软 件 学 院 《程序设计 C++》期终考试试题 Jan. 2003

姓名 学号 得分

一、 简述题（30 分）

1、 面向对象程序设计的主要特点？

答：封装，继承，多态。通过消息传递来实现程序的运转。

2、 什么是静态绑定和动态绑定？

答：所谓绑定是指，对于参与多态行为的类型，他们具有多态行为的接口是在公共基类的设计中就预先确定的。而非绑定则对于参与多态行为的类型，他们的接口没有预先定义。

在 C++中通过继承实现的多态是动态绑定，通过模板实现的多态是静态绑定。动态绑定的接口是在运行期间（动态）完成的，静态绑定的接口是在编译期间（静态）完成的

3、 在 C++中，重用一段代码，有哪两种方法？分别有什么特点？

答：一种可以使用内联函数；

一种可以使用继承。

4、 结构化程序设计的基本控制结构有哪几种？分别流程图的形式表示。

答：顺序，循环，条件

5、 什么是抽象数据类型（ADT）？试以时间类型为例，简单描述之。

答：抽象数据类型是指一个数学模型以及定义在此数学模型上的一组操作，需要通过固有数据类型(高级编程语言中已实现的数据类型)来实现。抽象数据类型(ADT):用于指定逻辑特性而不指定实现细节的数据结构

二、 程序理解题（20 分）

1．设有一个矩阵：

```
0    2    1
1    0    2
1    2    0
```

现把它放在一个二维数组 a 中，写出执行下面语句后 a 的值

```
for  (int  i=0;  i <=2;  i++)
   for  (int  j=0;  j <=2;  j++)
      a[i][j]  =  a[a[i][j]] [a[j][i]];
```

答：010

200

010

2 #include <iostream >
using namespace std;
void f(int &x,int y)
{ y = x + y;
x = y % 4;
cout < < x < < y < < endl;
}
void main()
{ int x = 4, y = 5;
f(y,x);
cout < < x < < y < < endl;
f(x,x);
cout < < x < < y < < endl;
}

答：09

05

08

04

3. #include <iostream.h>
class A
{

public:
A() { cout < < "in A 's constructor\n "; };

~A() { cout < < "in A 's destructor\n "; };
};
class B
{

public:
B() { cout < < "in B 's constructor\n "; };

~B() { cout < < "in B 's destructor\n "; };
};
class C: public B
{

private:

```
    A  a;
public:
   C(){  cout  < <  "in  C 's  constructor\n ";  };
   ~C(){  cout  < <  "in  C 's  destructor\n ";  };
};
void  main()
{   A  a;
 ①
    B  *p=new  C;
 ②
         delete  p;
 ③
}④
```

答：in A 's constructor

in B 's constructor

in A 's constructor

in C 's constructor

in B 's destructor

in A 's destructor

三、 指出下面程序片段中错误（10 分）

1.
```
      const  int  x=0;
     int  y  =  2;
     const  int  *p1  =  &x;
     int  *const  p2;
     int  *p3;
     p1  =  &y;
     *p1  =  10;
     p2  =  &y;
     p3  =  &x;
```

答："int *const p2;"指针常量声明时就应该被初始化；所以，"p2 = &y;"也错

"*p1 = 10;"错误，常量指针里的内容不可修改；

"p3 = &x;"错误，不能将普通的 int 型指针转化为 const int 型指针；

2.
```
   class  A
   {
       int  m1;
          static  char  m2;
       public:
            void  f1()  {  m1  =  0;  m2  =  1;  };
```

```
        static void f2() { m1 = 2; m2 = 3; };
        void f3() const { m1 = 4; m2 = 5; };
};
```

答："static void f2() { m1 = 2; m2 = 3; };"

错误，静态成员函数使用了非静态成员变量；

"void f3() const { m1 = 4; m2 = 5;};"

错误，在 const 类型的函数中改变了类的非静态数据成员

四、下面的 C++程序能正常结束吗？如果不能，请指出原因。（5 分）

```
class A
{   int   i, j;
  public:
    A() { i=j=0; }
};
class B
{   A *p;
  public:
     B()  {  p = new A;  }
    ~B() {  delete p;  }
};
void f(B x)
{ ......
}
void main()
{ B b;
  f(b);
}
```

答：不能。

在 b 调用函数 f()时，直接传递了 B b 对象，在函数结束时，会自动清除函数中的局部变量和参数所占用的内存，所以对象 B b 被清除掉了。等到 main 函数结束后，系统又会自动结束所有变量的声明周期，去调用 class B 中的析构函数，在 delete p 时，由于 p 所指内容已经被清除过了，就会产生错误清除系统的内存，从而产生错误，是程序不能正常结束。将函数 f 中的参数改为（B &b）即可。

五、编程题（35 分）

1.编写程序，使其能读入最多 10 个正整数，其中输入过程中一旦有零或付整数输入，则停止读取，然后反向输出已读入的正整数。其中，输入的部分要求以函数 GetNums 实现，反向输出的部分以 ReverseWrite 实现，并且要求使用递归。

2．参照以下表格，定义两个类 CStudent（学生） 和 CUnderGraduated（大学生），

要求使用继承，即定义 CUnderGraduated 为 CStudent 的子类，同时使其能完成如下功能：
[1]可以这样定义数组：

CStudent students[20]; CUnderGraduated A[20];
[2] CStudent S("Smith", 0, 1002);
0 代表女生
CUnderGraduated S("John", 3201, 1, 50, "Title: Programming Language C++");

1 代表男生
CUnderGraduated S1 = S;
[3] 在 CUnderGraduated 中提供函数 AddContent 用以增加该学生的论文内容
[4] 对于 CUnderGraduated 的对象 S，S+=X (X 为 int 型) 可以使 S 学生的学分数增加 X，例如 S + =20，可以使 S 学生的学分数增加 20
[5] 对于 CUnderGraduated 的对象 S1 和 S2，可以用 S1 <S2 比较两个大学生学号的先后关系
[6] 提供全局函数，统一输出学生的信息（输出格式没有要求），要求使用多态性

学生 Name 最长 16 个字符 No 正整数 Sex Male 或 Female Name 表示学生姓名 Sex 表示性别 No 表示学号
大学生 Name 最长 16 个字符 No 正整数 Sex Male 或 Female Credits 正整数 Thesis 不定长的一组字符 Name 表示学生姓名 Sex 表示性别 No 表示学号 Credits 表示学生的学分数 Thesis 表示毕业论文

注意：题目要求不得使用系统类 string

1 写出 C 和 C++的关系，并说明应注意的问题。10%
答：
C 和 C++的关系：
C++ 支持 C 所支持的全部编程技巧 ；
任何 C 程序都能被 C++ 用基本相同的方法编写，并具备相同的运行效率和空间；
完全包含 C 语言成分（C 的超集）；
C++ 由 4 个部分有机组成：
C 语言；OOP 面向对象（添加了 OOP 的完全支持）；STL 标准模板库；Inside-model 内置模型。\\xjj
C 和 C++混合编程:
name mangling：extern “C” { … }。
Static Initialization。静态初始化
Dynamic Memory Allocation（分配）
new / delete
malloc / free。
Compatibility（适合） of data structure。

2 给给出影响表达式求值的因素。10%
答：
操作数、操作符和标点符号组成的序列，表示一个计算过程
优先级 a+b*c 解决相邻两个运算符的运算顺序
结合性 a+b-c
求值次序 (a+b)*(a-b) 解决不相邻的两个运算符的运算顺序 与编译系统有关
类型转换 int x=10; float y=2.0; x*y
在求值次序上：操作符的副作用：改变操作数的值 （X+1）*(++X)

3 请举例说明 C++多态的几种表现方式。10%
答：
多态的一般含义是：某一论域中的一个元素存在多种解释。
虚函数：
只有类的成员函数才可以是虚函数
静态成员函数不可以是虚函数
构造函数不可以是虚函数
析构函数可以（往往）是虚函数
当用指针或者引用访问时采用动态绑定；或者子类对象访问时采用动态绑定
函数达到可对基类函数进行覆盖的目的
一名多用：

函数重载:
在同一个作用域中，相同的标识符可以用于定义不同的函数，但要求这些函数应拥有不同的参数（参数类型或个数）
函数重载主要用于定义多个功能相同而参数不同的函数

在 C++中，对重载函数的绑定采用静态绑定，由编译系统根据实参与形参的匹配实现

操作符重载:
动机
语言提供的操作符只定义了针对基本数据类型操作的语义
操作符重载机制提供了对自定义数据类型进行操作的语义描述手段
作用
提高程序的可读性
提高语言的灵活性、可扩充性
类属性：
模板:
源代码复用机制，实际上为提供了一系列的重载函数
参数化模块
对程序模块（如：类、函数）加上类型参数
对不同类型的数据实施相同的操作
多态的一种形式

C++
类属函数
类属类
在面向对象程序设计中，还有下面的多态：
消息的多态。一个公共消息集可以有多种解释。
对象类型的多态，子类对象既属于子类，也属于父类。
对象标识的多态，父类的应用或者指针可以引用或指向子类对象。

多态性可以严格的分为四类：重载多态，强制多态，包含多态，和参数多态，前面两种统称为专用多态，而后面两种也称为通用多态。//xjj

4 何为引用 ？其主要作用是什么？何时需要将一个函数的返回值类型定义为引用类型？如果随意将函数的返回值类型定义为引用类型会造成什么危害？10%
答：
引用定义： C++提供了引用类型，通过引用类型可以定义一个变量，它与另一个变量占用相同的内存空间；或为一个变量取一个别名。
引用作用：引用主要用于函数的形式参数和动态变量名。//C++ Primer
返回引用：当函数返回引用类型时，没有复制返回值。相反,返回对象本身。 //C++ Primer
如果函数返回值的类型是引用或指针类型，则函数不应该把局部量或局部量的地址作为返回值返回

5 如何利用析构函数防止内存漏洞？举例说明。10%
析构函数:
~?类名? 无参数和无返回类型的成员函数
对象消亡时，在系统收回他所占的存储空间之前，系统将自动调用析构函数

主要作用：一般情况下不需要定义析构函数，但是如果对象在创建后申请了一些资源并且没有归还这些资源，则应定义析构函数来在对象消亡时归还对象申请的资源

在对象创建时，系统会为对象分配一块存储空间来存储对象的数据成员，但对于指针类型的数据成员来说，系统只分配了存储该指针所需要的空间，而没有分配指针所指向的空间，对象自己需要申请（作为资源）。同样，在对象消亡时，系统收回的只是指针成员本身的存储空间，而指针所指向的空间需要对象自己归还（作为资源）。

```
class String
{       char *str;
  public:
        String()        {   str = NULL; }
        String(char *p)
    { str = new char[strlen(p)+1];   strcpy(str,p); }
        ~String()
        {   delete []str; }
}
```

当 String 对象创建初始化后，会有 char* str 对象在堆中。如果没有析构函数，但对象会撤销，char* str 仍在堆中，造成内存泄露。而用析构函数，可以在对象撤销时删除 str，防止内存泄露。//自己写的。

6 请说明 C++设计类依附于什么原则将你所定义的成员函数定义为纯虚函数、虚函数或非虚函数？5%

纯虚函数：只给出函数声明而没有给出函数实现的虚成员函数。

只有函数接口会被继承

子类必须继承函数接口

（必须）提供实现代码

一般虚函数

函数的接口及缺省实现代码都会被继承

子类必须继承函数接口

可以继承缺省实现代码

非虚函数

函数的接口和其实现代码都会被继承

必须同时继承接口和实现代码

7 请给出你认为最有价值的 C++程序设计应该遵守的 5 条原则，并简明分析其意义所在。10%

Use const whenever possible:

Guard against potential ambiguity

Never treat arrays polymorphically

Make non-leaf classes abstract

Strive for exception-safe code

Use destructor to prevent resource leaks

8 编写函数 int count_word(const char *text,const char*word)来统计一个英语文本（由参数 text 指向）中的某个单词（由 word 指向）出现的次数。例如：函数调用 count——word （“the theater is showing the film Gone With The Wind ”,"the"）返回值为 3. 10%

```
// Author : yankai
bool beginWith(const char *text, const char* word) {
    while (*word != '\0') {
        if (*word != *text) {
            return false;
        }
        ++text;
        ++word;
    }
    return true;
}

int count_word(const char *text, const char* word) {
    int count = 0;
    while (*text != '\0') {
        if (beginWith(text, word))
            ++count;
        ++text;
    }
    return count;
}
```

9 定义一个时间类 CTime，它表示时分秒，并能实现以下程序段所需的功能。25%

```
CTime t1;//t1 表示的时间为 0 时 0 分 0 秒
CTime t2;//t2 表示的时间为 18 时 10 分 40 秒
int s;cin??s;
t1=t2+s//把 t1 的时间设为 t2 表示的时间加 S 秒
cout??t1??endl;//输出时间格式时分秒
cout??t1-t2??endl;//输出时间差
```

```
//CTime.h
#ifndef CTIME_H_
#define CTIME_H_

#include ?iostream?
using namespace std;

class CTime {
private:
```

```
        long time;

public:
        CTime();

       CTime(long);

       CTime(long,long,long);

       CTime& operator+=(const CTime&);

       CTime& operator-=(const CTime&);

       friend CTime operator+(const CTime& cTime, const CTime& buffer);

       friend CTime operator-(const CTime& cTime, const CTime& buffer);

       friend std::ostream& operator??(std::ostream&, const CTime&);

};

#endif /* CTIME_H_ */



//CTime.cpp
 #include "CTime.h"
 #include ?iostream?

CTime::CTime() :
        time(0) {
 }

CTime::CTime(long l) :
        time(l) {
 }

CTime::CTime(long h, long m, long s) {
        m += 60 * h;
        s += 60 * m;
        time = s;
 }

CTime& CTime::operator+=(const CTime& buffer) {
```

```
        time += buffer.time;
        return *this;
 }

CTime& CTime::operator-=(const CTime& buffer) {
        time -= buffer.time;
        return *this;
 }

CTime operator+(const CTime& cTime, const CTime& buffer) {
        CTime newOne(cTime);
        newOne += buffer;
        return newOne;
 }

CTime operator-(const CTime& cTime, const CTime& buffer) {
        CTime newOne(cTime);
        newOne -= buffer;
        return newOne;
 }

std::ostream& operator??(std::ostream& out, const CTime& cTime) {
        long m = cTime.time / 60;
        long h = m / 60;
        m %= 60;
        long s = cTime.time % 60;
        out ?? h ?? ":" ?? m ?? ":" ?? s;
        return out;
 }

int main(int argc, char **argv) {
        CTime t1;//t1 表示的时间为 0 时 0 分 0 秒
        CTime t2(18, 10, 40);//t2 表示的时间为 18 时 10 分 40 秒
        cout ?? t1 ?? endl;
        cout ?? t2 ?? endl;
        int s;
        cin ?? s;
        t1 = t2 + s;//把 t1 的时间设为 t2 表示的时间加 S 秒
        cout ?? t1 ?? endl;//输出时间格式时分秒
        cout ?? t1 - t2 ?? endl;//输出时间差
 }
```

引用类型与指针类型相比，其优势在哪里？

答：引用类型与指针类型都可以实现通过一个变量访问另一个变量，但访问的语法形式不同：引用是
采用直接访问形式，指针则采用间接访问形式。
在作为函数参数类型时，引用类型参数的实参是一个变量，而指针类型参数的实参是一个变量的
地址。
除了在定义时指定的被引用变量外，引用类型变量不能再引用其他变量；而指针变量定义后可以
指向其他同类型的变量。因此，引用类型比指针类型要安全。
大多数编译程序往往把引用类型作为指针类型来实现，它对使用者而言是透明的。(igroo.com 更新中)

考试科目名称 C++程序设计（A 卷）

考试方式： 闭卷 考试日期 2010 年 1 月 20 日 教师 郑滔

系（专业） 软件学院（软件工程） 年级 2 班级

学号 姓名 成绩

题号	一	二	三	四	五	六	七	八	九	十
分数										

得分 | |

1、（本题满分 15 分）

请简述 C++程序设计语言的设计理念、演化历程（包括主要的贡献者），并阐明 C 和 C++的关系。

得分 | |

2、（本题满分 10 分）

请简述 C 与 C ++混合编程时要注意的问题。

栈的使用问题

数组的存储分配策略不同

在 c++中也允许在结构和联合中定义函数，他们也具有类的基本功能，与 CLASS 所不同的是：结构和联合的成员的默认访问控制为 PUBLIC.

得分	

3、（本题满分　10　分）

inline 函数的作用？随意使用所可能导致的问题？并请阐述合理使用的建议。

得分	

4、（本题满分　10　分）

请给出 C++语言中关键字 const 的几种使用方法，并给出示例？

Const 成员函数：在定义一个成员函数时，给它加上一个 const 声明，表示它是一个获取对象状态的成员函数。

作用：

在 const 成员函数定义的地方，告诉编译程序该函数不应该改变对象数据成员的值

在使用 const 成员函数的地方，告诉编译程序该成员函数不会改变对象的数据成员的值

常量对象：

Const A a；对 a 的操作只能是获取对象的状态，而不能改变它的状态。

指向常量的指针：const int*p;P 只能读取所指向的变量，不能修改所指向的变量的值

指针类型常量：int *const p;指针指向的变量可以改变，但是指针本身不能改变，即指针表示的地址初始化后不再改变。

得分	

5、（本题满分　15　分）

什么是纯虚函数、虚函数和非虚函数？合理定义三种成员函数所应遵循的基

本原则？请给出你认为合理定义的一个实例，并说明。

得分	

6、（本题满分　20　分）

在 C++程序设计中，可以利用析构函数防止资源泄露，请给出模板 auto_ptr 的基本定义、实现，以及应用实例。

得分	

7、（本题满分　20　分）

请阐述在面向对象程序设计语言中引入构造函数机制的原因，并请给出控制一个类创建实例个数的手段（举例说明）。

考试科目名称 高级程序设计 C++ （A 卷）

考试方式： 闭卷 考试日期 2014 年 1 月 12 日 教师 郑滔

系（专业） 软件学院（软件工程） 年级 班级

学号 姓名 成绩

题号	一	二	三	四	五	六	七	八	九	十
分数										

注意：所有作答请写直接写在卷面上。

得分 | | 一、简答题（本题满分 30 分，共三题）

1、 请简述 C++程序设计语言的设计理念、演化历程（包括主要的贡献者），并阐明 C 和 C++的关系（本题 15 分）

强大的表述能力，高效
Father of Simula67、Father of OO programming

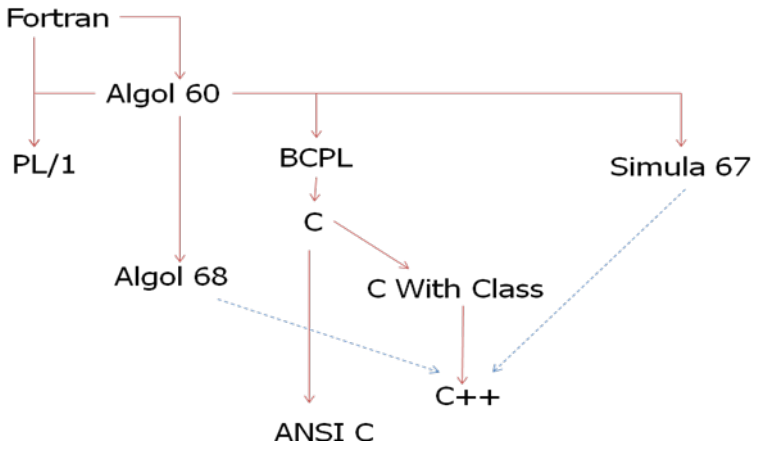
Ole-Johan Dahl、Kristen Nygaard **4 分**

Design PASCAL、Modula2、Oberon
Niklaus Wirth
Structural Programming
E.W.Dijkstra
Father of C、Co-inventing Unix

Dennis Ritchie 、Ken Thompson **2 分**

Design C++

Bjarne Stroustrup **4 分**



5 分

2、 影响表达式值的因素有哪些？请说明之（本题 7 分）

操作符、操作数、优先级、结合性、求值次序、类型转换约定

3、 C++编译系统赋予一个空类，如：class Empty{ } ，哪些成员函数？（本题 8 分）

Empty();
Empty(const Empty&);
~Empty();
Empty& operator=(const Empty&);
Empty *operator &();
const Empty* operator &() const;

得分	

二、请指出以下程序存在的问题（本题满分 10 分，共 2 题）

1、

```
#include <IOSTREAM>
#include <stdlib.h>
#include <time.h>
using namespace std;

class poker{
public:
    unsigned int id;
    poker(){ id = rand() % 13 + 1; } // 产生 1-13 的随机数
};
void main() {
    srand(unsigned(time(NULL)));  // 根据系统时间，设置随机种子值
    poker* p = new poker[5];
    int sum = 0;
    for(int i=0; i<5; i++, p++)
        sum += p->id;
```

```
    cout << "总点数为: " << sum << endl;
    delete p;
}
```

答案：

由于在 for 循环中，移动了 p 指针的位置，导致原先 p 所申请的内存空间的首地址信息丢失，无法归还系统，因此 delete p 时程序会出现异常中止 **5 分**

2、

```
int* get_inputs() {
    int numbers[10];
    int i;
    for(i = 0; i < 10; i ++) {
        cin >> numbers[i];
    }
    return numbers;
}
```

```
int find_max(int* inputs, int size) {
    int i;
    int max = -1;
    for (i = 0; i < size; i ++ ) {
        if (max < inputs[i]) {
            max = inputs[i];
        }
    }
    return max;
}
```

```
void main() {
    cout << "inputs" << endl;
    int* inputs = get_inputs();
    cout << "find the max" << endl;
    int max = find_max(inputs, 10);
    cout << "max: " << max << endl;
}
```

答案：

get_inputs 函数中，返回局部变量。在函数外部使用过程中会造成错误。 **5 分**

得分	

三、请给出以下程序运行结果（本题满分 20 分，共 2 题）

1、

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

class Vehicle{
public:
    virtual void run(int number = 10){
        cout << "we do not know how to run\n";
    }
```

```
    virtual void stop(){
        cout << “we do not know how to stop\n”;
    }
    void announce(){
        cout << "this is a vehicle\n";
    }
};
class Car:public Vehicle
{
public:
    void run(int number = 60){
        cout << "driving at " << number << " km/h\n";
    }
    void stop(){
        cout <<“brake to stop \n”;
    }
    void announce(){
        cout << "this is a car\n";
    }
};

void main()
{
    Vehicle v1,*v2;
    Car c1;
    v1.run();
    c1.announce();

    v2 = &c1;
    v2->run();
    v2->stop();
    v2->announce();
}
```

答案：

we do not know how to run **2 分**

this is a car **2 分**

driving at 10km/h **2 分**

brake to stop **2 分**

this is a vehicle **2 分**

2、

```
class Error {
public:
    virtual void show(){ cout << "something is error"<<endl;}
};
```

```
class nameError:public Error {
public:
	void show() {
		cout<<"name is error"<<endl;
	}
};
```

```
class ageError:public Error {
public:
	void show() {
		cout<<"age is error"<<endl;
	}
};
```

```
class Person {
private:
	int age;
	char* name;
public:
	void setAge(int a) {
		ageError ag;
		if(a<0||a>100)
			throw ag;
		this->age=a;
	}
	void setName(char* str){
		nameError ne;
		if(str=="exit")
			throw ne;
		this->name=str;
	}
};
```

```
void catcher(int command, Person p){
	try {
		switch(command){
		case 1:
			p.setAge(101);
			break;
		case 2:
			p.setName("exit");
			break;
		}
	}
```

```
	catch(nameError ner){
		ner.show();
	}
	catch(Error er) {
		er.show();
	}
	catch(ageError aer){
		aer.show();
	}
}
```

```
int main(void) {
	Person p;
	catcher(1, p);
	catcher(2, p);
```

```
    cout<<"program end"<<endl;
    return 0;
}
```

答案：

something is error　　**4 分**

name is error　　**2 分**

program end　　**4 分**

得分	

四、程序填充（本题满分 10 分）

下列程序输出如下星状图形，请将程序空白部分补充完整：

```
*
* *
* * *
* * * *
```

```
int printEachLine(int j){
    if (__________)
        return 1;
    else{
        ____________________
        ____________________
        return1;
    }
}

int print(int i){
    if (_________)
        return 1;
    else{
        ____________________
        ____________________
        cout<<endl;
        return1;
    }
}
void main(){
    print(4);
}
```

答案：

```
int printEachLine(int j){
    if (j==0)
        return 1;
    else{
        printEachLine(j-1);
```

1分

2分

```
            cout<<"* ";                                          2分
            return 1;
        }
}

int print(int i){
        if (i==0)                                                1分
            return 1;
        else{
            print(i-1);                                          2分

            printEachLine(i);                                    2分
            cout<<endl;
            return 1;
        }
}
```

得分	

五、（编程题。本题满分 30 分。）

观察者模式：定义对象间的一种一对多的依赖关系，当一个对象的状态发生改变时，所有依赖于它的对象都得到通知并被自动更新。

举个博客订阅的例子，当博主发表新内容的时候，即博客内容发生了改变，那些订阅的读者就会收到通知。博主与读者之间存在种一对多的依赖关系，**博客中可能有多个观察者**（即订阅者），当博客的内容发生变化时，通过 notify 成员函数通知所有的观察者，告诉他们博客的内容更新了。而观察者通过 update 成员函数获取博客的内容信息

请根据以下类图和部分类描述，请用 C++给出相关类的实现（Blog, BlogCSDN，Observer，ConsoleObserver）。

类名Blog：

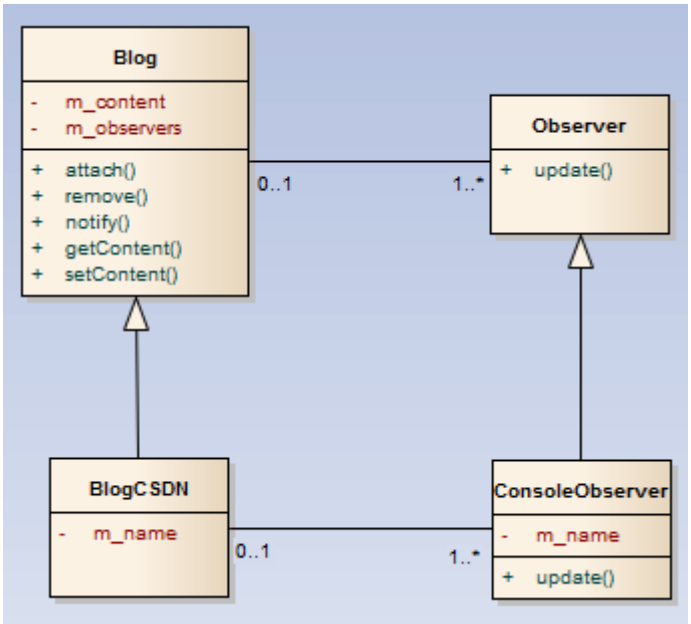
属性	描述
m_content	博客内容（长度< 1024个字符）
m_observers	多个观察者
方法	描述
attach	添加博客听众
remove	删除博客听众
notify	通知所有听众，获取当前博客内容
setContent	设置博客内容
getContent	获取博客内容

类名BlogCSDN:

属性	描述
m_name	博主名称（长度<16个字符）

类名ConsoleObserver：

属性	描述
m_name	观察者名称（长度< 128个字符）
方法	描述
update	获得观察的博客的更新内容，并在控制台显示



答案：

4 个类写出来，并写对继承关系，以及成员变量初始化 **5 分**

写出了 1 对多，即多个观察者 **7 分**

attach、remove、notify、update **每个方法各 4 分**

getContent、setContent **每个方法各 1 分**

```
#include <iostream>
#include <list>

using namespace std;
```

```
//观察者
class Observer
{
public:
	Observer() {}
	virtual ~Observer() {}
	virtual void update() {}
};

//博客
class Blog
{
public:
	Blog() {}
	virtual ~Blog() {}
	void attach(Observer *observer) { m_observers.push_back(observer); }	//添加观察者
	void remove(Observer *observer) { m_observers.remove(observer); }		//移除观察者
	void notify() //通知观察者
	{
		list<Observer*>::iterator iter = m_observers.begin();
		for(; iter != m_observers.end(); iter++)
			(*iter)->update();
	}
	void setContent(char* c) { strcpy(m_content, c); } //设置博客内容
	char* getContent() { return m_content; }	//获得博客内容
private:
	list<Observer* > m_observers; //观察者链表
protected:
	char m_content[1024]; //博客内容
};

//具体观察者
class ConsoleObserver : public Observer
{
private:
	char	m_name[128];
	Blog *m_blog;
public:
	ConsoleObserver(char* name,Blog *blog){
		strcpy(m_name, name);
		m_blog = blog;
	}
	~ConsoleObserver() {}
	void update()	//获得更新内容并输出
```

```
    {
        string content = m_blog->getContent();
        cout<<"Blog content update : " << content << ". My name is : " << m_name << endl;
    }
};

//具体博客类
class BlogCSDN : public Blog
{
private:
    char m_name[16]; //博主名称
public:
    BlogCSDN(char* name){strcpy(m_name, name);}
    ~BlogCSDN() {}

};

//测试用例
int main()
{
    Blog *blog = new BlogCSDN("ZT");

    Observer *observer1 = new ConsoleObserver("GSX", blog);
    Observer *observer2 = new ConsoleObserver("HT", blog);
    Observer *observer3 = new ConsoleObserver("QYG", blog);
    Observer *observer4 = new ConsoleObserver("WLX", blog);

    blog->attach(observer1);
    blog->attach(observer2);
    blog->attach(observer3);
    blog->attach(observer4);

    blog->setContent("C++期末试卷");
    blog->notify();
    blog->remove(observer1);

    delete blog;
    delete observer1;
    delete observer2;
    delete observer3;
    delete observer4;

    return 0;
}
```

考试科目名称 高级程序设计 C++ （A 卷）

考试方式： 闭卷 考试日期 2014 年 1 月 12 日 教师 郑滔

系（专业） 软件学院（软件工程） 年级 班级

学号 姓名 成绩

题号	一	二	三	四	五	六	七	八	九	十
分数										

注意：所有作答请写直接写在卷面上。

得分 | | 一、简答题（本题满分 30 分，共三题）

1、 请简述 C++程序设计语言的设计理念、演化历程（包括主要的贡献者），并阐明 C 和 C++的关系（本题 15 分）

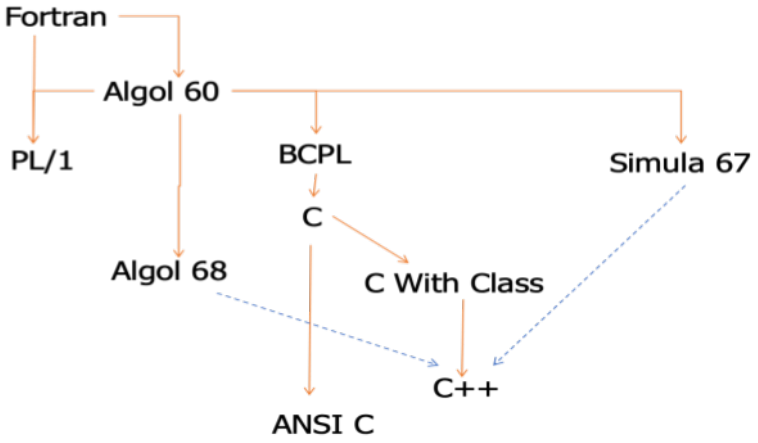
强大的表述能力，高效，实用性高于艺术严谨性，相信程序员

【答：设计理念：效率、实用性优于艺术性严谨性、相信程序员

演化历程：

Father of Simula67 :Kristen Nygaard
Father of OO programming:Ole-Johan Dahl
C 语言之父：Dennis Ritchie 、Ken Thompson
1980 形成 C with class:Bjarne Stroustrup
1983 正式命名 C++：Rick Mascitti
1994 制定 ANSI C++标准草案

/*这几个好像不太重要。。~
* Design PASCAL、Modula2、Oberon
*Niklaus Wirth
*Structural Programming: E.W.Dijkstra



*/】

Father of Simula67、Father of OO programming

Ole-Johan Dahl、Kristen Nygaard **4 分**

Design PASCAL、Modula2、Oberon
Niklaus Wirth
Structural Programming
E.W.Dijkstra
Father of C、Co-inventing Unix

Dennis Ritchie 、Ken Thompson **2 分**

Design C++

Bjarne Stroustrup **4 分**

5 分

2、 影响表达式值的因素有哪些？请说明之（本题 7 分）
操作符、操作数、优先级、结合性、求值次序、类型转换约定

3、 C++编译系统赋予一个空类，如：class Empty{ } ，哪些成员函数？（本题 8 分）
Empty();
Empty(const Empty&);
~Empty();
Empty& operator=(const Empty&);
Empty *operator &();
const Empty* operator &() const;

得分	

二、请指出以下程序存在的问题（本题满分 10 分，共 2 题）

1、

```
#include <IOSTREAM>
#include <stdlib.h>
#include <time.h>
using namespace std;

class poker{
public:
    unsigned int id;
    poker(){ id = rand() % 13 + 1; } // 产生 1-13 的随机数
};
void main() {
    srand(unsigned(time(NULL)));   // 根据系统时间，设置随机种子值
    poker* p = new poker[5];
    int sum = 0;
    for(int i=0; i<5; i++, p++)
        sum += p->id;
    cout << "总点数为: " << sum << endl;
    delete p;
}
```

答案：

由于在 for 循环中，移动了 p 指针的位置，导致原先 p 所申请的内存空间的首地址信息丢失，无法归还系统，因此 delete p 时程序会出现异常中止 **5 分**

2、

```
int* get_inputs() {
    int numbers[10];
    int i;
    for(i = 0; i < 10; i ++) {
        cin >> numbers[i];
    }
    return numbers;
}

void main() {
    cout << "inputs" << endl;
    int* inputs = get_inputs();
```

```
int find_max(int* inputs, int size) {
    int i;
    int max = -1;
    for (i = 0; i < size; i ++ ) {
        if (max < inputs[i]) {
            max = inputs[i];
        }
    }
    return max;
}
```

```
    cout << "find the max" << endl;
    int max = find_max(inputs, 10);
    cout << "max: " << max << endl;
}
```

答案：

get_inputs 函数中，返回局部变量。在函数外部使用过程中会造成错误。 **5 分**

得分	

三、请给出以下程序运行结果（本题满分 20 分，共 2 题）

1、

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

class Vehicle{
public:
    virtual void run(int number = 10){
        cout << "we do not know how to run\n";
    }

    virtual void stop(){
        cout << “we do not know how to stop\n”;
    }
    void announce(){
        cout << "this is a vehicle\n";
    }
};

void main()
{
    Vehicle v1,*v2;
    Car c1;
    v1.run();
    c1.announce();

    v2 = &c1;
    v2->run();
    v2->stop();
    v2->announce();
}
```

```
class Car:public Vehicle
{
public:
    void run(int number = 60){//子类改了也没卵用！~
        cout << "driving at " << number << " km/h\n";
    }
    void stop(){
        cout <<“brake to stop \n”;
    }
    void announce(){
        cout << "this is a car\n";
    }
};
```

答案：

we do not know how to run **2 分**

this is a car **2 分**

driving at 10km/h **2 分**

brake to stop **2 分**

this is a vehicle **2 分**

2、

```
class Error {
public:
    virtual void show(){ cout << "something is error"<<endl;}
};

class nameError:public Error {
public:
    void show() {
        cout<<"name is error"<<endl;
    }
};

class ageError:public Error {
public:
    void show() {
        cout<<"age is error"<<endl;
    }
};

class Person {
private:
    int age;
    char* name;
public:
    void setAge(int a) {
        ageError ag;
        if(a<0||a>100)
            throw ag;
        this->age=a;
    }
    void setName(char* str){
        nameError ne;
        if(str=="exit")
            throw ne;
        this->name=str;
```

```
    }
};

void catcher(int command, Person p){
    try {
        switch(command){
        case 1:
            p.setAge(101);
            break;
        case 2:
            p.setName("exit");
            break;
        }
    }
    catch(nameError ner){
        ner.show();
    }
    catch(Error er) {
        er.show();
    }
    catch(ageError aer){
        aer.show();
    }
}

int main(void) {
    Person p;
    catcher(1, p);
    catcher(2, p);
    cout<<"program end"<<endl;
    return 0;
}
```

答案：

something is error **4 分**

name is error **2 分**

program end **4 分**

得分	

四、程序填充（本题满分 10 分）

下列程序输出如下星状图形，请将程序空白部分补充完整：

```
*
* *
* * *
* * * *
```

```
int printEachLine(int j){
    if (__________)
        return 1;
    else{
        _____________________
        _____________________
        return1;
    }
```

```
}
int print(int i){
    if (_________)
        return 1;
    else{
        ____________________
                                    ____________________
                                    cout<<endl;
                                    return1;
                                }
                            }
                            void main(){
                                print(4);
                            }
```

答案：

```
int printEachLine(int j){
    if (j==0)                                   1分
        return 1;
    else{
        printEachLine(j-1);                     2分
        cout<<"* ";                             2分
        return 1;
    }
}

int print(int i){
    if (i==0)                                   1分
        return 1;
    else{
        print(i-1);                             2分
        printEachLine(i);                       2分
        cout<<endl;
        return 1;
    }
}
```

得分	

五、（编程题。本题满分 30 分。）

观察者模式：定义对象间的一种一对多的依赖关系，当一个对象的状态发生改变时，所有依赖于它的对象都得到通知并被自动更新。

举个博客订阅的例子，当博主发表新内容的时候，即博客内容发生了改变，那些订阅的读者就会收到通知。博主与读者之间存在种一对多的依赖关系，**博客中可能有多个观察者**（即订阅者），当博客的内容发生变化时，通过 notify 成员函数通知所有的观察者，告诉他们博客的内容更新了。而观察者通过 update 成员函数获取博客的内容信息

请根据以下类图和部分类描述，请用 C++给出相关类的实现（Blog, BlogCSDN，Observer，ConsoleObserver）。

类名Blog：

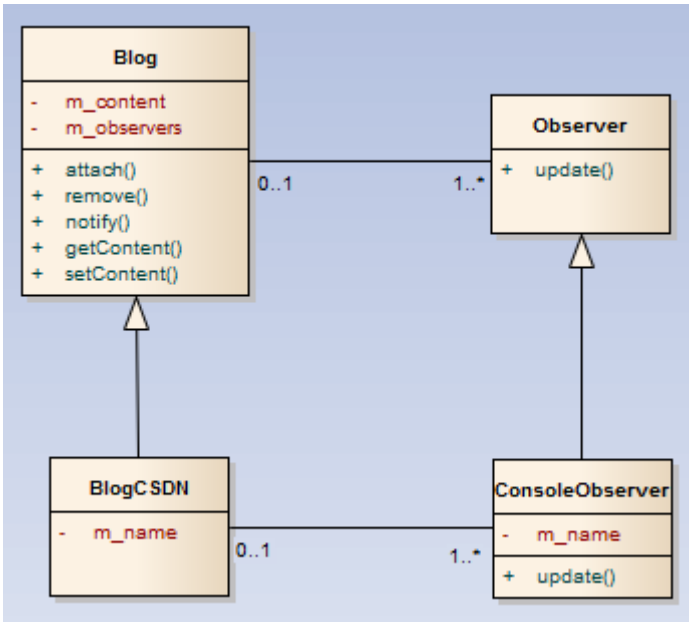
属性	描述
m_content	博客内容（长度< 1024个字符）
m_observers	多个观察者
方法	描述
attach	添加博客听众
remove	删除博客听众
notify	通知所有听众，获取当前博客内容
setContent	设置博客内容
getContent	获取博客内容

类名BlogCSDN:

属性	描述
m_name	博主名称（长度<16个字符）

类名ConsoleObserver：

属性	描述
m_name	观察者名称（长度< 128个字符）
方法	描述
update	获得观察的博客的更新内容，并在控制台显示



答案：

4 个类写出来，并写对继承关系，以及成员变量初始化　　**5 分**

写出了 1 对多，即多个观察者　　**7 分**

attach、remove、notify、update　　**每个方法各 4 分**

getContent、setContent　　**每个方法各 1 分**

```
#include <iostream>
#include <list>

using namespace std;

//观察者
class Observer
{
public:
	Observer() {}
	virtual ~Observer() {}
	virtual void update() {}
};

//博客
class Blog
{
public:
```

```
    Blog() {}
    virtual ~Blog() {}
    void attach(Observer *observer) { m_observers.push_back(observer); }    //添加观察者
    void remove(Observer *observer) { m_observers.remove(observer); }        //移除观察者
    void notify() //通知观察者
    {
        list<Observer*>::iterator iter = m_observers.begin();
        for(; iter != m_observers.end(); iter++)
            (*iter)->update();
    }
    void setContent(char* c) { strcpy(m_content, c); } //设置博客内容
    char* getContent() { return m_content; }      //获得博客内容
private:
    list<Observer* > m_observers; //观察者链表
protected:
    char m_content[1024]; //博客内容
};

//具体观察者
class ConsoleObserver : public Observer
{
private:
    char    m_name[128];
    Blog *m_blog;
public:
    ConsoleObserver(char* name,Blog *blog){
        strcpy(m_name, name);
        m_blog = blog;
    }
    ~ConsoleObserver() {}
    void update()    //获得更新内容并输出
    {
        string content = m_blog->getContent();
        cout<<"Blog content update : " << content << ". My name is : " << m_name << endl;
    }
};

//具体博客类
class BlogCSDN : public Blog
{
private:
    char m_name[16]; //博主名称
public:
    BlogCSDN(char* name){strcpy(m_name, name);}
```

```
    ~BlogCSDN() {}

};

//测试用例
int main()
{
    Blog *blog = new BlogCSDN("ZT");

    Observer *observer1 = new ConsoleObserver("GSX", blog);
    Observer *observer2 = new ConsoleObserver("HT", blog);
    Observer *observer3 = new ConsoleObserver("QYG", blog);
    Observer *observer4 = new ConsoleObserver("WLX", blog);

    blog->attach(observer1);
    blog->attach(observer2);
    blog->attach(observer3);
    blog->attach(observer4);

    blog->setContent("C++期末试卷");
    blog->notify();
    blog->remove(observer1);

    delete blog;
    delete observer1;
    delete observer2;
    delete observer3;
    delete observer4;

    return 0;
}
```

2019年期末

1. 问答题：一共7个题，第一个9分，剩下六个每6个分
    1. C++发展历史上做出贡献的人，simula 67对C++的贡献，C++的设计思路
    2. define替换和template的区别
    3. ……
2. 编译器题：以下代码能不能运行，能运行结果是啥（我回忆下考点）
    1. 忘了，不难
    2. static变量的声明、定义和使用
        1. 能什么时候初始化啦啥的
    3. 数字我瞎编的，重点在那个**:a **和默认参数上

```
int a=1;
 int function(int a,int b=10, int c=11){
   return :a+b+c;
 }

int main(){
  res=function(1.0,5.0)
}
```

    4. 父类有虚函数，子类有虚函数，构造函数里各自调自己的虚函数，构造一个子类，构造函数调的是谁的

```
我大概写一下哈

父类{
virtual void print(){cout<<"我是爸爸"}
构造函数{print()}
}

子类{
void print(){cout<<"我是儿子"}
构造函数{print()}
}

然后new一个子类的实例
```

5. 忘得差不多了……
6. 构造顺序：先父类，然后成员对象（就是自己的属性里有对象），然后自己；析构顺序，反的
    - 举个例子，B继承A，B还持有一个C对象，构造的时候就是先A，然后C，最后B
7. 异常，看清楚抛出的是父类还是子类

3. 代码题：自己写代码。拿分的
    - 写一个表驱动的，就那种分数区间的题
    - 包装一个自动析构，智能指针的类（上课会讲）

考试科目名称 高级程序设计 C++ （A 卷）

考试方式：闭卷 考试日期 2020 年 8 月 22 日 教师 郑滔、潘敏学

系（专业） 软件学院（软件工程） 年级 大二 班级

学号 姓名 成绩

题号	一	二	三	四	五	六	七	八	九	十
分数										

注意：所有作答请直接写在卷面上。

得分	

一、简答题（本题满分 44 分，第 1 小题 8 分，其余小题每题 6 分）

1、 请简述 C++程序设计语言的设计理念、演化历程（包括主要的贡献者），并讨论 Simula 67 在其中的作用。

参见历史题？

2、表达式的值有哪些因素决定？表达式会存在副作用吗？

优先级，结合性，求值次序，类型转换

++X X++会产生副作用

3、请解释指针类型和引用类型的差别。

调用方式不同，一个是—>间接访问 一个是.直接访问

安全要求不同
（1）引用被创建的同时必须被初始化（ 指针则可以在任何时候被初始化）。
（2）不能有 NULL 引用，引用必须与合法的 存储单元关联（指针则可以是 NULL）。
（3）一旦引用被初始化，就不能改变引用的关系（指针则可以随时改变所指的对象）

4、简述 C++中继承与虚继承的差异。

虚继承的虚基类会被合并，只有一个副本
虚继承时，不会先构造虚基类的
虚继承更像是一种组合关系，虚继承的基类无法用 static_cast 强制转换成派生类

虚基类的构造函数优先于非虚基类的构造函数
虚基类由最新派生出的类的构造函数调用

5、 为什么要实现两个版本的下标运算符重载？

因为对非常量对象和常量对象的隐含参数 this 类型是不一样的
需要分别对应
const A* const this
A* const this

此外对 const 对象返回 const 值也更符合语义

6、请简述右值引用与左值引用的异同点。

左值引用必须用于左值，右值引用必须用于右值////（这不废话吗= =）
引用建立后，实际上把右值转换成了左值，两者都能进行正常的赋值等有内存变量的举措

7、简述 Lambda 表达式的作用，并比较它与重载了函数调用操作符的函数对象的差别。

实现匿名函数，使函数声明更加简洁
EMMM 不会写

得分	

二、程序理解题（本题满分 36 分，每小题 6 分）

请仔细阅读代码，并判断代码是否存在错误。若不能通过编译，请指出错误代码和错误原因；若能通过编译，请写出代码运行结果。答案直接写在题目右边空白处。注意，本节所有题目的代码，均已#include <iostream>和 using namespace std。

1、

```
typedef double F;
int main()
{
    int a=8,b=3;
    F c=2.0;
    cout << (a/b/c>1?b/a>0?1:2:3);
    return 0;
}
```

Ans :3

2、

```
void func( long a, long b) {
    cout << "long";
}
void func(double a, double b) {
    cout << "double";
}

int main(int argc, const char * argv[]) {
    int a=1, b=1;
    func(a, b);
    return 0;
}
```

编译不通过，两个都是整型提升？

3、

```
class A{
    int x;
public:
    A(inti = 0) {
        x= i;
        cout<< x << " is constructed" << endl;
    }
```

```
    A(A&A) {
        x= A.x;
        cout<< "Copy of " << x<< " is construced." << endl;
    }
    ~A() {
        cout<< x << " is destructed" << endl;
    }
};

int main(intargc, const char* argv[]) {
    Aa(10), b(a);
    return 0;
}
```

10 is constructed

Copy of 10 is construced.

10 is destructed //这个有顺序吗？

10 is destructed

4、

```
int main()
{
    char str1[10]="aaaaaa";
    char str2[10]="aaa";

    char *p1 = str1;
    char *p2 = str2;
    int sum = 0;
    while (*p1 != '\0')
    {
        if (*p1 == *p2)
        {
            p2++;
            int i = 1;
            while (p1[i] != '\0' && p1[i] == *p2)
            {
                i++; p2++;
            }

            if (*p2 == '\0')
            {
                sum++;
                cout << p1 << endl;
            }

            p2 = str2;
        }
        p1++;
    }

    cout << sum << endl;
    return 0;
}
```

Aaaaaa

Aaaaa

Aaaa

aaa

4

Char* 貌似比较到短方结束

5、

```
class FileErrors{  };
class NonExist : public FileErrors {  };

int main() {
    NonExist e;
    try {
        throw e;
    }
    catch (FileErrors& e) {
        cout << "FileErrors" << endl;
    }
    catch (NonExist& e) {
        cout << "NonExist" << endl;
    }
    return 0;
}
```

Ans:FileErrors

6、

```
class Person {
private:
    string name;
public:
    virtual void printName () const {cout << name<<" ";}
    Person(string name) : name(name) {}
};

class Student: public Person{
private:
  string id;
public:
    Student(string id, string name):id(id), Person(name){printName();}
    void printName() const {
        cout << id << " ";
        Person::printName();
    }
};

void Print(Person& p) {p.printName();}

int main() {
    Student s("101", "Tommy");
    Print(s);
    return 0;
}
```

101 Tommy 101 Tommy 好迷，貌似用名空间限定了就不会找虚函数了

得分	

三、 编程题（本题满分 20 分，每小题 10 分）

1、 利用函数模板设计一个求数组元素总和的函数 sum，使得下方的 main 函数可以正确运行。

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

int main ()
{
```

```
    int a[5]={1,2,3,4,5};
    int s1 = sum(a,5);
    cout<< s1<< endl;
    double b[3]={1.5,2.5,3.5};
    double s2 = sum(b,3);
    cout<< s2<< endl;
    return 0;
}
```

```
template <class T>
T sum(T a[] , int size) {
    T sum=a[0];
    for (int i = 1; i < size; i++) {
        sum += a[i];
    }
    return sum;
};
```

2、 编写一个智能指针类 SmartPtr。

1、 实现基于 RAII 的堆上对象资源的管理，即将一个堆上对象资源封装在一个 SmartPtr 对象的生命周期内，避免资源泄漏（4 分）；

2、 实现 SmartPtr 类的解引用运算符（*）和箭头运算符（->）的重载，从而使得 SmartPtr 对象可以像封装在其中的堆上对象的指针一样使用（4 分）；

3、 使用模板类编写 SmartPtr 类，使得它可以封装各种类型的堆上对象资源（2 分）。

```
template<class T>
class SmartPtr {
public:
    SmartPtr(T* p=nullptr):ptr(p){};
    ~SmartPtr() { delete ptr};
    T* operator ->() const{ return ptr};
    T& operator *() const { return *ptr };
private:
    T* ptr;
};
```