.
```

图 8.2: 运行结果

8.3 ex03p178-重载函数模板

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

template <class T> //定义一个模板类型
T sum(T a, T b) //定义一个重载的函数模板
{
    return a + b; //返回两个数之和
}
template <class T> //定义一个模板类型
T sum(T arr[], int nLen) //定义另一个重载的函数模板
{
    T temp = 0; //定义一个变量
    for(int i=0; i<nLen; i++) //利用循环累计求和
    {
        temp += arr[i];
    }
    return temp; //返回结果
}

int main( )
{
    int s1 = sum(100, 200); //调用第一个重载的函数模板，实现两个数的求和运算
    cout << "整数之和：" << s1 << endl; //输出结果
    int a[5]= {1, 2, 3, 4, 5};
    int s2 = sum(a, 5); //调用第2个重载的函数模板，实现数组元素的求和运算
    cout << "数组元素之和：" << s2 << endl;
    return 0;
}
```

运行结果：


E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap08\prog080...

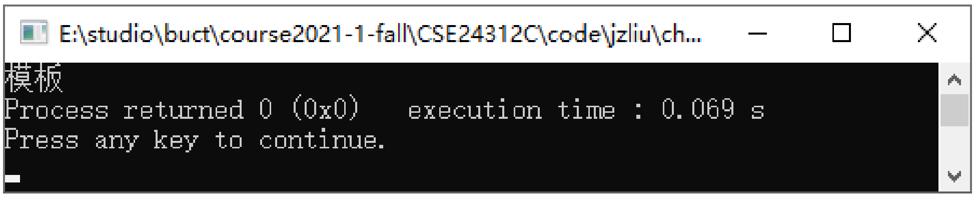
```
整数之和: 300
数组元素之和: 15

Process returned 0 (0x0)   execution time : 0.119 s
Press any key to continue.
```

图 8.3: 运行结果

8.4 ex04p181-类模板的例子

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

template <class T>//带参数T的模板声明
class A
{
private:
    T x;//类型为T的私有数据对象
    int y;//类型为int的私有数据对象
public:
    A(T a, int b)
    {
        x=a; //类构造函数
        y=b;
    }
    T getx()
    {
        return x; //类成员函数，返回类型为T
    }
    int gety()
    {
        return y;
    }
};
```

运行结果：



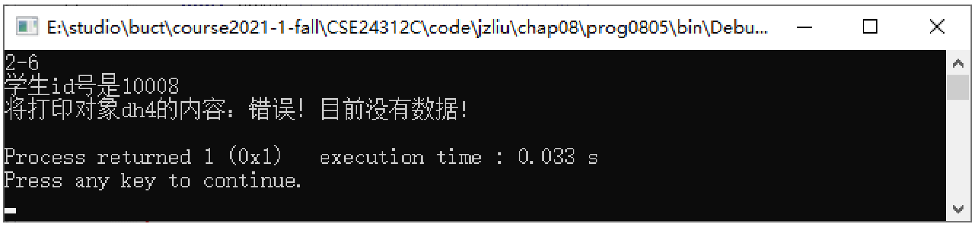
```
模板
Process returned 0 (0x0)   execution time : 0.069 s
Press any key to continue.
```

图 8.4: 运行结果

8.5 ex05p182-声明存储任意数据类型的类模板 Datahouse，然后用具体的数据类型参数对类模板进行实例化

```
#include<iostream>
#include<cstdlib>
using namespace std;
struct Student
{
    int id; //学号
    float avg; //平均分
};

template <class T> //类模板；实现对任意数据类型的存取
class Datahouse
{
private:
    T item; //item用于存放任意类型的数据
    bool saved;//saved标记item是否已被存放入
public:
    Datahouse(void);//默认形式（无形参）的构造函数
    T getitem(void);//提取数据函数
    void putitem(T x);//存入数据函数
};

//模板类成员函数的实现
template<class T>//默认形式构造函数的实现
Datahouse<T>::Datahouse(void):saved(false) {}
template<class T>
T Datahouse<T>::getitem(void)
{
    if(!saved)//如果试图提取未初始化的数据，则终止程序
    {
        cout<<"错误！目前没有数据！"<<endl;
        exit(1); //使程序完全退出，返回到操作系统
    }
    return item; //返回item 中存放的数据
}

template<class T>
void Datahouse<T>::putitem(T x)
{
    saved=true;//将saved值为真
    item=x;//赋值
}

int main()
{
    Student stu1= {10008, 89}; //声明结构体的对象并初始化
    Datahouse<int> dh1,dh2;
    Datahouse<Student>dh3;
    Datahouse<double>dh4;
    dh1.putitem(2);
    dh2.putitem(-6);
    cout<<dh1.getitem()<<""<<dh2.getitem()<<endl;//输出dh1,dh2的值
    dh3.putitem(stu1);//向对象s3中存入数据
    cout<<"学生id号是"<<dh3.getitem().id<<endl;//输出对象dh3的数据成员
    cout<<"将打印对象dh4的内容：";
    cout<<dh4.getitem()<<endl;//输出对象dh4的数据成员，由于dh4无数据，导致程序终止
    return 0;
}
```

运行结果：



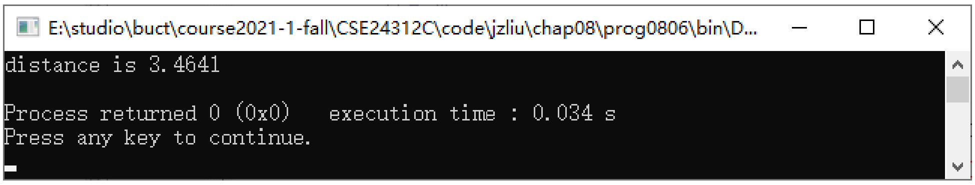
E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap08\prog0805\bin\Debu...

```
2-6
学生id号是10008
将打印对象dh4的内容：错误！目前没有数据！

Process returned 1 (0x1)   execution time : 0.033 s
Press any key to continue.
```

图 8.5: 运行结果

8.6 ex06p185-实现一个三维坐标的点类模板，并定义一个友元函数来判断两个点是否重合

运行结果：



E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap08\prog0806\bin\D...

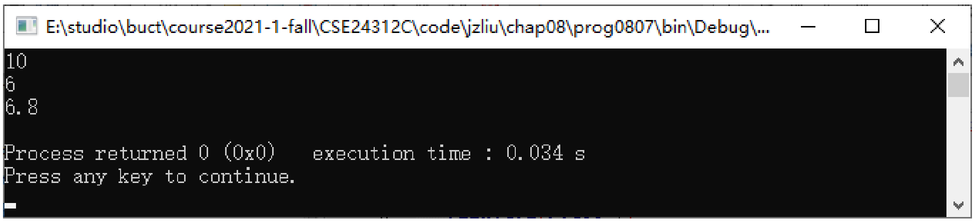
```
distance is 3.4641

Process returned 0 (0x0)   execution time : 0.034 s
Press any key to continue.
```

图 8.6: 运行结果

8.7 ex07p186-模板类继承非模板类例子

```
#include <iostream>
using namespace std;
class Base//非模板类
{
    int data;
public:
    Base (int n):data(n) {}
    int getData() const
    {
        return data;
    }
};
template <class T>//类模板
class Derived:public Base //继承一个普通类
{
    T value;
public:
    Derived(int n, T m):Base(n), value(m) {}
    T sum() const
    {
        return value+(T)getData();
    }
};
int main()
{
    Base a(10);
    cout<<a.getData()<<endl;
    Derived<int>b(2,4);
    cout<<b.sum()<<endl;
    Derived<double>c(2, 4.8);
    cout<<c.sum()<<endl;
    return 0;
}
```

运行结果:


E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap08\prog0807\bin\Debug\...

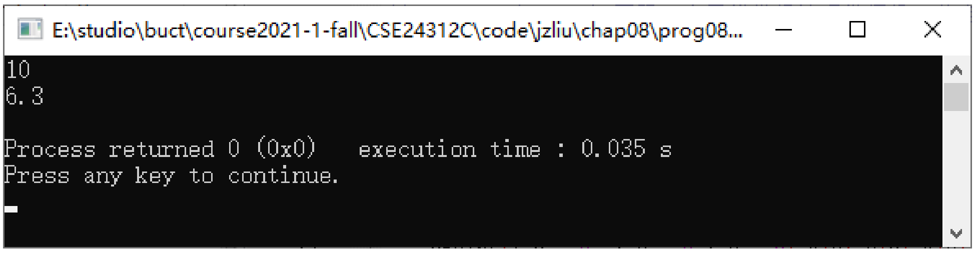
```
10
6
6.8

Process returned 0 (0x0)   execution time : 0.034 s
Press any key to continue.
```

图 8.7: 运行结果

8.8 ex08p186-非模板类继承模板类例子

```
#include <iostream>
using namespace std;
template <class T>
class Base
{
    T data;
public:
    Base(T n):data(n) {}
    T getData() const
    {
        return data;
    }
};
class Derived: public Base<int>//继承基类模板的一个实例
{
    double value;
public:
    Derived(int n, double m): Base<int>(n),value(m) {}
    double sum() const
    {
        return value+(double)getData();
    }
};
int main()
{
    Base<int> a(10);
    cout<<a.getData()<<endl;
    Derived b(2,4.3);
    cout<<b.sum()<<endl;
    return 0;
}
```

运行结果:



E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap08\prog08...

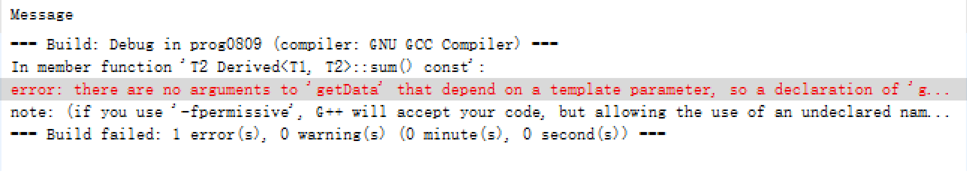
```
10
6.3

Process returned 0 (0x0)   execution time : 0.035 s
Press any key to continue.
```

图 8.8: 运行结果

8.9 ex09p187-从类模板派生一个类模板的例子【程序有问题，GCC 不能编译】

```
#include <iostream>
using namespace std;
template <class T>
class Base
{
    T data;
public:
    Base(T n):data(n) {}
    T getData() const
    {
        return data;
    }
};

template <class T1, class T2>
class Derived: public Base<T1>
{
    T2 value;
public:
    Derived(T1 n, T2 m):Base<T1>(n),value(m) {}
    T2 sum() const
    {
        return value+(T2)getData();
    }
};
int main()
{
    Base<int> a(10);
    cout<<a.getData()<<endl;
    Derived<int,double> b(2,4.3);
    cout<<b.sum()<<endl;
    Derived<double,double> c(2.8,4.3);
    cout<<c.sum()<<endl;
    return 0;
}
```

运行结果:



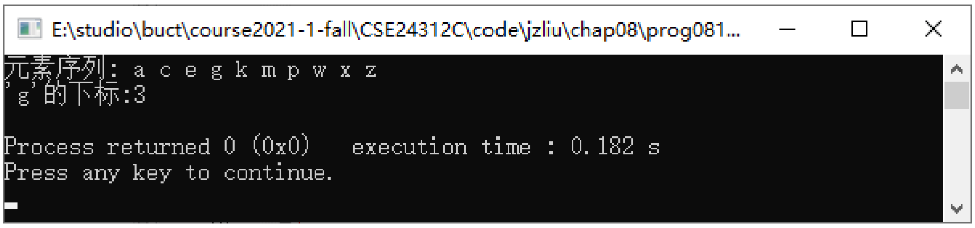
```
Message
--- Build: Debug in prog0809 (compiler: GNU GCC Compiler) ---
In member function 'T2 Derived<T1, T2>::sum() const':
error: there are no arguments to 'getData' that depend on a template parameter, so a declaration of 'g...
note: (if you use '-fpermissive', G++ will accept your code, but allowing the use of an undeclared nam...
--- Build failed: 1 error(s), 0 warning(s) (0 minute(s), 0 second(s)) ---
```

图 8.9: 运行结果

8.10 ex10p188-设计类模板 Sample，用于对一个有序数组进行二分法查找元素

```
#include <iostream>
using namespace std;
#define Max 100
template <class T>
class Sample
{
    T A[Max];
    int n;
public:
    Sample() {}
    Sample(T a[],int i);
    int seek(T c);
    void disp()
    {
        for(int i=0; i<n; i++)
            cout<<A[i]<<" ";
        cout<<endl;
    }
};
template <class T>
Sample<T>::Sample(T a[],int i)
{
    n=i;
    for(int j=0; j<i; j++)
        A[j]=a[j];
}
template <class T>
int Sample<T>::seek(T c)
{
    int low=0,high=n−1,mid;
    while(low<=high)
    {
        mid=(low+high)/2;
        if(A[mid]==c)
            return mid;
        else if(A[mid]<c) low=mid+1;
        else high=mid−1;
    }
    return −1;
}
int main()
{
    char a[]="acegkmpwxz";
    Sample<char>s(a,10);
    cout<<"元素序列: ";
    s.disp();
    cout<<"'g'的下标:"<<s.seek('g')<<endl;
    return 0;
}
```

运行结果：



E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap08\prog081...

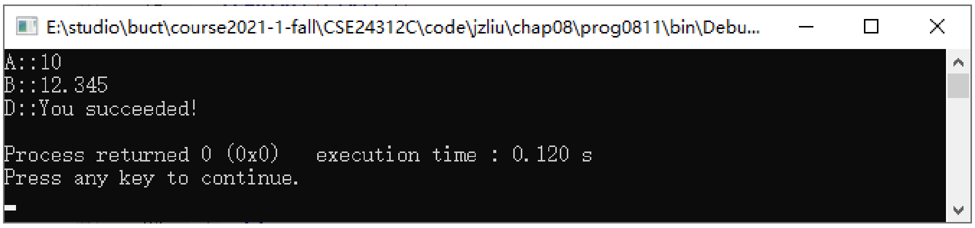
```
元素序列: a c e g k m p w x z
'g'的下标:3

Process returned 0 (0x0)   execution time : 0.182 s
Press any key to continue.
```

图 8.10: 运行结果

8.11 ex11p189-类模板的多继承

```
#include <iostream>
using namespace std;
template<class T>
class A
{
public:
   A(T a)
   {
      cout << "A::"<< a << endl;
   }
};
template<class T>
class B
{
public:
   B(T b)
   {
      cout << "B::" << b << endl;
   }
};
template<class X, class Y, class Z>
class D:public A<Y>, public B<Z>
{
public:
   D(X a, Y b, Z c) : A<Y> (b), B<Z> (c)
   {
      cout << "D::" << a << endl;
   }
};
int main()
{
   D< char*, int, double > obj1 ("You succeeded!", 10, 12.345);
   return 0;
}
```

运行结果:



E:\studio\buct\course2021-1-fall\CSE24312C\code\jzliu\chap08\prog0811\bin\Debu...

```
A::10
B::12.345
D::You succeeded!

Process returned 0 (0x0)   execution time : 0.120 s
Press any key to continue.
```

图 8.11: 运行结果

参考文献

[1] 刘建舟、徐承志等. *C++ 面向对象程序设计*. 机械工业出版社,http://www.hzbook.com, 2012.

[2] Author. *Title*. http://www.baidu.com, 2020.