《新标准 C++程序设计》习题解答

第 11 章-第 20 章

郭炜

第十一章习题

1. 结构化程序设计有什么不足？面向对象的程序设计如何改进这些不足？

2. 以下说法正确的是：

A) 每个对象内部都有成员函数的实现代码

B) 一个类的私有成员函数内部不能访问本类的私有成员变量

C) 类的成员函数之间可以互相调用

D) 编写一个类时，至少要写一个成员函数

#C

3. 以下对类 A 的定义，哪个是正确的？

A)
```
class A  {
    private:    int v;
    public :    void Func() { }
  }
```

B)
```
class A {
    int v; A * next;
    void Func() { }
   };
```

C)
```
class A  {
       int v;
  public:
       void Func();
   };
  A::void Func() { }
```

D)
```
class A  {
       int v;
  public:
       A next;
       void Func() { }
```

```
};
#B
```

4. 假设有以下类 A:

```
class A   {
  public:
   int func(int a) { return a * a; }
};
```

以下程序片段，哪个是不正确的？

```
A) A a;  a.func(5);
B) A * p = new A;  p->func(5);
C) A a; A & r = a; r.func(5);
D) A a,b; if( a != b) a.func(5);
#D
```

5. 以下程序，哪个是不正确的？

```
A) int main(){
       class A { int v; };
       A a; a.v = 3; return 0;
   }
B) int main()    {
    class A { public: int v; A * p; };
    A a;  a.p = & a;  return 0;
   }
C)int main()  {
     class A { public:    int v; };
     A * p = new A;
     p->v = 4;  delete p;
     return 0;
   }
D) int main() {
    class A { public: int v; A * p; };
      A a;  a.p = new A; delete a.p;
      return 0;
    }
    #A
```

6. 实现一个学生信息处理程序。输入：姓名，年龄，学号（整数），第一学年平均成绩，第二学年平均成绩，第三学年平均成绩，第四学年平均成绩。输出：姓名，年龄，学号，四年平均成绩。例如：输入：Tom,18,7817,80,80,90,70

输出：Tom,18,7817,80

要求实现一个代表学生的类，并且所有成员变量都应该是私有的。

```
#include <iostream>
#include <cstdio>
#include <cstring>
#include <cstdlib>
using namespace std;
class CStudent
{
  private:
      int age;
      int id;
      char name[20];
      int averageScore[4];
  public:
      int average() {
          int sum = 0;
          for( int i = 0;i < 4; ++ i)
              sum += averageScore[i];
          return sum/4;
      }
      void printInfo() {
          printf("%s,%d,%d,%d",name,age,id,average());
      }
      void readInfo() {
          char buf[110];
          cin.getline(buf,100);
          char * p = strchr(buf,',');
          p[0] = 0;
          strcpy( name,buf);
          sscanf(p + 1, "%d,%d,%d,%d,%d,%d",&id,&age,
              averageScore,averageScore+1,averageScore+2,averageScore+3);
      }
};
int main()
{
```

```
    CStudent s;
    s.readInfo();
    s.printInfo();
    return 0;
}
```

第十二章习题

1. 以下说法中正确的是：

A) 一个类一定会有无参构造函数

B) 构造函数的返回值类型是 void

C) 一个类只能定义一个析构函数，但可以定义多个构造函数

D) 一个类只能定义一个构造函数，但可以定义多个析构函数

#C

2. 对于通过 new 运算符生成的对象

A) 在程序结束时自动析构

B) 执行 delete 操作时才能析构

C) 在包含该 new 语句的函数返回时自动析构

D) 在执行 delete 操作时会析构，如果没有执行 delete 操作，则在程序结束时自动析构

#B

3. 如果某函数的返回值是个对象，则该函数被调用时，返回的对象

A) 是通过复制构造函数初始化的

B) 是通过无参数的构造函数初始化的

C) 用哪个构造函数初始化取决于函数中 return 语句是怎么写的

D) 不需要初始化

#A

4. 以下说法正确的是：

A) 在静态成员函数中可以调用同类的其他任何成员函数

B) const 成员函数不能作用于非 const 对象

C) 在静态成员函数中不能使用 this 指针

D) 静态成员变量每个对象有各自的一份

#C

5. 以下关于 this 指针的说法中不正确的是：

A) const 成员函数内部不可以使用 this 指针

B) 成员函数内的 this 指针，指向成员函数所作用的对象。

C) 在构造函数内部可以使用 this 指针

D) 在析构函数内部可以使用 this 指针

#A

6. 请写出下面程序的输出结果：

```
class CSample {
    int x;
public:
  CSample() { cout << "C1" << endl;  }
  CSample(int n ) {
    x = n;
    cout << "C2,x=" << n << endl;    }
};
int main(){
    CSample array1[2];
    CSample array2[2] = {6,8};
    CSample array3[2] = {12};
    CSample * array4 = new CSample[3];
    return 0;
}
/*
C1
C1
C2,x=6
C2,x=8
C2,x=12
C1
C1
C1
C1
*/
```

7. 请写出下面程序的运行结果：

```
#include <iostream>
using namespace std;
class Sample{
public:
    int v;
    Sample() { };
    Sample(int n):v(n) { };
    Sample( const Sample & x) { v = 2 + x.v ; }
};
Sample PrintAndDouble( Sample o) {
```

```
    cout << o.v;
    o.v = 2 * o.v;
    return o;
}
int main()  {
    Sample a(5);
    Sample b = a;
    Sample c = PrintAndDouble( b );
    cout << endl;
    cout << c.v << endl;
    Sample d;
    d = a;
    cout << d.v ;
}
/*
9
22   (20 也算对)
5
*/
```

8. 下面程序输出的结果是

4,6

请填空：

```
class A {
        int val;
    public:
        A( int n) { val = n; }
        int GetVal() { return val;}
};
class B: public A {
    private:
        int val;
    public:
        B(int n):_____________{ }
        int GetVal() { return val;}
};
int main() {
    B b1(2);
```

```
    cout<<b1.GetVal()<< ","
        << b1.A::GetVal()<< endl;
    return 0;
}

/*
val(2*n),A(3*n)
*/
```

9. 下面程序输出结果是：

0

5

请填空：

```
class A {
public:
    int val;
    A(__________________ ){ val = n; };
    __________________ GetObj() {
        return __________________;
    }
};
int main()  {
    A a;
    cout <<a.val << endl;
    a.GetObj() = 5;
    cout << a.val << endl;
    return 0;
}
/*
int n = 0
int &
val
或：
int n = 0
A &
*this
*/
```

10. 下面程序的输出是：

10

请补足 Sample 类的成员函数。不能增加成员变量。

```
#include <iostream>
using namespace std;
class Sample{
public:
    int v;
    Sample(int n):v(n) { };
};
int main() {
    Sample a(5);
    Sample b = a;
    cout << b.v ;
    return 0;
}
/*
Sample(Sample & a):v(2 * a.v) { }
*/
```

11. 下面程序的输出结果是：

5,5

5,5

请填空

```
#include <iostream>
using namespace std;
class Base {
public:
    int k;
    Base(int n):k(n) { }
};
class Big  {
public:
    int v; Base b;
    Big _____________ { }
    Big _____________{ }
};
int main()  {
    Big a1(5);    Big a2 = a1;
```

```
    cout << a1.v << "," << a1.b.k << endl;
    cout << a2.v << "," << a2.b.k << endl;
    return 0;
}
/*
(int n):v(n),b(n)
(Big & x):v(x.v),b(x.v)
*/
```

12. 完成附录“魔兽世界大作业”里提到的第一阶段作业。

```
// by Guo Wei
#include <iostream>
#include <cstdio>
#include <cstring>
using namespace std;
#define WARRIOR_NUM 5
/*
char * CWarrior::Names[WARRIOR_NUM] = { "dragon","ninja","iceman","lion","wolf" };
红方司令部按照 iceman、lion、wolf、ninja、dragon 的顺序制造武士。
蓝方司令部按照 lion、dragon、ninja、iceman、wolf 的顺序制造武士。
*/
class CHeadquarter;
class CWarrior
{
        private:
                CHeadquarter * pHeadquarter;
                int nKindNo; //武士的种类编号 0 dragon 1 ninja 2 iceman 3 lion 4 wolf
                int nNo;
        public:
                static char * Names[WARRIOR_NUM];
                static int InitialLifeValue [WARRIOR_NUM];
                CWarrior( CHeadquarter * p,int nNo_,int nKindNo_ );
                void PrintResult(int nTime);
};
class CHeadquarter
{
```

```
        private:
                int nTotalLifeValue;
                bool bStopped;
                int nTotalWarriorNum;
                int nColor;
                int nCurMakingSeqIdx; //当前要制造的武士是制造序列中的第几个
                int anWarriorNum[WARRIOR_NUM]; //存放每种武士的数量
                CWarrior * pWarriors[1000];
        public:
                friend class CWarrior;
                static int MakingSeq[2][WARRIOR_NUM]; //武士的制作顺序序列
                void Init(int nColor_, int lv);
                ~CHeadquarter () ;
                int Produce(int nTime);
                void GetColor( char * szColor);

};
CWarrior::CWarrior( CHeadquarter * p,int nNo_,int nKindNo_ ) {
        nNo = nNo_;
        nKindNo = nKindNo_;
        pHeadquarter = p;
}
void CWarrior::PrintResult(int nTime)
{
                char szColor[20];
                pHeadquarter->GetColor(szColor);
                printf("%03d %s %s %d born with strength %d,%d %s in %s headquarter\n" ,
                                nTime, szColor, Names[nKindNo], nNo,
InitialLifeValue[nKindNo],

        pHeadquarter->anWarriorNum[nKindNo],Names[nKindNo],szColor);
}
void CHeadquarter::Init(int nColor_, int lv)
{
        nColor = nColor_;
        nTotalLifeValue = lv;
```

```
        nTotalWarriorNum = 0;
        bStopped = false;
        nCurMakingSeqIdx = 0;
        for( int i = 0;i < WARRIOR_NUM;i ++ )
                anWarriorNum[i] = 0;
}
CHeadquarter::~CHeadquarter () {
        for( int i = 0;i < nTotalWarriorNum;i ++ )
                delete pWarriors[i];
}
int CHeadquarter::Produce(int nTime)
{

        if( bStopped )
                return 0;
        int nSearchingTimes = 0;
        while( CWarrior::InitialLifeValue[MakingSeq[nColor][nCurMakingSeqIdx]] >
nTotalLifeValue &&
                nSearchingTimes < WARRIOR_NUM ) {
                nCurMakingSeqIdx = ( nCurMakingSeqIdx + 1 ) % WARRIOR_NUM ;
                nSearchingTimes ++;
        }
        int nKindNo = MakingSeq[nColor][nCurMakingSeqIdx];
        if( CWarrior::InitialLifeValue[nKindNo] > nTotalLifeValue ) {
                bStopped = true;
                if( nColor == 0)
                        printf("%03d red headquarter stops making warriors\n",nTime);
                else
                        printf("%03d blue headquarter stops making warriors\n",nTime);
                return 0;
        }
        nTotalLifeValue -= CWarrior::InitialLifeValue[nKindNo];
        nCurMakingSeqIdx = ( nCurMakingSeqIdx + 1 ) % WARRIOR_NUM ;
        pWarriors[nTotalWarriorNum] = new CWarrior( this,nTotalWarriorNum+1,nKindNo);
        anWarriorNum[nKindNo]++;
        pWarriors[nTotalWarriorNum]->PrintResult(nTime);
```

```
            nTotalWarriorNum ++;
            return 1;
}
void CHeadquarter::GetColor( char * szColor)
{
            if( nColor == 0)
                        strcpy(szColor,"red");
            else
                        strcpy(szColor,"blue");
}

char * CWarrior::Names[WARRIOR_NUM] = { "dragon","ninja","iceman","lion","wolf" };
int CWarrior::InitialLifeValue [WARRIOR_NUM];
int CHeadquarter::MakingSeq[2][WARRIOR_NUM] = { { 2,3,4,1,0 },{3,0,1,2,4} }; //两个司令部武士的制作顺序序列

int main()
{
            int t;
            int m;
            CHeadquarter RedHead,BlueHead;
            scanf("%d",&t);
            int nCaseNo = 1;
            while ( t -- ) {
                        printf("Case:%d\n",nCaseNo++);
                        scanf("%d",&m);
                        for( int i = 0;i < WARRIOR_NUM;i ++ )
                                    scanf("%d", & CWarrior::InitialLifeValue[i]);
                        RedHead.Init(0,m);
                        BlueHead.Init(1,m);
                        int nTime = 0;
                        while( true) {
                                    int tmp1 = RedHead.Produce(nTime);
                                    int tmp2 = BlueHead.Produce(nTime);
                                    if( tmp1 == 0 && tmp2 == 0)
                                                break;
```

```
                                nTime ++;
                        }
                }
                return 0;
}
```

第十三章习题

如果将运算符 “[]”重载为某个类的成员运算符（也即成员函数），则该成员函数的参数个数是：

A) 0 个 B) 1 个 C) 2 个 D) 3 个

#B

2. 如果将运算符 “ * ” 重载为某个类的成员运算符（也即成员函数），则该成员函数的参数个数是：

A) 0 个 B) 1 个 C) 2 个 D) 0 个 1 个均可

#D

3. 下面程序的输出是：

3+4i

5+6i

请补足 Complex 类的成员函数。不能加成员变量。

```
#include <iostream>
#include <cstring>
using namespace std;
class Complex {
private:
    double r,i;
public:
    void Print() {
        cout << r << "+" << i << "i" << endl;
    }
};
int main() {
    Complex a;
    a = "3+4i"; a.Print();
    a = "5+6i"; a.Print();
    return 0;
}
/*
    Complex(const char * s = NULL ) {
        if( s ) {
            int len = strlen(s);
            char * tmp = new char[len+1];
            strcpy( tmp,s);
```

```
            char * p = strchr(tmp,'+');
            p[0] = 0;
            tmp[len-1] = 0;
            r = atof(s);
            i = atof(p+1);
            delete [] tmp;
        }
        else
            r = i = 0;
    }
```

或

```
    Complex():r(0),i(0) { }
    Complex & operator = (const char * s)
    {
        if( s ) {
            int len = strlen(s);
            char * tmp = new char[len+1];
            strcpy( tmp,s);
            char * p = strchr(tmp,'+');
            p[0] = 0;
            tmp[len-1] = 0;
            r = atof(s);
            i = atof(p+1);
            delete [] tmp;
        }
        else
            r = i = 0;
        return * this;
    }
*/
```

4. 下面的 MyInt 类只有一个成员变量。MyInt 类内部的部分代码被隐藏了。假设下面的程序能编译通过，且输出结果是:

4,1

请写出被隐藏的部分。(您写的内容必须是能全部放进 MyInt 类内部的，MyInt 的成员函数里不允许使用静态变量)。

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

```
class MyInt  {
     int nVal;
     public:
          MyInt( int n) { nVal = n ;}
          int ReturnVal() { return nVal;}          ……
};
int main ()  {
     MyInt objInt(10);
     objInt-2-1-3;
     cout << objInt.ReturnVal();
     cout <<",";   objInt-2-1;
     cout << objInt.ReturnVal();
     return 0;
}
/*
          MyInt & operator-(int n) {
               nVal -= n;
               return * this;
          }
*/
```

5. 下面的程序输出结果是：

(4,5)

(7,8)

请填空:

```
#include <iostream>
using namespace std;
class Point {
private:
     int x;
     int y;
public:
     Point(int x_,int y_ ):x(x_),y(y_) { };
     ____________________________________;
};
_____ operator << ( _____, const Point & p){
     ____________________________________;
     return ___________________;
```

```
}
int main(){   cout << Point(4,5) << Point(7,8); return 0;}
/*
friend ostream & operator<<(ostream & ,const Point & p);
ostream &
ostream & o
o << "(" << p.x << "," << p.y << ")"
o
*/
```

6. 写一个二维数组类 Array2,使得下面程序的输出结果是：

```
0,1,2,3,
4,5,6,7,
8,9,10,11,
next
0,1,2,3,
4,5,6,7,
8,9,10,11,
```

程序：

```
#include <iostream>
using namespace std;
int main() {
    Array2 a(3,4);
    int i,j;
    for(  i = 0;i < 3; ++i )
        for(  j = 0; j < 4; j ++ )
            a[i][j] = i * 4 + j;
    for(  i = 0;i < 3; ++i ) {
        for(  j = 0; j < 4; j ++ ) {
            cout << a(i,j) << ",";
        }
        cout << endl;
    }
    cout << "next" << endl;
    Array2 b;     b = a;
    for(  i = 0;i < 3; ++i ) {
        for(  j = 0; j < 4; j ++ ) {
            cout << b[i][j] << ",";
```

```
            }
            cout << endl;
        }
        return 0;
}
/*
class Array2
{
    private:
        int * buf;
        int row,col; //数组是 row 行，col 列
    public:
        Array2(int r,int c):row(r),col(c),buf(new int[r*c+2]) { }
        Array2():buf(new int[2]),row(0),col(0) { }
        //构造函数确保 buf 不会是 NULL，省得还要考虑 buf == NULL 的特殊情况，麻烦
        //多分配点空间，无所谓
        ~Array2() {
            delete [] buf;
        }
        int * operator [](int i) const {
            return buf + i * col;
        }
        void duplicate(const Array2 & a) {
            if( a.row == 0 || a.col == 0 )
                row = col = 0; //空间暂时不回收也无所谓
            else {
                if( row * col < a.row * a.col ) { //空间不够大才重新分配空间
                    delete [] buf;
                    buf = new int[a.row*a.col];
                }
                memcpy(buf,a.buf,sizeof(int)*a.row * a.col);
                row = a.row;
                col = a.col;
            }
        }
        Array2 & operator = (const Array2 & a) {
            if( a.buf == buf )
```

```
                return * this;
            duplicate(a);
            return * this;
        }
        Array2(const Array2 & a):buf(new int[2]),row(0),col(0) {
            duplicate(a);
        }
        int & operator() ( int r,int c) const {
            return buf[r * col + c];
        }
};
*/
```

7. 写一个 MyString 类，使得下面程序的输出结果是：

1. abcd-efgh-abcd-
2. abcd-
3.
4. abcd-efgh-
5. efgh-
6. c
7. abcd-
8. ijAl-
9. ijAl-mnop
10. qrst-abcd-
11. abcd-qrst-abcd- uvw xyz

about

big

me

take

abcd

qrst-abcd-

程序：

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

```
int CompareString( const void * e1,
          const void * e2) {
    MyString * s1 = (MyString * ) e1;
    MyString * s2 = (MyString * ) e2;
    if( * s1 < *s2 )     return -1;
    else if( *s1 == *s2) return 0;
    else if( *s1 > *s2 ) return 1;
}
main()   {
    MyString s1("abcd-"),s2,
             s3("efgh-"),s4(s1);
    MyString SArray[4] =
          {"big","me","about","take"};
    cout << "1. " << s1 << s2 << s3<< s4<< endl;
    s4 = s3;    s3 = s1 + s3;
    cout << "2. " << s1 << endl;
    cout << "3. " << s2 << endl;
    cout << "4. " << s3 << endl;
    cout << "5. " << s4 << endl;
    cout << "6. " << s1[2] << endl;
    s2 = s1;    s1 = "ijkl-";
    s1[2] = 'A' ;
    cout << "7. " << s2 << endl;
    cout << "8. " << s1 << endl;
    s1 += "mnop";
    cout << "9. " << s1 << endl;
    s4 = "qrst-" + s2;
    cout << "10. " << s4 << endl;
    s1 = s2 + s4 + " uvw " + "xyz";
    cout << "11. " << s1 << endl;
    qsort(SArray,4,sizeof(MyString),
          CompareString);
    for( int i = 0;i < 4;++i )
        cout << SArray[i] << endl;
//输出 s1 从下标 0 开始长度为 4 的子串
    cout << s1(0,4) << endl;
//输出 s1 从下标为 5 开始长度为 10 的子串
```

```
    cout << s1(5,10) << endl;
}

/*

class MyString
{
    private:
        char * str;
        int size;
    public:
        MyString(){
            str = new char[2]; //确保分配的是数组
            str[0] = 0;//既然是个字符串，里面起码也是个空串，不能让 str == NULL
            size = 0;
        }
        MyString(const char * s) {
            //如果 s == NULL，就让它出错吧
            size = strlen(s);
            str = new char[size+1];
            strcpy(str,s);
        }
        MyString & operator=(const char * s ) {
            //如果 s == NULL，就让它出错吧
            int len = strlen(s);
            if( size < len ) {
                delete [] str;
                str = new char[len+1];
            }
            strcpy( str,s);
            size = len;
            return * this;
        }


        void duplicate(const MyString & s) {
            if( size < s.size ) { //否则就不用重新分配空间了
```

```
            delete [] str;
            str = new char[s.size+1];
        }
        strcpy(str,s.str);
        size = s.size;
}
MyString(const MyString & s):size(0),str(new char[1]) {
        duplicate(s);
}
MyString & operator=(const MyString & s) {
        if( str == s.str )
            return * this;
        duplicate(s);
        return * this;
}

bool operator==(const MyString & s) const {
        return strcmp(str,s.str ) == 0;
}
bool operator<(const MyString & s) const {
        return strcmp(str,s.str ) < 0;
}
bool operator>(const MyString & s) const {
        return strcmp(str,s.str ) > 0;
}
MyString operator + ( const MyString & s )     {
        char * tmp = new char[size + s.size + 2];//确保能分配一个数组
        strcpy(tmp,str);
        strcat(tmp,s.str);
        MyString os(tmp);
        delete [] tmp;
        return os;
}
MyString & operator += ( const MyString & s) {
        char * tmp = new char [size + s.size + 2];
        strcpy( tmp,str);
        strcat( tmp,s.str);
```

```
            size += s.size;
            delete [] str;
            str = tmp;
            return * this;
        }

        char & operator[](int i) const {
            return str[i];
        }
        MyString operator()(int start,int len) const {
            char * tmp = new char[len + 1];
            for( int i = 0;i < len ; ++i)
                tmp[i] = str[start+i];
            tmp[len] = 0;
            MyString s(tmp);
            delete [] tmp;
            return s;
        }
        ~MyString() { delete [] str; }

        friend ostream & operator << ( ostream & o,const MyString & s);
        friend MyString operator +( const char * s1,const MyString & s2);


};
ostream & operator << ( ostream & o,const MyString & s)
{
    o << s.str ;
    return o;
}
MyString operator +( const char * s1,const MyString & s2)
{
    MyString tmp(s1);
    tmp+= s2;
    return tmp;
}
*/
```

第十四章习题

1. 以下说法不正确的是（假设在公有派生情况下）

A) 可以将基类对象赋值给派生类对象

B) 可以将派生类对象的地址赋值给基类指针

C) 可以将派生类对象赋值给基类的引用

D) 可以将派生类对象赋值给基类对象

#A

2. 写出下面程序的输出结果：

```
#include <iostream >
using namespace std;
class B {
public:
    B(){ cout << "B_Con" << endl; }
    ~B() { cout << "B_Des" << endl; }
};
class C:public B {
public:
    C(){ cout << "C_Con" << endl; }
    ~C() { cout << "C_Des" << endl; }
};
int main() {
    C * pc = new C;
    delete pc;
    return 0;
}
/*
B_Con
C_Con
C_Des
B_Des
*/
```

3. 写出下面程序的输出结果：

```
#include <iostream >
```

```
using namespace std;
class Base {
public:
    int val;
    Base()
    { cout << "Base Constructor" << endl; }
    ~Base()
    { cout << "Base Destructor" << endl;}
};
class Base1:virtual public Base {  };
class Base2:virtual public Base {  };
class Derived:public Base1, public Base2 { };
int main() {  Derived d; return 0;}
/*
Base Constructor
Base Destructor
*/
```

4. 按照第十三章的第 7 题的要求编写 MyString 类，但 MyString 类必须是从 string 类派生而来。提示 1：如果将程序中所有 "MyString" 用"string" 替换，那么题目的程序中除了最后两条语句编译无法通过外，其他语句都没有问题，而且输出和前面给的结果吻合。也就是说，MyString 类对 string 类的功能扩充只体现在最后两条语句上面。提示 2: string 类有一个成员函数 string substr(int start,int length); 能够求从 start 位置开始，长度为 length 的子串

```
/*
class MyString:public string
{
    public:
        MyString() { }
        MyString(const char * s):string(s) { }
        MyString(string s):string(s) { }
        MyString operator()(int start,int len) const {
            return substr(start,len);
        }
    };
*/
```

5. 完成附录“魔兽世界大作业”里提到的第二阶段作业。

```
// by Guo Wei
#include <iostream>
```

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

//下面这些东西都是常量，而且不止一个类都要用到，就声明为全局的较为简单
#define WARRIOR_NUM 5
#define WEAPON_NUM 3
enum { DRAGON,NINJA,ICEMAN,LION,WOLF };

/*
char * CWarrior::Names[WARRIOR_NUM] = { "dragon","ninja","iceman","lion","wolf" };
红方司令部按照 iceman、lion、wolf、ninja、dragon 的顺序制造武士。
蓝方司令部按照 lion、dragon、ninja、iceman、wolf 的顺序制造武士。
*/
class CHeadquarter;
class CDragon;
class CNinja;
class CIceman;
class CLion;
class CWolf;
class CWeapon;
class CWarrior;

class CWeapon
{
	public:
		int nKindNo;
		int nForce;
		static int InitialForce[WEAPON_NUM];
		static const char * Names[WEAPON_NUM];
};
class CWarrior
{
	protected:

		CHeadquarter * pHeadquarter;
		int nNo;
```

```
        public:

                static const char * Names[WARRIOR_NUM];
                static int InitialLifeValue [WARRIOR_NUM];
                CWarrior( CHeadquarter * p,int nNo_):
                        pHeadquarter(p),nNo(nNo_) {    }

                virtual void PrintResult(int nTime,int nKindNo);
                virtual void PrintResult(int nTime) = 0;
                virtual ~CWarrior() { }

};
class CHeadquarter
{
        private:
                static const int    MAX_WARRIORS = 1000;
                int nTotalLifeValue;
                bool bStopped;
                int nColor;
                int nCurMakingSeqIdx;
                int anWarriorNum[WARRIOR_NUM];
                int nTotalWarriorNum;
                CWarrior * pWarriors[MAX_WARRIORS];
        public:
                friend class CWarrior;
                static int MakingSeq[2][WARRIOR_NUM];
                void Init(int nColor_, int lv);
                ~CHeadquarter () ;
                int Produce(int nTime);
                void GetColor( char * szColor);
                int GetTotalLifeValue() { return nTotalLifeValue; }

};

void CWarrior::PrintResult(int nTime,int nKindNo)
{
        char szColor[20];
```

```
        pHeadquarter->GetColor(szColor);
        printf("%03d %s %s %d born with strength %d,%d %s in %s headquarter\n"         ,
                        nTime, szColor, Names[nKindNo], nNo, InitialLifeValue[nKindNo],
                        pHeadquarter->anWarriorNum[nKindNo],Names[nKindNo],szColor);
}
class CDragon:public CWarrior
{
        private:
                CWeapon wp;
                double fmorale;
        public:
                void Countmorale()
                {
                        fmorale = pHeadquarter -> GetTotalLifeValue() /(double)CWarrior::InitialLifeValue [0];
                }
                CDragon( CHeadquarter * p,int nNo_):
                        CWarrior(p,nNo_) {
                        wp.nKindNo = nNo % WEAPON_NUM;
                        wp.nForce = CWeapon::InitialForce[wp.nKindNo ];
                        Countmorale();
                }
                void PrintResult(int nTime)
                {
                        CWarrior::PrintResult(nTime,DRAGON);
                        printf("It has a %s,and it's morale is %.2f\n",
                                CWeapon::Names[wp.nKindNo], fmorale);
                }
};
class CNinja:public CWarrior
{
        private:
                CWeapon wps[2];
        public:

                CNinja( CHeadquarter * p,int nNo_):
                        CWarrior(p,nNo_) {
                        wps[0].nKindNo = nNo % WEAPON_NUM;
```

```
            wps[0].nForce = CWeapon::InitialForce[wps[0].nKindNo];

            wps[1].nKindNo = ( nNo + 1) % WEAPON_NUM;
            wps[1].nForce = CWeapon::InitialForce[wps[1].nKindNo];
        }
        void PrintResult(int nTime)
        {
            CWarrior::PrintResult(nTime,NINJA);
            printf("It has a %s and a %s\n",
                CWeapon::Names[wps[0].nKindNo],
                CWeapon::Names[wps[1].nKindNo]);
        }
};
class CIceman:public CWarrior
{
    private:
        CWeapon wp;
    public:
        CIceman( CHeadquarter * p,int nNo_):
            CWarrior(p,nNo_)
        {
            wp.nKindNo = nNo % WEAPON_NUM;
            wp.nForce = CWeapon::InitialForce[ wp.nKindNo ];
        }
        void PrintResult(int nTime)
        {
            CWarrior::PrintResult(nTime,ICEMAN);
            printf("It has a %s\n",
                CWeapon::Names[wp.nKindNo]);
        }
};
class CLion:public CWarrior
{
    private:
        int nLoyalty;
    public:
        void CountLoyalty()
```

```
        {
            nLoyalty = pHeadquarter ->GetTotalLifeValue();
        }
        CLion( CHeadquarter * p,int nNo_):CWarrior(p,nNo_) {
            CountLoyalty();
        }
        void PrintResult(int nTime)
        {
            CWarrior::PrintResult(nTime,LION);
            CountLoyalty();
            printf("It's loyalty is %d\n",nLoyalty);
        }
};
class CWolf:public CWarrior
{
    public:

        CWolf( CHeadquarter * p,int nNo_):
            CWarrior(p,nNo_) { }
        void PrintResult(int nTime)
        {
            CWarrior::PrintResult(nTime,WOLF);
        }

};

void CHeadquarter::Init(int nColor_, int lv)
{
    nColor = nColor_;
    nTotalLifeValue = lv;
    bStopped = false;
    nCurMakingSeqIdx = 0;
    nTotalWarriorNum = 0;
    for( int i = 0;i < WARRIOR_NUM;i ++ )
        anWarriorNum[i] = 0;
}
CHeadquarter::~CHeadquarter () {
```

```
        int i;
        for( i = 0;i < nTotalWarriorNum; i ++ )
                delete pWarriors[i];
}
int CHeadquarter::Produce(int nTime)
{
        int nSearchingTimes = 0;
        if( bStopped )
                return 0;
        while( CWarrior::InitialLifeValue[MakingSeq[nColor][nCurMakingSeqIdx]] > nTotalLifeValue &&
                nSearchingTimes < WARRIOR_NUM ) {
                nCurMakingSeqIdx = ( nCurMakingSeqIdx + 1 ) % WARRIOR_NUM ;
                nSearchingTimes ++;
        }
        int nKindNo = MakingSeq[nColor][nCurMakingSeqIdx];
        if( CWarrior::InitialLifeValue[nKindNo] > nTotalLifeValue ) {
                bStopped = true;
                if( nColor == 0)
                        printf("%03d red headquarter stops making warriors\n",nTime);
                else
                        printf("%03d blue headquarter stops making warriors\n",nTime);
                return 0;
        }
        nTotalLifeValue -= CWarrior::InitialLifeValue[nKindNo];
        nCurMakingSeqIdx = ( nCurMakingSeqIdx + 1 ) % WARRIOR_NUM ;
        int nTmp = anWarriorNum[nKindNo];
        anWarriorNum[nKindNo] ++;
        switch( nKindNo ) {
                case DRAGON:
                        pWarriors[nTotalWarriorNum] = new CDragon( this,nTotalWarriorNum+1);
                        break;
                case NINJA:
                        pWarriors[nTotalWarriorNum] = new CNinja( this,nTotalWarriorNum+1);
                        break;
                case ICEMAN:
                        pWarriors[nTotalWarriorNum] = new CIceman( this,nTotalWarriorNum+1);
                        break;
```

```
            case LION:
                pWarriors[nTotalWarriorNum] = new CLion( this,nTotalWarriorNum+1);
                break;
            case WOLF:
                pWarriors[nTotalWarriorNum] = new CWolf( this,nTotalWarriorNum+1);
                break;

        }
        pWarriors[nTotalWarriorNum]->PrintResult(nTime);
        nTotalWarriorNum ++;
        return 1;
}
void CHeadquarter::GetColor( char * szColor)
{
        if( nColor == 0)
            strcpy(szColor,"red");
        else
            strcpy(szColor,"blue");
}

const char * CWeapon::Names[WEAPON_NUM] = {"sword","bomb","arrow" };
int CWeapon::InitialForce[WEAPON_NUM];
const char * CWarrior::Names[WARRIOR_NUM] = { "dragon","ninja","iceman","lion","wolf" };
int CWarrior::InitialLifeValue [WARRIOR_NUM];
int CHeadquarter::MakingSeq[2][WARRIOR_NUM] = { { 2,3,4,1,0 },{3,0,1,2,4} };



int main()
{
        int t;
        int m;
        //freopen("war2.in","r",stdin);
        CHeadquarter RedHead,BlueHead;
        scanf("%d",&t);
        int nCaseNo = 1;
        while ( t -- ) {
```

```
            printf("Case:%d\n",nCaseNo++);
            scanf("%d",&m);
            int i;
            for(i = 0;i < WARRIOR_NUM;i ++ )
                scanf("%d", & CWarrior::InitialLifeValue[i]);
//          for(i = 0;i < WEAPON_NUM;i ++ )
//              scanf("%d", & CWeapon::InitialForce[i]);
            RedHead.Init(0,m);
            BlueHead.Init(1,m);
            int nTime = 0;
            while( true) {
                int tmp1 = RedHead.Produce(nTime);
                int tmp2 = BlueHead.Produce(nTime);
                if( tmp1 == 0 && tmp2 == 0)
                    break;
                nTime ++;
            }
        }
        return 0;
}
```

第十五章习题

1. 以下说法正确的是

A) 在虚函数中不能使用 this 指针

B) 在构造函数中调用虚函数，不是动态联编

C) 抽象类的成员函数都是纯虚函数

D) 构造函数和析构函数都不能是虚函数

#B

2. 写出下面程序的输出结果:

```
#include <iostream>
using namespace std;
class A {
public:
    A( ) {  }
```

```
    virtual void func()
    { cout << "A::func" << endl; }
    ~A( ) { }
    virtual void fund( )
    { cout << "A::fund" << endl; }
};
class B:public A {
public:
    B ( ) {  func( ) ;  }
    void fun( ) {   func( ) ; }
    ~B ( ) {  fund(  ); }
};
class C : public B {
public :
    C( ) { }
    void func( )
    {cout << "C::func" << endl; }
    ~C() {  fund( );   }
    void fund()
    { cout << "C::fund" << endl;}
};
int main()
{   C c;  return 0; }

/*
A::func
C::fund
A::fund
*/
```

3. 写出下面程序的输出结果:

```
#include <iostream>
using namespace std;
class A {
public :
virtual ~A() {cout<<"DestructA" <<endl; }
};
class B: public A {
```

```
public:
virtual ~B() {cout<<"DestructB" << endl;  }
};
class C: public B {
public:
~C() { cout << "DestructC" << endl; }
};
int main() {
    A * pa = new C;
    delete pa;  A a;
    return 0;
}
/*
DestructC
DestructB
DestructA
DestructA
*/
```

4. 写出下面程序的输出结果:

```
#include <iostream >
using namespace std;
class A {
public:
    A( ) {  }
    virtual void func()
    { cout << "A::func" << endl; }
    virtual void fund( )
    { cout << "A::fund" << endl; }
    void fun()
    { cout << "A::fun" << endl;}
};
class B:public A {
public:
    B ( ) {  func( ) ;  }
    void fun( ) {   func( ) ; }
};
class C : public B {
```

```
public :
    C( ) { }
    void func( )
    {cout << "C::func" << endl; }
    void fund()
    { cout << "C::fund" << endl;}
};
int main()
{
    A * pa = new B();
    pa->fun();
    B * pb = new C();
    pb->fun();
    return 0;
}
/*
A::func
A::fun
A::func
C::func
*/
```

5. 下面程序的输出结果是：

```
A::Fun
C::Do
```

请填空

```
#include <iostream >
using namespace std;
class A {
    private:
        int nVal;
    public:
        void Fun()
        { cout << "A::Fun" << endl; };
        void Do()
        {  cout << "A::Do" << endl; }
};
class B:public A {
```

```
    public:
        virtual void Do()
        { cout << "B::Do" << endl;}
};
class C:public B {
    public:
    void Do( )
    {  cout <<" C::Do" <<endl;  }
    void Fun()
    {  cout << "C::Fun" << endl; }
};
void Call( __________ )  {
    p.Fun();  p.Do();
}
int main()  {
    C c; Call( c);
    return 0;
}
/*
B & p
*/
```

6. 下面程序的输出结果是：

```
destructor B
destructor A
```

请完整写出 class A。 限制条件：不得为 class A 编写构造函数

```
#include <iostream >
using namespace std;
class A {  ····· };
class B:public A {
    public:
        ~B() { cout << "destructor B"
          << endl; }
};
int main() {
    A * pa;
    pa = new B;
    delete pa;
```

```
    return 0;
}
/*
class A {

public:
virtual  ~A(){
cout << "destructor A" << endl;
    }
};
*/
```

7. 下面的程序输出结果是：

```
A::Fun
A::Do
A::Fun
C::Do
```

请填空

```
#include <iostream >
using namespace std;
class A {
    private:
        int nVal;
    public:
        void Fun()
{ cout << "A::Fun" << endl; };
        virtual void Do()
{  cout << "A::Do" << endl; }
};
class B:public A {
    public:
        virtual void Do()
{ cout << "B::Do" << endl;}
};
class C:public B {
    public:
    void Do( )
{  cout <<"C::Do"<<endl;  }
```

```
void Fun()
{  cout << "C::Fun" << endl; }
};
void Call(____________)  {
     p->Fun();  p->Do();
}
int main()  {
     Call( new A());
     Call( new C());
     return 0;
}
/*
A * p
*/
```

8. 完成附录“魔兽世界大作业”里提到的终极版作业。

第十六章习题

1. C++标准类库中有哪几个流类？用途分别如何？他们之间的关系如何？

2. cin 是哪个类的对象？cout 是哪个类的对象？

3. 编写程序，读取一行文字，然后将此行文字颠倒后输出。

输入样例：

```
These are 100 dogs.
```

输出样例：

```
.sgod 001 era esehT
```

```
/*
#include <iostream>
#include <cstring>
using namespace std;
char line[3000];
int main()
{
     cin.getline(line,2900);
     int len = strlen(line);
```

```
    for( int i = len - 1; i >= 0; -- i)
        cout.put(line[i]);
    return 0;
}
*/
```

4. 编写程序，输入若干个实数，对于每个实数，先以非科学计数法输出，小数点后面保留 5 位有效数字；再以科学计数法输出，小数点后面保留 7 位有效数字。输入以 Ctrl+Z 结束。

输入样例：

12.34

123456.892255

输出样例：

12.34000

1.2340000e+001

123456.89226

1.2345689e+005

```
/*
#include <iostream>
#include <cstring>
#include <iomanip>
using namespace std;
char line[3000];
int main()
{
    double f;
    while( cin >> f) {
        cout << fixed << setprecision(5)<< f << endl;
        cout << scientific << setprecision(7) << f << endl;
    }
    return 0;
}
*/
```

5. 编写程序，输入若干个整数，对每个整数，先将该整数以十六进制输出，然后再将该整数以 10 个字符的宽度输出，宽度不足时在左边补 0。输入以 Ctrl+Z 结束。

输入样例：

23

16

输出样例:

17
0000000023
10
0000000016

```
/*
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
    int n;
    while( cin >> n) {
        cout << hex << n << endl;
        cout << dec << setw(10) << setfill('0') << n << endl;
    }
    return 0;
}
```

6. printf,scanf 比起 cout 和 cin 有什么优势？

第十八章习题

1. 下列函数模板中定义正确的是：

A)
```
template<class T1, class T2>
T1 fun (T1,T2)  { return T1 + T2; }
```

B)
```
template< class T>
T fun(T a) { return T + a;}
```

C)
```
tempmlate<class T1,class T2>
T1 fun(T1,T2) { return T1 + T2 ; }
```

D)
```
template<class T>
T fun(T a,T b) { return a + b ; }
```

#D

2. 下列类模板中定义正确的是：

A)
```
template<class T1,class T2>
class A : {
  T1 b;
```

```
    int  fun( int a ) { return T1+T2; }
  };
```
B)
```
  template<class T1,class T2>
  class A  {
    int T2;
    T1 fun( T2 a ) { return a + T2; }
  };
```
C)
```
  template<class T1,class T2>
  class A {
     public:
         T2  b; T1 a;
         A<T1>() { }
         T1 fun() { return a; }
  };
```
D)
```
  template<class T1,class T2>
  class A  {
    T2 b;
    T1 fun( double a ) { b =  (T2) a;
         return (T1) a; }
  };
```
#D

3. 写出下面程序的输出结果:

```
#include <iostream>
using namespace std;
template <class T>
T Max( T a,T b)  {
    cout << "TemplateMax" <<endl;
    return 0;   }
double Max(double a,double b){
    cout << "MyMax" << endl;
    return 0;  }
int main()  {
     int i=4,j=5;
     Max( 1.2,3.4);  Max(i,j);
     return 0;
}
/*
```

```
MyMax
TemplateMax
*/
```

4. 填空使得下面程序能编译通过，并写出输出结果：

```
#include <iostream >
using namespace std;
template <_____________>
class myclass {
    T i;
    public:
    myclass (T a)
    { i = a; }
    void show( )
    { cout << i << endl;   }
};
int main() {
    myclass<________> obj("This");
    obj.show();
    return 0;
}
/*
class T
char *
输出：
Thiis
*/
```

5. 下面的程序输出是：

TomHanks

请填空。注意，不允许使用任何常量。

```
#include <iostream>
#include <string>
using namespace std;
template <class T>
class myclass {
    _________;
    int nSize;
public:
```

```
    myclass ( _____, int n)  {
        p = new T[n];
        for( int i = 0;i < n;++i )
            p[i] = a[i];
        nSize = n;
    }
    ~myclass( ) {
        delete [] p;
    }
    void Show()
    {
        for( int i = 0;i < nSize;++i ) {
            cout << p[i];
        }
    }
};
int main() {
    char * szName = "TomHanks";
    myclass<char >obj(_____);
    obj.Show();  return 0;
}
/*
T * p
 T * a
 szName
 输出：
 TomHanks
*/
```

6. 程序员马克斯的程序风格和他的性格一样怪异。很不幸他被开除后老板命令你接替他的工作。马克斯走之前分愤然删除了他写的一个类模板 MyMax 中的一些代码，你只好将其补出来。你只知道 MyMax 模板的作用与求数组或向量中的最大元素有关，而且下面程序的输出结果是：

5

136

请补出马克斯删掉的那部分代码。该部分代码全部位于 "//开头" 和 "//结尾"之间，别处一个字节也没有。

马克在空白处留下了以下三个条件：

1)不准使用除 true 和 false 以外的任何常量,并且不得假设 true 的值是 1 或任何值

2)不得使用任何库函数或库模板（包括容器和算法）
3)不得使用 static 关键字
你不想表现得不如马克斯，所以不论你是否保留马克斯留下的 MyMax 类中的代码，你都要遵守这三个条件。
提示：copy 函数模板的第三个参数是传值的

```
#include <vector>
#include <iostream>
#include <algorithm>
using namespace std;
template <class T>
class MyMax
{
    public:
    T * pMax;  //指向用于存放最大值的变量
    bool bFirst;//记录最大值时会用到的标记
    MyMax (T * p):bFirst(true),pMax(p) { };
   //开头

//……
//结尾
};
class A {
public:
    int i;
    A( int n) :i(n) { };
    A() { };
};
bool operator < ( const A & a1, const A & a2)
{   return a1.i < a2.i ;  }
ostream & operator<<(ostream &o,const A &a )
{   o << a.i;return o;  }
int main() {
    A a[5] = {A(1),A(5),A(3),A(4),A(2)};
    int b[9] = {1,5,30,40,2,136,80,20,6};
    int nMax;
    A aMax;
    MyMax<A> outputa( & aMax);
```

```
    copy(a,a+5,outputa);
    cout << outputa() << endl;
    MyMax<int> output( & nMax);
    copy(b,b+9,output);
    cout << output() << endl;
    return 0;
}
```

```
/*
template <class T>
class MyMax:public iterator<output_iterator_tag,T>

{
	public:
	T * pMax;   //指向用于存放最大值的变量
	bool bFirst;//记录最大值时会用到的标记
	MyMax (T * p):bFirst(true),pMax(p) { };
//开头
	T operator()() {
		return * pMax;
	}
	void operator++() const { }
	MyMax<T> & operator *()   { return * this; }
	MyMax<T> & operator =( const T & val) {
		if( bFirst) {
			bFirst = false;
			* pMax = val;
		}

		else {
			if( * pMax < val )
				*pMax = val;
		}
	}
};
*/
```

第十九章习题

1. 假设 p1,p2 是 STL 中的 list 容器上的迭代器，那么以下语句哪个不符合语法

A) p1 ++ ; B) p1 --;

C) p1 += 1; D) int n = (p1 == p2);

#C

2. 将一个对象放入 STL 中的容器里时：

A) 实际上被放入的是该对象的一个拷贝（副本）

B) 实际上被放入的是该对象的指针

C) 实际上被放入的是该对象的引用

D) 实际上被放入的就是该对象自身

#A

3. 以下关于函数对象的说法正确的是:

A) 函数对象所属的类将()重载为成员函数

B) 函数对象所属的类将 []重载为成员函数

C) 函数对象生成时不需用构造函数进行初始化

D) 函数对象实际上就是一个函数

#A

4. 以下关于 STL 中 set 类模板的正确说法是：

A) set 是顺序容器

B) 在 set 中查找元素的时间复杂度是 o(n) 的(n 代表 set 中的元素个数)

C) 往 set 中添加一个元素的时间是 o(1)的

D) set 中元素的位置和其值是相关的

#D

5. 写出下面程序的输出结果：

```
#include <vector>
#include <iostream>
using namespace std;
class A {
    private :
        int nId;
    public:
    A(int n) {   nId = n;
    cout << nId << " contructor" << endl;  }
    ~A( )
     {cout << nId << " destructor" << endl; }
};
int main()  {
    vector<A*> vp;
    vp.push_back(new A(1));
```

```
    vp.push_back(new A(2));
    vp.clear();   A a(4);
    return 0;
}
/*
1 contructor
2 contructor
4 contructor
4 destructor
*/
```

6. 写出下面程序的输出结果：

```
#include <iostream>
#include <map>
using namespace std;
class Gt
{
public:
    bool operator() (const int & n1,
           const int & n2) const {
        return ( n1 % 10 ) > ( n2 % 10);
    }
};
int main()    {
  typedef map<int,double,Gt> mmid;
  mmid MyMap;
  cout  << MyMap.count(15) << endl;
  MyMap.insert(mmid::value_type(15,2.7));
  MyMap.insert(mmid::value_type(15,99.3));
  cout << MyMap.count(15) << endl;
MyMap.insert(mmid::value_type(30,111.11));
MyMap.insert(mmid::value_type(11,22.22));
    cout << MyMap[16] << endl;
for( mmid::const_iterator i = MyMap.begin();
    i != MyMap.end() ;++i )
    cout << "(" << i->first << ","
          << i->second  << ")" << ",";
    return 0;
```

```
}
/*
0
1
0
(16,0),(15,2.7),(11,22.22),(30,111.11),
*/
```

7. 下面程序的输出结果是：

```
Tom,Jack,Mary,John,
```

请填空：

```
#include <vector>
#include <iostream>
#include <string>
using namespace std;
template <class T>
class MyClass
{
     vector<T>  array;
     public:
          MyClass ( T * begin,int n ):array(n)
{ copy( begin, begin + n, array.begin());}
  void List() {
   ____________________;
   for(i=array.begin();i!=array.end();++i )
          cout << * i << "," ;
  }
};
int main() {
     string array[4] =
    { "Tom","Jack","Mary","John"};
    ____________________;
     obj.List();
     return 0;
}
/*
typename vector<T>::iterator i
MyClass<string> obj(array,4)
```

或

vector<T>::iterator i

MyClass<string> obj(array,4)

*/

8. 下面程序的输出结果是：

A::Print: 1

B::Print: 2

B::Print: 3

请填空：

```
template <class T>
void PrintAll( const T & c ) {
    T::const_iterator i;
    for( i = c.begin(); i != c.end(); ++i)
        _____________________;
};
class A {
    protected:
    int nVal;
    public:
        A(int i):nVal(i) { }
        virtual void Print()
    { cout << "A::Print: " << nVal << endl; }
};
class B:public A {
    public:
        B(int i):A(i) { }
        void Print()
    { cout << "B::Print: " << nVal << endl; }
};
int main(){
    ___________________;
    v.push_back( new A(1));
    v.push_back (new B(2));
    v.push_back (new B(3));
    PrintAll( v);  return 0;
}
/*
```

```
(*i)->Print();
vector<A*> v;
第二个空用 list 或 deque 也可以
*/
```

9. 下面的程序输出结果是:

1 2 6 7 8 9

请填空

```
#include <iostream>
using namespace std;
int main() {
    int a[] = {8,7,8,9,6,2,1};
    ____________________;
    for( int i = 0;i < 7;++i)
        ____________________;
    ostream_iterator<int> o(cout," ");
    copy( v.begin(),v.end(),o);
    return 0;
}
/*
set<int> v
v.insert(a[i])
或第一个空格填:
set<int> v(a,a+7)
第二个空格不填也可
*/
```

第二十章习题

1. static_cast,interpret_cast,dynamic_cast,const_cast 分别用于哪些场合?

2. dynamic_cast 在什么情况下会抛出异常?抛出的异常是什么类型的?用 dynamic_cast 进行基类指针到派生类指针的转换,如何判断安全性?

3. 下面程序的输出结果是:

2 constructed

step1

2 destructed

3 constructed

step2

3 destructed

before return

请填空：

```
#include <iostream>
#include <memory>
using namespace std;
class A
{
    int v;
    public:
    A(int n):v(n) { cout << v << " constructed" << endl; }
    ~A() { cout << v << " destructed" << endl; }
};
int main()
{
    A * p = new A(2);
    ____________________________;
    p = NULL;
    cout << "step1" << endl;
    ptr.reset(NULL);
    p = new A(3);
    ___________________;
    p = NULL;
    cout << "step2" << endl;
    p = ptr.____________________;
    delete p;
    cout << "before return" << endl;
    return 0;
}
/*
    auto_ptr<A> ptr(p)
    ptr.reset(p)
    release()
*/
```

4. 写出下面程序的输出结果：

```
#include <iostream>
#include <memory>
using namespace std;
class A
{
     int v;
public:
     A(int n):v(n) { cout << v << " constructed" << endl; }
     ~A() { cout << v << " destructed"
    << endl; }
};
int main() {
     auto_ptr<A> ptr1(new A(3));
     auto_ptr<A> ptr2;
     ptr2 = ptr1;
     ptr1.reset(NULL);
     cout << "step1" << endl;
     return 0;
}
/*
3 constructed
step1
3 destructed
*/
```

5. 写出下面程序的输出结果：

```
#include <iostream>
using namespace std;
class A { };
class B:public A {};
int main() {
     try {
          cout << "before throwing" << endl;
          throw B();
          cout << "after throwing" << endl;
     }
     catch( A & )
```

```
    { cout << "catched 1" << endl;  }
    catch(B & )
    {    cout << "catched 2"    << endl; }
    catch(...)
    {    cout << "catched 3" << endl; }
    cout << "end" << endl;
    return 0;
}
/*
before throwing
catched 1
end

*/
```

6. 写出下面程序的输出结果：

```
#include <iostream>
#include <exception>
using namespace std;
class A { };
int  func1(int m, int n){
    try {
        if( n == 0 )
            throw A();
        cout << "in func1" << endl;
        return m / n;
    }
    catch(exception ){
        cout << "catched in func1"<< endl;
    }
    cout << "before end of func1" << endl;
    return m/n;
}
int main()
{
    try {
        func1(5,0);
        cout << "in main" << endl;
```

```
    }
    catch(A & a) {
        cout << "catched in main" << endl;
    }
    cout << "end of main" << endl;
    return 0;
}
/*
catched in main
end of main
*/
```

7. 下面程序输出结果是:

catched 2

请填空:

```
#include <iostream>
#include <stdexcept>
#include <typeinfo>
#include <exception>
using namespace std;
class A {
    public:
        virtual void Print()
        { cout << "A::print" << endl;}
};
class B:public A {
    public:
        virtual void Print()
        { cout << "B::print" << endl;}
};
int main()
{
    A a;
    try {
        B & r = dynamic_cast<B&>(a);
        r.Print();
    }
    catch(A &)
```

```
    { cout << "catched 1" << endl;  }
    catch( _______________)
    {    cout << "catched 2" << endl;}
    catch(...)
    {    cout << "catched 3"    << endl;}
    return 0;
}
/*
4种可选答案
exception e
exception & e
bad_cast e
bad_case & e
*/
```