中山大学计算机科学系 2007 级

计算机科学与技术专业、网络工程专业、信息安全专业（ABCDE 班）

程序设计　A 卷

学号 _____　姓名 _____　成绩_____

(试卷共 6 页，答案全写在答题纸上，交卷时连试卷一同交回)

考试形式：闭卷　　任课老师：林瑛、肖菁、杨永红　　2008-6

警示 **《中山大学授予学士学位工作细则》第六条："考试作弊不授予学士学位。"**

一、单项选择 (每小题 1 分，共 15 分)

1. C++**源程序**文件的扩展名为：
   A) lik　　B) cpp　　C) obj　　D) exe

2. 如果函数 A 被声明为类 B 的友元，则：
   A) A 不得访问类 B 的私有成员　　B) A 是类 B 的成员函数
   C) A 可以访问类 B 的私有成员　　D) 类 B 的成员是 A 的友员

3. 关于虚函数的描述中，正确的是：
   A) 派生类的虚函数与基类的同名虚函数具有不同的参数个数和类型
   B) 虚函数是一个非成员函数
   C) 虚函数是一个 static 类型的成员函数
   D) 基类中说明了虚函数后，派生类中对同名函数的重定义时可不必说明为 virtual，其虚特性保持不变

4. 在 C++语言中，允许派生类可兼容基类类型，这是类型兼容性规则，但是这样的类型兼容性只适用于：
   A) 公有派生　　B) 私有派生　　C) 受保护派生　　D) 所有类型的派生

5. 函数的引用性声明（函数原型）不包含哪一部分：
   A) 返回类型　　B) 函数名称　　C) 形式参数　　D) 函数体

6. 对以下声明的类 A，sizeof(A)的值为：

```
class  A {
        static float std;
        float max, min;
public:
        float fun();
};
```

   A) 4　　B) 8　　C) 12　　D) 16

7. 要将一个局部变量的生存期扩展为全局的，则定义它时应加上修饰符：
   A) register　　B) extern　　C) static　　D) auto

8. 在 C++中，什么类只能用作派生其他类的基类，而本身不能创建对象实例：
   A) 基类　　B) 派生类　　C) 虚基类　　D) 抽象类

9. 友元运算符 @obj 被 C++编译器解释为(@表示某种运算符，obj 是其操作数)：
   A) operator@(obj)　　B) operator@(obj,0)
   C) obj.operator@()　　D) obj.operator@(0)

10．下列函数中，不能重载的是：

A）构造函数　　B）析构函数　　C）非类成员的普通函数 D）类的成员函数

11．template<class T>

class APPLE{……};

根据上述语句，定义类模板 APPLE 的成员函数的正确格式是：

A）T APPLE<T>::Push(T obj) {……}　　B）T APPLE::Push(T obj) {……}

C）template<class T>　　D）template<class T>

T APPLE<T>::Push(T obj) {……}　　T APPLE::Push(T obj) {……}

12．可以用友元方式重载的运算符是：

A）+ :: <<　　B）= >> /　　C）+ & []　　D）+ || !

13．编译程序在对函数调用进行静态绑定时，根据以下因素决定调用哪一个函数，**不正确**的是：

A）函数类型　　B）函数名　　C）实际参数个数　　D）实参相应位置的类型

14．在 C++语言中，以下哪个表达式采用了八进制表示整型常量：

A）k = 0123;　　B）k = 123;　　C）k = '\x23';　　D）k = 0x123;

15．类 A 中有一成员函数说明如下 void A::Set(A & a); 其中 A & a 的含义是：

A）指向类 A 的指针为 a

B）将 a 的地址值赋给变量 Set

C）a 是类 A 的对象引用，用作函数 Set()的形参

D）变量 A 与 a 按位与作为函数 Set()的参数

二、程序改错 (共 20 分)

1．指出下列的程序片段的错误（每小题一个错），说明错误原因，并改正：

1）
```
int how_to_do(int x = 0, int y = 0, int z, int w);
```

2）
```
class CIRCLE {
    public:
        CIRCLE(int x = 0, int y = 0, int radius = 1);
        ~CIRCLE();
        int draw();
    private:
        auto int radius;
};
```

3）
```
char *department = "Computer Science Department";
cout << "Depart: " << department << "\n";
delete department;
```

2．阅读下面的程序，指出在哪些行上有语法错误（共 2 个错误），说明错误原因，并在**不改动类数据成员的访问方式**的前提下，改正错误：

```
1)  #include <iostream.h>
2)  class POINT {
3)      public:
4)         POINT(int x, int y): x(x), y(y) {}
5)         int getX() { return x; }
6)         int getY() { return y; }
7)         void print()
8)         { cout << "Point: (" << getX() << ", " << getY() << ")";}
9)      private:
10)        int x = 0, y = 0;
11) };
12) class LINE {
13)     public:
```

```
14)       LINE(int x0, int y0, int x1, int y1)
15)       {  start.x = x0;  start.y = y0;
16)          end.x = x1;    end.y = y1;
17)       }
18)       POINT getStart()  {   return start; }
19)       POINT getEnd()    {   return end; }
20)       void print()
21)       {  cout << "Line from ";  start.print();
22)          cout << " To ";        end.print();
23)          cout << ".\n";
24)        }
25)    private:
26)       POINT start, end;
27) };
28) void main()
29) {   LINE line(0, 0, 50, 50);
30)    line.print();
31) }
```

3. 阅读下面的程序，指出在哪些行上有语法错误（共 3 个错误），说明错误原因，并改正：

```
1)  #include <iostream.h>
2)  template <class TYPE>
3)  class BASE {
4)     public:
5)        void show(TYPE obj)
6)        {  cout << obj << "\n";}
7)  };
8)  template <class TYPE, class TYPE1>
9)  class DERIVED: public BASE<TYPE1> {
10)    public:
11)       void show(TYPE obj1, TYPE1 obj2)
12)       {  cout << obj1 << "\n";
13)          BASE::show(obj2);
14)       }
15) }
16) void main()
17) {   DERIVED<char*, double> obj;
18)    BASE<double> *pBase = &obj;
19)    DERIVED<char*, double> *pDerived = pBase;       // pDerived 指向 obj 对象
20)    obj.show("Pi is ", 3.14);
21) }
```

三、程序填空（共 16 分，每空 2 分）

1. 在以下程序中，希望对成员函数 void who()实现动态绑定，请填相应的语句

```
#include <iostream.h>
class BASE{
    public:
        ____①____{cout<<"BASE\n";}
};
class DERIVED:public BASE{
    public:
        void who(){cout<<"Derivation\n";}
};
void main()
{   ____②____   *p;     // 定义指针 p
    DERIVED  obj;
    ____③____
    p->who();         // 此时调用的是 DERIVED 类中定义的 who()
}
```

2. 要求执行下面的程序输出结果，请填写适当的语句：

```
Calling special version, (10,20)
Calling generic version, (A,B)
```

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

```
template<class TYPE>
TYPE max(TYPE k, TYPE t)
{   cout<< "Calling generic version, (" << k << "," << t << ")\n" ;
    return ____④____;
}

int max(int k, int t)
{   cout<< "Calling special version, (" << k << "," << t << ")\n" ;
    if (k>t)  return k;
    else      return t;
}
// 上述两个 max 版本实现相同功能

void main()
{   ____⑤____;
    ____⑥____;
}
```

3. 下面程序求正整数 a 和 b 之间的奇数之和。

```
#include <iostream.h>
void SumOddEven (int a, int b,____⑦____)
{   *odd = 0;
    for(; a<=b; a++)
       if (a%2)   *odd+=a;
}

void main()
{   int a, b, sodd;
    Cout<<"请输入求和的范围[a，b]："
    cin>>a>>b;
    while (a<0||b<0||b-a<=10)
    {   cout<<"a 或 b 不是正整数，或者不满足 b-a<=10 的条件，请重新输入 a 和 b 的值：";
        cin>>a>>b;
    }

    SumOddEven(a, b,____⑧____);
    cout<<a<<"和"<<b<<"之间的奇数之和为："<<sodd;
}
```

四、程序输出 (共 25 分)

1. 给出以下程序的输出结果

```
#include <iostream.h>
int fib(int n)
{   int result;
    cout << n << " ";
    if (n < 2) result = 1;  else result = fib(n - 1) + fib(n - 2);
    return result;
}
int main()
{   cout << fib(3) << "\n";
}
```

2. 给出以下程序的输出结果

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

const int MAXLEN = 20;

class PERSON {
    public:
        PERSON(char* name = "Zhao", int salary = 300): salary(salary)
        {   strcpy(PERSON::name, name);
            cout << "Constructing Person [" << name << "]\n";
        }
        PERSON(const PERSON& other) // 拷贝构造函数
        {   strcpy(name,other.name);   salary  = other.salary;
            cout << "Copy constructing Person: [" << name << "]" << endl;
        }
```

```
        virtual void show()
        {   cout << "Salary per month for Person [" << name;
            cout << "] is " << salary << " yuan\n";
        }
        PERSON operator =(const PERSON& other)
        {   cout << "Calling Person operator =, set [" << name;
            cout << "] equal to [" << other.name << "]" << endl;
            strcpy(name, other.name);  salary  = other.salary;
            return *this;
        }
        ~PERSON()
        {   cout << "Destructing Person [" << name << "]\n";}
    protected:
        char name[MAXLEN+1];
        int salary;
};

class EMPLOYEE: public PERSON {  //雇员类
    public:
        EMPLOYEE(char* name = "Qian", int salary = 200): PERSON(name, salary)
        {   cout << "Constructing Employee [" << name << "]\n";}
        virtual void show()
        {   cout << "Salary per week for Emplyee [" << name;
            cout << "] is " << salary << " yuan\n" ;
        }
        ~EMPLOYEE()
        {   cout << "Destructing Employee [" << name << "]\n";}
};

class PROFESSOR: public EMPLOYEE { //教授类，每位教授都是一个雇员
    public:
        PROFESSOR(char* name = "Sun", int salary = 10):
        EMPLOYEE(name, salary), assistant("NULL",0)//设当前教授对象雇用名为 NULL 的助教
        {   cout << "Constructing Professor [" << name <<"]\n";}
        virtual void show()
        {   cout << "Salary per hour for Professor [" << name;
            cout << "] is " << salary << " yuan\n";
        }
        void setAssistant(PERSON ass)//指定该教授的助教是谁
        {   assistant = ass; }
        ~PROFESSOR()
        {   cout << "Destructing Professor [" << name << "]\n";}
    protected:
        PERSON assistant;  // 每位教授都雇用了一名助教，助教是 PERSON 类的对象
};

void main()
{   PERSON *  personPtr[3];
    PERSON    person;
    EMPLOYEE  employee;
    PROFESSOR professor;
    cout<<"----------------------------------------\n";   //输出时只画一条直线即可，
                                                           //不必计算输出多少个-号
    personPtr[0] = &person;
    personPtr[1] = &employee;
    personPtr[2] = &professor;
    for (int i = 0; i < 3; i++) personPtr[i]->show();
    cout<<"----------------------------------------\n";
    professor.setAssistant(person);
    cout<<"----------------------------------------\n";
}
```

五、程序设计（24 分）

1. （8 分）类属类 *ARRAY* 描述一个**动态**数组。数组中的元素记录在一个长度为 n 的数组 X 中，要求 X 根据使用时的实际长度 n 动态分配。*ARRAY* 提供如下操作：
   - ✧ 赋值运算（将一个 *ARRAY* 数组赋给另一个 *ARRAY* 数组）；
   - ✧ 以友元形式重载了运算符+实现两个 *ARRAY* 数组对应位置上的元素相加；
   - ✧ 求该 *ARRAY* 数组的最大值。

ARRAY 的用法如以下主程序所示。要求根据主程序中的应用，考虑 *ARRAY* 类应该提供什么功能，并给出 *ARRAY* 类界面的声明（类中声明了什么成员），**不必提供 *ARRAY* 的类实现（不必定义类的成员函数）**。

提示：定义该类属类的操作，使其可以实现主程序中使用到该类的功能。

```
void main()
{  ARRAY<int> s1;      //  ARRAY 类如此使用，考虑它是否是一个普通的类？
   ARRAY<int> s3(10); // 10 是数组长度
   int max1;
   ……
   s3    = s1 + s3;
   max1  = s3.max();
}
```

2. （16 分）设计一个词典类 Dic，每个单词包括英文单词及对应的中文含义，提供构造这个词典的操作，并有一个英汉翻译成员函数，通过查词典的方式将一段英语翻译成对应的汉语。

提示：

- ✧ 例如；要把英语“I am a student”翻译为中文“我是一个学生”，则应先在词典类中添加("a","一个")、("I","我")、("am","是")、("student","学生")中英文单词对。
- ✧ 可考虑使用以下库函数：
  - char* strcpy(char* str1, const char* str2);
    将 str2 指向的字符串复制到 str1 指向的位置中并返回 str1。注意为 str1 分配的存储空间必须能放得下 str2 指向的字符串。
  - int strcmp(const char* str1, const char* str2);
    比较两个字符串 str1 和 str2 的内容是否相同（按字典排序方法）。如果 str1 小于 str2 则返回负数，str1 等于 str2 则返回零，str1 大于 str2 则返回正数。

要求：

1). 从文本文件 in.data 中读入要翻译的英语，例如“I am a student”；
2). 从文本文件 dic.data 中读入中英文单词对，建立词典类；
3). 将翻译结果，例如“我是一个学生”，输出到文本文件 out.data；
4). 上述指定文件名不得更改；
5). 不考虑程序效率问题；
6). **请尽量多提供注释。**

《完》