中山大学计算机科学系 2004 级

程序设计　A1 卷

学号 _____　姓名 _____　成绩_____

(试卷共 6 页)

请考生认真阅读以下注意事项后再答题：

1．本卷为闭卷考试，考生要独立完成，请大家自觉遵守考试纪律；
2．监考员不解释题目。本卷考试时间为 60 分钟，满分为 50 分。开卷后，考生要先交卷后方能退出考场；
3．请考生认真审好题，并按题目要求答题。若在多页答题纸上答题，请在各页答题纸右上角标上页号，并在各页答题纸上左上角写上自己的姓名和学号；
4．在答题纸上答题时可不抄题目，但必须写清楚题号，不写清楚题号或写错题号时将无法记分。用铅笔或红颜色笔答题将视为无效；
5．请考生不要把手机、呼机等电子通讯工具带入考场，若已带入，请关闭电源，并摆在自己桌面上；
6．请把自己的学生证(或校园卡、身份证)放在自己桌面左前方，以便监考员检查。

一、单项选择(每小题 1 分，共 10 分)

1．假定一个类有两个数据成员 a 和 b，其构造函数为：

```
A(int aa=1,int bb=0){ a = aa;  b = bb;  }
```

则执行语句 A x(4); 后，x.a 和 x.b 值分别是：

A) 1 和 0　　B) 1 和 4　　C) 4 和 0　　D) 4 和 1

2．派生类的对象可访问：

A) 公有继承的基类公有成员　　B) 公有继承的基类私有成员
C) 公有继承的基类保护成员　　D) 私有继承的基类公有成员

3．假定 AB 为一个类，则执行语句

```
AB  a(2), *p[3], b[4];
```

时，自动调用该类构造函数的次数为：

A) 3　　B) 5　　C) 6　　D) 9

4．设运行环境中 int 类型长度为 2 字节，且 ptr 是一个 int 类型的指针，且 ptr 的值是 2400，ptr 指向的整数变量的值是 24，那么执行 a = *ptr++; 后，a 和 ptr 的值分别是：

A) 25 和 2400　　B) 24 和 2401　　C) 24 和 2402　　D) 25 和 2402

5．如果类 A 被说明成类 B 的友元，则：

A) 类 A 的成员即类 B 的成员　　B) 类 B 的成员即类 A 的成员
C) 类 A 的成员函数不得访问类 B 的成员　　D) 类 B 不一定是类 A 的友元

6．运算结果类型相同的是：

A) 9/2.0 和 9/2　　B) 9.0/2.0 和 9.0/2
C) 9.0/2 和 9/2　　D) 9/2 和 9.0/2.0

7．while (!x) 中的 !x 与下面哪个条件等价：

A) x==1　　B) x!=1　　C) x!=0　　D) x==0

8．重载赋值运算符时，应声明为什么函数：

A) 友元函数　　B) 虚函数　　C) 成员函数　　D) 构造函数

9．假定一个二维数组的定义语句为“int a[3][4]={{3,4},{2,8,6}};”，则元素 a[1][2]的值为：

A) 2　　B) 4　　C) 6　　D) 8

10．类成员运算符 @obj 被 C++编译器解释为(@表示某种运算符)：

A) obj.operator@()　　B) obj.operator@(0)

C) operator@(obj)　　D) operator@(obj,0)

二、程序改错(每个错 2 分，共 10 分)：指出以下题目所示程序段的语法错误(程序的每一行前加上了行号，可通过行号来指出错误位置)，说明其错误原因？并改正之！(共 5 个错误)

1． 此程序段包含三个错误，请在不修改主函数以及不添加函数的前提下改正之：

```
(1)  class MyClass{
(2)  public:
(3)      MyClass(int ini) { member = ini; }
(4)      int GetMember() const { return member; }
(5)      void SetMember(int m) { member = m;}
(6)      void ~MyClass(){ }
(7)  private:
(8)      int member = 0;
(9)  };
(10) void main()
(11) {   MyClass obj1;
(12)     MyClass obj2(3);
(13)     obj1.SetMember(10);
(14) }
```

2． 此程序包含两个错误：

```
(1)  #include <iostream.h>
(2)  template <class TYPE>
(3)  class BASE {
(4)  public:
(5)        BASE(int flag)
(6)        {     cout<<"Constructing: BASE : "<<flag<<endl; }
(7)        void show(TYPE obj)
(8)        {     cout<<obj<<"\n";}
(9)  };
(10) template <class TYPE, class TYPE1>
(11) class DERIVED: public BASE<TYPE1> {
(12) public:
(13)       DERIVED()
(14)       {     cout<<"Constructing: DERIVED!\n"; }
(15)       void show(TYPE obj1, TYPE1 obj2)
(16)       {     cout<<obj1<<"\n";
(17)             BASE::show(obj2);
(18)       }
(19) };
(20) void main()
(21) {     DERIVED<char *,int> obj;
(22)       obj.show("abc", 100);
(23) }
```

三、**程序输出(共 10 分)：**写出以下程序的输出结果

1.
```
#include <iostream.h>
#include <string.h>
class CARTOON{
  public :
     CARTOON(char *name = "NULL")
     {  strcpy(CARTOON::name,name);
        cout<<"Construct: Cartoon ["<<name<<"]\n";
     }
     CARTOON(const CARTOON& other)
     {  strcpy(name,other.name);
        cout<<"Copy Construct: Cartoon ["<<name<<"]\n";
     }
     ~CARTOON()
     {  cout<<"Destruct: Cartoon ["<<name<<"]\n"; }
     CARTOON operator= (const CARTOON& other)
     {  if (&other==this) return *this;
        cout << "Calling operator =, set [" << name;
        cout << "] equal to [" << other.name << "]" << endl;
        strcpy(name, other.name);
        return *this;
     }
  protected:
     char name[30];
};

class MOUSE: public CARTOON {
  public:
     MOUSE(char* name = "Mickey"): CARTOON(name)
     {  cout << "Construct: MOUSE [" << name << "]\n";}
     ~MOUSE()
     {  cout<<"Destruct: MOUSE ["<<name<<"]\n";}
     void set(CARTOON other)
     {  friends = other;
     }
  protected:
     CARTOON friends; // 老鼠的朋友
};

void main()
{ CARTOON   duck("Donald");
  MOUSE     mouse;
  mouse.set(duck);
}
```

2.
```
#include <iostream.h>
template<class T>
void sort(T* a,int n)
{  T num;
   for(int i=n-2; i>=0; i--){
      for(int j=0; j<=i; j++)
         if (a[j]<a[j+1]){
            num=a[j]; a[j]=a[j+1]; a[j+1]=num;
         }
   }
   for(i=0; i<n; i++)
      cout<< a[i] << " " ;
   cout<<endl;
}
void main()
{  int iver[5]={ 12,45,9,23,37 };
   double dver[5]= { 22.3,56.7,43.5,13.8,8.6 };
   sort(iver,5);
   sort(dver,5);
}
```

四、程序填空（共 12 分，每空 1.5 分）：根据以下各小题的描述和要求在指定位置填入适当语句

1. 完成如下的程序，使得输出为：

```
base::10
base::12
derived::24
```

```
#include <iostream.h>
class base {
      int x;
public:
      base(int a) { x=a; }
      ____①____{ cout<<"base::"<< x << endl; }
};
class derived: public base {
      int y;
public:
      derived(int a,int b):base(a) { y=b; }
      void print() { ____②____; cout<<"derived::"<<y<<endl; }
};
void main()
{     base b(10), *p;
      derived d(12,24);
      b.print();
      ____③____;
      p->print();
}
```

2. 在以下程序中，函数 fill(int square[MAX_LEN][MAX_LEN], int max_num)

将数字 1, 2, ..., max_num, ..., max_num^2 按逆时针填入 max_num×max_num 的矩阵中，下面是 max_num = 5 和 max_num = 6 时主程序的输出结果：

max_num = 5

1	16	15	14	13
2	17	24	23	12
3	18	25	22	11
4	19	20	21	10
5	6	7	8	9

max_num = 6

1	20	19	18	17	16
2	21	32	31	30	15
3	22	33	36	29	14
4	23	34	35	28	13
5	24	25	26	27	12
6	7	8	9	10	11

请完成该函数中的空白，使程序运行能得到正确结果。

```
#include <iostream.h>
#include <iomanip.h>
const int MAX_LEN = 20;
void fill(int square[MAX_LEN][MAX_LEN], int max_num)
{  int curr_num;
   int curr_level;
   int row, col; //row 是行号，col 是列号
   curr_num = 1;   curr_level = 1;
   do {   //逆时针生成一圈的矩阵数据
      row = curr_level;  col = curr_level;
      do {
         square[row][col] = curr_num;  row++;  curr_num++;
      } while (____④____);
      do {
         square[row][col] = curr_num;  col++;  curr_num++;
      } while (____⑤____);
      do {
         square[row][col] = curr_num;  row--;  curr_num++;
      } while ( row > curr_level );
      do {
         square[row][col] = curr_num;  col--;  curr_num++;
      } while ( col > curr_level );
      curr_level++;
   } while (curr_level <= (max_num / 2));
   if ((max_num % 2) != 0) square[max_num/2+1][max_num/2+1] = curr_num;
   return;
}
void print(int square[MAX_LEN][MAX_LEN], int max_num)    //输出矩阵
{  int row, col;
   for (row = 1; row <= max_num; row++) {
         for (col = 1; col <= max_num; col++)
            cout << setw(4) << ____⑥____;
         cout << "\n";
   }
   return;
}
void main()
{  int square[MAX_LEN][MAX_LEN];
   int max_num;
   cout<<"Please input the max num of matrix: ";
   cin>>max_num;
   fill(square, max_num);
   print(square, max_num);
}
```

3. 假定 NODE 的定义为：

```
struct NODE{
      int   data;
      NODE *next;
};
```

下面算法是依次显示输出以 L 为表头指针的链表中各结点的值。

```
void ff(NODE* L)
{     for( ____⑦____ ; p!=NULL; ____⑧____ )
            cout<< p->data <<' ';
      cout<<endl;
}
```

五、程序设计（6 分）

类 *ARRAY* 描述一个集合。集合中的元素记录在一个长度为 n 的数组 X 中，数组的每个元素都是整数类型，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* 的用法如以下程序所示。要求给出 *ARRAY* 的类界面。

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，不必提供 *ARRAY* 的类实现。

```
void main()
{  ARRAY  s1, s2
   ARRAY  s3(10); // 10 是数组长度
   int max1;
   ……
   s3    = s1 + s2;
   max1  = s3.max();
}
```