重庆大学 计算机学院硕士研究生复试题

C++程序设计语言

2007 年 4 月

考试方式：○ 开卷 ◉ 闭卷 ○ 其他　　考试时间：120 分钟

题 号	一	二	三	四	五	六	七	八	九	十	总 分
得 分											

一、 简答题（1 分/每小题，共 10 分）

1. 结构化程序设计所规定的三种基本控制结构是：(C)
   A)输入、处理、输出　　B)树形、网形、环形
   C)顺序、选择、循环　　D)主程序、子程序、函数

2. 考虑函数原型 void test(int a, int b=7, char* c="*")。下面的函数调用中，属于不合法调用的是：(C)
   A test(5);　　B test(5,8);　　C test(6,"#");　　D test(0,0,"*");

3. 下面有关重载函数的说法中正确的是：(C)
   A 重载函数必须具有不同的返回值类型；　　B 重载函数形参个数必须不同；
   C 重载函数必须有不同的形参列表；　　D 重载函数名可以不同。

4. 下面关于静态成员的描述中，正确的是：(A)
   A 静态数据成员是类的所有对象共享的数据
   B 类的每个对象都有自己的静态数据成员
   C 类的不同对象有不同的静态数据成员值
   D 静态数据成员不能通过类的对象访问

5. 下列关于构造函数的描述中，错误的是：(D)
   A 构造函数可以设置默认参数；　　B 构造函数在定义类对象时自动执行；
   C 构造函数可以是内联函数；　　D 构造函数不可以重载

6. 实现运行时的多态性用(D)
   A 重载函数　　B 构造函数　　C 析构函数　　D 虚函数

7. 在 C++中，数据封装要解决的问题是：(D)
   A 数据规范化排列　　B 数据高速转换
   C 避免数据丢失　　D 切断了不同模块之间的数据的非法使用

8. 在标准模板库中，提供了访问容器和序列中每个元素的方法的是(A)
   A 容器　　B 适配器　　C 迭代器　　D 算法

9. 一个普通函数或者类的成员函数可以访问封装于另外一个类中的数据，通常我们采用 (D)
   A 虚函数　　B 函数重载　　C 继承　　D 友元关系

10. C++程序中，如果在某段程序中发现了自己不能处理的异常，就可以使用(B)表达式抛掷这个异常，将它抛给调用者。
   A try　　B throw　　C catch　　D exception

二、 填空题（2 分/每空，共 20 分）

1. 由 char const *str="stucture";所以定义的指针称为 常量指针。

2. 对某个运算符的重载，实际上是用关键字 operator 与该运算符组成一个运算符函数。

3. 在面向对象程序设计中，不同的对象可以调用相同名称的函数并导致完全不同的行为的现象称为 多态。

4. 在用 class 定义一个类时，数据成员和成员函数的默认访问权限是 private。

5. 已知 int DBL(int n){return n + n;}和 long DBL(long n){return n + n;}是一个函数模板的两个实例，则该函数模板的定义是 template<class T> T DBL(T n)

6. 一个类中可以有 多 个构造函数，可以有 1 个析构函数。

7. 在 C++程序中，派生新类的过程包括三个步骤：包含基类头文件、指定是 public、private或protected继承、添加新的成员

三、 综合分析题（12 分/每小题，共 30 分）

1. 有哪些方法可以让函数返回多个值？请举例说明

向函数传递多个数据的指针或引用
向函数传递 vo. 数组名
使用全局变量，在函数中改变全局变量的值

函数不指定返回值类型时默认为 int

2. 写出下列程序的运行结果。

```
#include <iostream>
using namespace std;
class A{
   public:
      A (){
         cout<<" A created"<<endl;
      }
      ~A (){
         cout<<" A destroyed"<<endl;
      }
};
class member {
   public:
      member(int n){
         this->n = n;
         number++;
         cout<<"member "<<n<<" created;number:"<< number <<endl;
      }
      ~member(){
         cout<<"member "<<n<<" destroyed;number:"<< number <<endl;
         number--;
      }
    private:
      int n;
      static int number;
};
class B: public A{
   public:
      B():m2(2),m1(1){
         cout<<"B created"<<endl;
      }
      ~B(){
         cout<<"B destroyed"<<endl;
      }
   private:
      member m1;
      member m2;
};
int member::number = 0;
void main(){
   B b;
   cout<<"------"<<endl;
}
```

A created
member1 created; number:1
member2 created; number:2
B created
------ 换行
B destroyed
member 2 destroyed; number:2
member 1 destroyed; number:1
A destroyed

3、请指出下面实现成员函数的代码中的一处错误并说明错误原因和改正方法：

```
class Fract
{
   private:
      int x,y;
      //其他代码
};
//++运算符的重载
Fract& Fract::operator++(int)
{
   x++; y++;
   return *this;
}
```

public: Fract &operator++(int);

前置单目运算符重载.

在基中没有声明运算符重载函数.

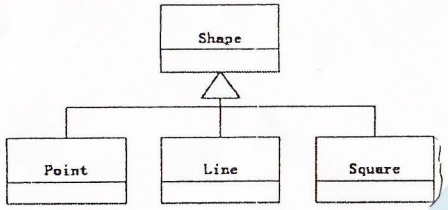
4、阅读下面的程序，在空格处填上合适的语句

某绘图系统存在 Point、Line、Square 三种图元，它们具有 Shape 接口，图元的类图关系如图 1 所示。现要将 Circle 图元加入此绘图系统以实现功能扩充。已知某第三方类库已经提供了 XCircle，但它不是由 Shape 派生而来，它提供的接口不被系统直接使用。代码 1 既使用了 XCircle 又遵循了 Shape 规定的接口，即避免了从头开发一个新的 Circle 类，又可以不修改绘图系统中已经定义的接口。代码 2 根据用户指定的参数生成特定的图元实例，并对之进行显示操作。

绘图系统定义的接口与 XCircle 提供的显示接口及其功能如下表所示：

Shape	XCircle	功能
display()	DisplayIt()	显示图元

[图 1]



[代码 1]

```
class Circle:public   (a)   {
    private:
        (b)   m_circle;
    public:
        void display () {
            (c)   ;
        }
};
```

[代码 2]

```
class Factory{
    public:
    (d)   getShapeInstance(int type){ //生成特定类实例
            switch(type){
                case 0:return new Point;
                case 1:return new Rectangle;
                case 2:return new Line;
                case 3: return new Circle;
                default: return NULL;
            }
        }
};

void main(int argc,char *argv[]){
    if(argc!=2){
        cout<<"error parameters ! "<<endl;
        return;
    }

    int type=atoi(argv[1]);
    Factory factory;
    Shape *s;
    s=factory.   (e)   ;
    if(s==NULL){
        cout<<"Error get the instance!"<<endl;
        return;
    }
    s->display();
       (f)   ;
    return;
}
```

四、 写程序（12 分/每小题，共 36 分）

1. 定义一个处理日期的类 TDate，它有 3 个私有数据成员：Month, Day, Year 和若干个公有成员函数，并实现如下要求：①构造函数重载；②成员函数设置缺省参数；③定义一个友元函数来打印日期；④定义一个非静态成员函数设置日期；⑤可使用不同的构造函数来创建不同的对象。

2. 请定义一个堆栈类模板，并写出测试程序。

```
#ifndef STACK_CLASS
#define STACK_CLASS
#include <iostream>
#include <cstdlib>
using namespace std;
const int MaxStackSize=50;
template < class T >
class Stack
{
 private:
   T stacklist[MaxStackSize];
   int top;
 public:
   Stack ( void);
   void Push (const T& item);
   T Pop (void);
   void ClearStack (void);
   T Peek (void) const;
   int StackEmpty (void) const;
   int StackFull (void) const;
};
```

```
template < class T >
Stack <T>::Stack (void):top(-1)
{ }
template < class T >
void Stack <T>:: Push (const T& item)
{
 if (top==MaxStackSize -1)
 {
  cerr << "Stack overflow!" <<endl;
  exit(1);
 }
 top++;
 stacklist [top] = item;
}
template <class T>
T Stack <T>::Pop (void)
{
 T temp;
 if (top==-1)
 {
  cerr <<"Attempt to Pop an
   empty stack!" <<endl;
  exit(1);
 }
 temp= stacklist [top];
 top--;
 return temp;
}
```

```
template < class T >
T Stack <T>::Peek (void) const
{
 if (top==-1)
 {
  cerr <<"Attempt to Peek at
    an empty stack!" <<endl;
  exit(1);
 }
 return stacklist [top];
}
template < class T >
int Stack <T>::StackEmpty (void) const
{ return top==-1; }
template < class T >
int Stack <T>::StackFull (void) const
{ return top==MaxStackSize-1; }
template < class T >
void Stack <T>::ClearStack (void)
{ top=-1; }
#endif // STACK_CLASS
```

3. MyString，使之能够满足下面程序所实现的功能，不能使用系统提供的 string 类，要求使用字符数组）

```
void main()
{
    MyString s1,s2,s3;
    s1.set_str("This is my first string.");   // set_str 成员函数用于设置字符串
    s2=MyString("This is my second string.");
    s3=s1+s2;                                 //set_str 成员函数用于设置字符串
    cout<<s3.get_length()<<endl;    // get_length 成员函数用于获取字符串的长度
    cout<<s3<<endl;                 //输出字符串的内容
}
```

String operator + (MyString &