《新标准 C++程序设计》习题解答

第 1 章-第 10 章

郭炜

第一章

1. 将下列十进制数表示成 16 位二进制形式和 4 位十六进制形式：255，-254，-1,10,20，-12。

解答：题目的意思是，如果在计算机内部用 16 位二进制形式和 4 位 16 进制形式表示上面的数，会是什么样子。要求最高位是符号位，负数的符号位是 1。因此答案为：

255:	0000 0000 1111 1111,	00FF
-254:	1111 1111 0000 0010,	FF02
-1:	1111 1111 1111 1111,	FFFF
10:	0000 0000 0000 1010,	000A
20:	0000 0000 0001 0100,	0014
-12:	1111 1111 1111 0100,	FFF4

2. 将下列 16 位的有符号二进制数转换成十进制形式：

1000 1111 0000 1111, 0000 1011 0000 1111, 1111 1111 0000 1111

1111 1111 1111 1110, 1000 0000 0000 0000, 0000 0000 1100 1110

解答： -28913,2831,-241,-2,-32768,206

3. 将下列有符号 4 位 16 进制数转换为十进制数：

FC34， 7000， 00a5, 1004， 7F45, 7700, C0C0, 0FFF,FFFF

解答：-972,28672,165,4100,32581,30464,-16192,4095,-1,

第二章

1. 以下哪些是合法的 C++标识符，哪些不是？

2Peter

__day

_num_of

sch-name;

解答：第一个和第四个不是，因为标识符不能以数字开头，中间不能有除了“_”和“-”以外的标点符号。其他的是合法的。

2. 编写一个程序，输入 3 个整数，输出他们的平均数。

解答：

```
#include <iostream>
using namespace std;
int main()
{
    int a,b,c;
    cin >> a >> b >>c;
    cout << (a+b+b)/3.0;
    return 0;
}
```

3. 说出下面各个类型的变量所占的字节数和表示范围：

short , int, unsigned int, long long, unsigned char, char

解答：参见本章正文

4. 已知字母’a’的 ASCII 码是 97，请写出下面程序的输出结果:

```
#include <iostream>
using namespace std;
int main()
{
  int n1 = 'a';
  unsigned short n2 = 0xffff;
  int n3 = n2;
  short n4 = n2;
  cout << n1 << "," << n2 << "," << n3 << "," << n4  << endl;
  double f = 6/5;
  n3 = 5/(double) 2;
  char c = 102;
  int n5 = 0xffffffff + 2;
  cout << c << "," << f << "," << n3 << "," << n5 << endl;
  return 0;
```

}

解答：

97,65535,65535,-1

f,1,2,1

解释：n4 是有符号的，会表示负数，n4=n2 执行后，n4 的内容是 n2 的拷贝，即 n4 最高位为 1，表示负数，因此输出 n4，得-1

5. 计算下列表达式的值(答案可写十六进制）

(1) 5 * 4 / 3 + (7 % 2)

(2) 0xfff4 >> 2

(3) 0xea8 << 3

(4) 12 ^ 23

(5) ~24

(6) 0x7fff0000 >> 3

解答：

(1)7

(2)3ffd

(3)7540

(4)1b

(5)fffffffe7

(6)fffe000

6. 已知有 int a = -10, b = 20, c = 30; 请写出以下每个表达式计算结束后 a 的值。

(1) a = b = ++c

(2) a = b | c

(3) a = (b > c)

(4) b ++ && (a += 10)

(5) a ^= b

(6) a <<= 5

(7) a >> 4

(8) a >>= 4

(9) a = sizeof(int)

(10) a = sizeof(char)

(11) a = sizeof(double)

(12) a+=a-=a*a

解答：

(1)31

(2)30　10100 | 11110 = 11110 即是 30

(3)0　(4)0

(5)-30　a=-10,其十六进制形式是：FFFF FFF6

(6)-320, 因左移 5 位后十六进制形式为 FFFF FEC0

(7)-10　不会改变 a

(8)-1　左移动 4 位后，高位补符号位 1，因此结果的为 FFFF FFFF

(9)4　(10)1　(11)8　(12) -220

7. a 是 int 型变量，请写一个表达式，表达式的值和 a 的第 i 位相等(i = 0 ... 31)。

解答：(a >> i) & 1

8. a 是 int 型变量，请写一个表达式，表达式的值等于 a 的第 i 位取反(i= 0 ... 31)。

解答：((a >> i) & 1)^1

9. 已知有 int 类型变量 a,b，请写一条语句，使得 a 的第 3 位到第 7 位和 b 相同，其余位都是 0。

解答：a = b & 0xf8;

10. 已知有 int 类型变量 a,b,c, 请写一条语句，使得 a 的第 3 位到第 7 位和 b 相同，其余位都和 c 相同。

解答：a = (b & 0xf8) | (c & 0xff07);

11. 已知有 int 类型变量 a,b,请写一条语句，使得 a 的第 3 位到第 7 位和 b 的第 27 到 31 位相同，其余位都是 0。

解答：a = (b & 0xF8000000) >> 24;

12. 写出下面程序片断的输出结果：

(1)

```
int a = 0, b = 30;
bool c = a ++ || b ++;
```

```
cout << a << ″,″ << b << ″,″ << c << endl;
```

解答：1,31,1

(2)

```
int a = 0, b = 30;
bool c = a ++ && b ++;
cout << a << ″,″ << b << ″,″ << c << endl;
```

解答： 1,30,0

逻辑表达式是短路计算的，a++的值 0，即为假，则整个表达式 a ++ && b ++ 就可判断为假，后面的 b++不会被执行，因此 b 的值还是 30

(3)

```
char c = 'a' + 4;
cout << c << ″,″ << (int) c + 3 << endl;
```

解答：e,104

(4)

```
int a = 0,b = 10, c;
c = a++;
c = ++ b;
cout << a << ″,″ << b << ″,″ << c << endl;
```

解答：1,11,11

(5)

```
int a = 0,b = 10;
bool c = ( a == b);
cout << c << endl;
```

解答：0

第三章

1. 编写程序，每读入 3 个整数，就将他们从大到小排序输出。读到连续的 3 个 0，则程序结束。

输入样例：

3 4 5

7 2 9

0 0 0

输出样例：

5 4 3

9 7 2

解答：

```
#include <iostream>
using namespace std;
int main()
{
    int a,b,c;
    while(1) {
        cin >>a >> b >>c;
        if( a == 0 &&b == 0 &&c == 0)
            break;
        if( a >= b && b >=c )
            cout << a << " " << b << " " << c <<endl;
        else if( a >= c && c >= b)
            cout << a << " " << c << " " << b <<endl;
        else if( b >= a && a >= c)
            cout << b << " " << a << " " << c <<endl;
        else if( b >= c && c >= a)
            cout << b << " " << c << " " << a <<endl;
        else if( c >= a && a >= b)
            cout << c << " " << a << " " << b <<endl;
        else
            cout << c << " " << b << " " << a <<endl;
    }
    return 0;
}
```

2. 编写程序，输入一个整数 n,则输出一个由 n 行"*"构成的等腰三角形，第一行有一个"*"，第二行有 3 个"*"，第三行有 5 个"*"……每行比上一行多 2 个"*"。最后一行不能有空格。

输入样例：

4

输出样例：

```
   *
```

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

解答：

```
#include <iostream>
using namespace std;
int main()
{
   int n;
   cin >> n;
   int m = 2 * (n-1) + 1;
   for(int i = 0;i < n; ++i) {
       int k = 2 * i + 1;

       for(int j = 0; j < (m - k)/2; ++j )
           cout << " ";
       for(int j = 0;j < k; ++j )
           cout << "*";
       for(int j = 0; j < (m - k)/2; ++j )
           cout << " ";
       cout <<endl;
   }
   return 0;
}
```

4. 斐波那契数列第一项和第二项都是 1,此后各项满足：$F_n=F_{n-1}+F_{n-2}$。编写程序，输入整数 n，输出斐波那契数列第 n 项。

解答：

```
#include <iostream>
using namespace std;
int main()
{
   int n;
   cin >> n;
   int a1 = 1,a2 = 1;
```

```
    for(int i = 0;i < n-2; ++i) {
        int tmp = a1+a2;
        a1 = a2;
        a2 = tmp;
    }
    cout << a2;
    return 0;
}
```

5. 已知今天是星期二，问 n 天后是星期几(n>=0)。程序输入 n,输出“Monday"、"Tuesday"、"Wednesday"、"Thursday"、"Friday"、"Saturday"或"Sunday"。

解答：

```
#include <iostream>
using namespace std;
int main()
{
   int n;
   cin >>n;
   switch( n % 7 ) {
       case 0:
           cout << "Tuesday";
           break;
       case 1:
           cout << "Wednesday";
           break;
       case 2:
           cout << "Thursday";
           break;
       case 3:
           cout << "Friday";
           break;
       case 4:
           cout << "Saturday";
           break;
       case 5:
           cout << "Sunday";
           break;
```

```
        case 6:
            cout << "Monday";
            break;
    }
    return 0;
}
```

6. 编写程序，输入两个正整数，输出他们的最小公倍数。

解答:

```
#include <iostream>
using namespace std;
int main()
{
    int s,g;
    cin >> s >>g;
    if( s > g) {
        int tmp = s;
        s = g;
        g = tmp;
    }
    int ans = g;
    while( ans % s != 0)
        ans += g;
    cout << ans ;
    return 0;
}
```

7. 编写程序，输入两个正整数，输出他们的最大公约数。

解答：

```
#include <iostream>
using namespace std;
int main()
{
    int s,g;
    cin >> s >>g;
```

```
    if( s > g) {
        int tmp = s;
        s = g;
        g = tmp;
    }
    int i ;
    for( i = s;i >= 1 ; --i)
        if( s % i == 0 && g % i == 0)
            break;
    cout << i;
    return 0;
}
```

8. 计算有两个运算符的算术表达式的值。输入数据第一行是表达式的个数 n，后面每行是一个表达式。表达式中没有括号，只有数字和“+”、“-”、“*”、“/”。对每个表达式，输出一行，即计算的结果。结果要精确到小数点后面 6 位。输入数据保证除数不会为 0。

输入样例：

2

3+4*5

9/6+2

输出样例：

23.000000

3.500000

解答：

```
#include <iostream>
using namespace std;
int main()
{
    int n1,n2,n3;
    char op1,op2;
    double ans;
    int t;
    cin >>t;
    while(t--) {
        cin >> n1 >> op1 >> n2 >> op2 >> n3;
```

```
switch(op1) {
    case '+':
        switch(op2) {
            case '+':
                ans = n1+n2+n3;
                break;
            case '-':
                ans = n1+n2-n3;
                break;
            case '*':
                ans = n1+n2*n3;
                break;
            case '/':
                ans = n1+(double)n2/n3;
                break;
        }
        break;
    case '-':
        switch(op2) {
            case '+':
                ans = n1-n2+n3;
                break;
            case '-':
                ans = n1-n2-n3;
                break;
            case '*':
                ans = n1-n2*n3;
                break;
            case '/':
                ans = n1-(double)n2/n3;
                break;
        }
        break;
    case '*':
        switch(op2) {
            case '+':
                ans = n1*n2+n3;
```

```
                    break;
                case '-':
                    ans = n1*n2-n3;
                    break;
                case '*':
                    ans = n1*n2*n3;
                    break;
                case '/':
                    ans = n1*(double)n2/n3;
                    break;
            }
            break;
        case '/':
            switch(op2) {
                case '+':
                    ans = (double)n1/n2+n3;
                    break;
                case '-':
                    ans = (double)n1/n2-n3;
                    break;
                case '*':
                    ans = (double)n1/n2*n3;
                    break;
                case '/':
                    ans = (double)n1/n2/n3;
                    break;
            }           break;
        }
        cout << ans << endl;
    }
    return 0;
}
```

9. 输入一个不超过 6 位的整数，输出其倒过来后的结果，符号不变。例如，输入“1234”，则应输出“4321”，输入“-9876”，则应输出“-6789”。

解答：

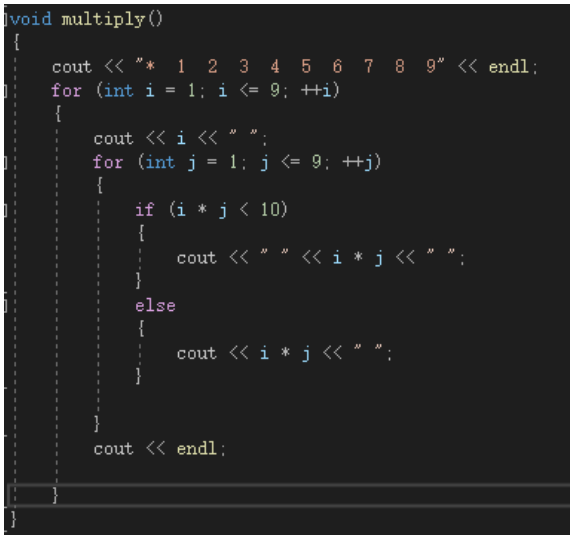
```
#include <iostream>
using namespace std;
int main()
{
   int n,m;

   cin >> n;
   if( n == 0)
       cout << "0";
   else {
       if( n < 0 ) {
           cout << "-";
           n = - n;
       }
       while( n > 0) {
           cout << n % 10 ;
           n /= 10;
       }
   }
   return 0;
}
```

10. 打印如下形式的乘法口诀表：

```
*  1  2  3  4  5  6  7  8  9
1  1  2  3  4  5  6  7  8  9
2  2  4  6  8 10 12 14 16 18
3  3  6  9 12 15 18 21 24 27
4  4  8 12 16 20 24 28 32 36
5  5 10 15 20 25 30 35 40 45
6  6 12 18 24 30 36 42 48 54
7  7 14 21 28 35 42 49 56 63
8  8 16 24 32 40 48 56 64 72
9  9 18 27 36 45 54 63 72 81
```

本代码运行结果是不对的，参考图片



解答：

```
#include <iostream>
using namespace std;
int main()
```

```
{
   for(int i = 1;i <= 9; ++i) {
       for(int j = 1; j <= 9; ++j ) {
           if( i == 1 && j == 1)
               cout << " * ";
           else {
               int n = i * j;
               if( n >= 10)
                   cout << n <<" ";
               else
                   cout << " " << n << " ";
           }
       }
       cout << endl;
   }
   return 0;
}
```

11. 输出所有“水仙花数”。“水仙花数”是一个 3 位正整数，其值等于各位数的立方和。例如 153 就是水仙花数，因为 $153=1^3+5^3+3^3$。

解答：

```
#include <iostream>
using namespace std;
int main()
{
   for(int i = 100;i <= 999; ++i) {
       int sum = 0;
       int n = i;
       for(int k = 0; k < 3; ++k) {
           int a = n % 10;
           sum += a * a * a;
           n /= 10;
       }
       if( sum == i)
           cout << i << endl;
```

```
    }
    return 0;
}
```

12. 输入一个二进制数（不超过 8 位），将其转换成一个十进制数输出。

解答：

```
#include <iostream>
using namespace std;
int main()
{
   int n;
   cin >> n;
   int sum = 0;
   int base = 1;
   while( n > 0) {
       sum +=  (n % 10) * base;
       base *= 2;
       n /= 10;
   }
   cout << sum ;
   return 0;
}
```

13. 井底的蜗牛距离井口 m 米。白天蜗牛向上爬若干米，晚上蜗牛滑下来 1 米。每天白天向上爬的距离是昨天的 1/3。已知第一天白天能向上爬 n 米，问几天后蜗牛能爬出井外（到井口就出去了）。输入数据是两个整数，m 和 n,输出第几天爬出井口。如果永远爬不出，则输出“Never”。

解答：

```
#include <iostream>
using namespace std;
int main()
{
   int m,n;
   while(cin >> m >> n) {
       if( n >= m)
           cout << 1 <<endl;
```

```
        else {
            double height = 0;
            double dist = n;
            int days = 0;
            while(true) {
                ++days;
                height += dist;
                if( height - m >= -(1e-6)) {
                    cout << days << endl;
                    break;
                }
                height -= 1;
                dist /= 3;
                if( dist - 1 <= -(1e-6)) {
                    cout << "Never" << endl;
                    break;
                }
            }
        }
    }
    return 0;
}
```

14. 编程输出 100 内所有 5 的倍数。

解答：

```
#include <iostream>
using namespace std;
int main()
{
   for(int i = 5; i <= 100; i+= 5)
       if( i % 5 == 0)
           cout << i << endl;
   return 0;
}
```

第四章

1. 编写一个接收三个 int 类型参数的函数，返回三个参数中的最大值。

解答：

```
int max(int a,int b,int c)
{
   int mx = a;
   if( mx < b)
       mx = b;
   if( mx < c)
       mx = c;
   return mx;
}
```

2. 参数的传值和传引用有什么区别？

解答：如果参数是传值的，则形参是实参的拷贝，在函数内部修改了形参，实参也不会改变。如果参数是传引用的，则形参是实参的引用，在函数内部修改了形参，实参也会改变。

3. 什么情况下函数调用的返回值可以作为左值？

解答：函数返回值为引用的时候，其返回值可以作为左值。

4. inline 函数有什么优点？

解答：调用速度快于非 inline 函数。

5. 如果有多个重载的函数，编译器根据什么来判断调用这些函数的语句，到底调用的是哪一个？

解答：根据实参的个数和类型。

6. 编写一个函数，可用于交换两个 double 类型实参的值。

解答：

```
void swap(double & a,double & b) {
    double tmp = a;
    a = b;
    b = tmp;
```

}

7. 以下有关函数的叙述中,正确的是:

A)函数必须返回一个值

B)函数体中必须有 return 语句

C)两个同名函数，参数表相同而返回值不同不算重载

D)函数执行中形参的改变会影响到实参

解答：C

8. 函数的原型中可以不指出：

A) 函数的返回值类型

B) 函数的参数个数

C) 函数的参数的名字

D) 函数的名字

B对

解答：C

9. 以下正确的函数声明形式是：

A) int func(int x,int y)

B) int func(int ,int);

C) int func(int x;int y);

D) int func(int x,y);

明显A对

解答：B

10. 以下说法不正确的是：

A) 不同函数中可以使用相同名字的变量

B) 函数的实参和形参不能同名

C) 函数内也可以没有 return 语句

D) 函数形参改变，实参也可能改变

解答：B

第五章

1. 以下对数组 a 不正确的定义是：

A) int a[10];

B) int a[];

C) const int n = 5; double a[n];

D) double a[3 * 5];

解答：B

2. 有数组 double a[10];假设 a[0]的地址是 n,那么 a[i]的地址是_____, sizeof(a)等于_____

解答： n + i * 8 , 80

3. 有数组 int a[4][20];假设 a[0][0]的地址是 n,那么 a[i][j]的地址是_____,sizeof(a)等于_____

解答： n + i * 20 + 4 * j, 320

4. 下面程序片段的输出结果是 _____

```
int a[6];
for( int i = 1; i < 6;++i ) {
    a[i] = 8 * ( i + 2 * ( i > 3)) %5;
    cout << a[i] << " " ;
}
```

解答：3 1 4 3 1

5. 下面程序片段的输出结果是 _____

```
int a[][3] = { { 2,1},{3,4}};
cout << sizeof(a) << endl;
for( int i = 0;i < 2 ; ++i ) {
    for( int j = 0; j < 3; ++j)
        cout << a[i][j] << " ";
    cout << endl;
}
```

解答:

24

2 1 0

3 4 0

6. 编程求两个矩阵相乘的结果。输入第一行是整数 m,n，表示第一个矩阵是 m 行 n 列的。接下来时一个 m×n 的矩阵。再下一行的输入是整数 p,q，表示下一个矩阵是 p 行 q 列的（n=p)。再接下来就是一个 p 行 q 列的矩阵。要求输出两个矩阵相乘的结果矩阵(1 < m,n,p,q <= 8)。

输入样例：

2 3

2 4 5

2 1 3

3 3

1 1 1

2 3 2

0 1 4

输出样例：

10 19 30

4 8 16

解答：

```
#include <iostream>
using namespace std;
int a[10][10],b[10][10], c[10][10];
int main()
{
   int m,n,p,q;
   cin >> m >> n;

   for(int i = 0;i < m; ++i)
       for(int j = 0; j < n; ++j)
           cin >> a[i][j];
   cin >> p >> q;
   for(int i = 0;i < p; ++i)
       for(int j = 0; j < q; ++j)
```

```
            cin >> b[i][j];

    for(int i = 0;i < m; ++i) {
        for(int j = 0;j < q; ++j) {
            c[i][j] = 0;
            for(int k = 0; k < n; ++k)
                c[i][j] += a[i][k] * b[k][j];
        }
    }
    for(int i = 0; i < m; ++i) {
        for(int j = 0; j < q; ++j )
            cout << c[i][j] << " ";
        cout << endl;
    }
    return 0;
}
```

7. 打印以下杨辉三角形，要求输出到第 10 行:

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
.....

该三角形的特点是：第 n 行有 n 个数字，每个数字是上方和左上方的数字之和。

解答：

```
#include <iostream>
using namespace std;
int a[20];
int main()
{
    a[0] = 0;
    for(int i = 1;i <= 10; ++i) {
        for(int j = i - 1; j >= 1; --j)
```

```
            a[j] = a[j] + a[j-1];
        a[i] = 1;
        for(int j = 1; j <= i; ++j)
            cout << a[j] << " ";
        cout << endl;
    }
    return 0;
}
```

8. 已知 2012 年 1 月 25 日是星期三，编写一个程序，输入用“年 月 日”表示的一个日期，输出该日期是星期几。

解答：下面的程序，输出 0 代表是星期天。

```
#include <iostream>
using namespace std;
int a[20];
int main()
{
   int monthDays[] = {-1,31,28,31,30,31,30,31,31,30,31,30,31};
   int days = 0 ;//从 2012-01-22 开始过了多少天   2012-01-22 是星期天
   int year,month,date;
   cin >> year >> month >> date;

   for (int y = 2012; y < year; ++y ) {
      if  (y%4 ==0 && y%100!= 0 || y%400 == 0)    //闰年
         days += 366;
      else
         days += 365;
   }
   if (year%4 ==0 && year%100!= 0 || year%400 == 0)
      monthDays[2] = 29;
   for (int i = 1; i < month; ++i)
      days += monthDays[i];
   days += date;
   days -= 22;  //2012 年 1 月 22 日是星期天
   cout << days % 7 ;  //星期天算一周的第 0 天
```

```
    return 0;
}
```

第六章

1. 下面对 s 的初始化正确的是：

A) char s[5] = "abcde";

B) char s = "abcd";

C) string s = "abcd";

D) string s= 'ab';

解答: C

2. 对两个数组 a,b 如下初始化：

```
char a[] = "abcd";
char b[]={'a','b','c','d'};
```

这两个数组完全相同吗？为什么？

解答：不一样。第一个数组会有 5 个元素，包括字符串结尾的'\0'。第二个数组只有 4 个元素。

3. 写出下列程序片段的输出结果:

(1)
```
char s[] = { 'a','b','\0','c','\0'};
cout << s ;
```

解答：ab

(2)
```
char s1[] = "abcdef";
char s2[] = "123";
strcpy(s1,s2);
int n = s1[3];
cout << n;
```

解答：0

(3)
```
string s1 = "123";
s1 += "abc";
```

```
char s3[10];
strcpy( s3,s1.c_str());
cout << s3 ;
```

解答：123abc

(4)
```
char cities[3][30] = { "Beijing",
   "Shanghai", "London" };
cout << cities[2] << endl;
cout << cities[1][3] << endl;
```

解答：

London

n

4. 下面的程序片段输出结果是：

Beijing Shanghai Chengdu

请填空:

```
_____{ "Beijing","Shanghai",
        "Chengdu"};
for( int i = 0;i < 3; ++i )
    cout << s[i] << " ";
```

解答：string s[] =

5. 下面的程序片段，输入一个字符串，则输出相同的字符串,请填空：

```
string s1;   cin >> s1;   char s[20];
_____;
cout << s << endl;
```

解答：strcpy(s,s1.c_str())

6. 下面的程序片段，输入两个字符串，输出这两个字符串连接在一起后的字符串。例如，输入: Hello world ↙ 输出：Helloworld 请填空:

```
char s1[20],s2[20],s3[40];
cin >> s1 >> s2;
```

```
strcpy( s3,s1);
_____;
cout << s3 << endl;
```

解答：strcat(s3,s2);

7. 输入一个十进制数，将其转换成二进制数输出。

解答：

```
#include <iostream>
using namespace std;
int main()
{
 int a[100];
 int n;
 cin >> n;
 int i = 0;
 while(n) {
        a[i++] = n%2;
        n/=2;
 }
 for(int k = i-1;k >=0 ; --k)
        cout << a[k] ;
 return 0;
}
```

8. 输入两个字符串，判断第二个串是不是第一个串的子串。如果是，输出 Yes，否则输出 No。

```
#include <iostream>
#include <cstring>
using namespace std;
char s1[1000],s2[1000];
int main()
{
 cin >> s1 >> s2;
 int L1 = strlen(s1);
 int L2 = strlen(s2);
 for(int i = 0;i < L1; ++i) { //i 是比较的起点
        int k = i,j=0;
```

```
        for(; j < L2; ++j) {
            if( s1[k] == s2[j] )
                ++k;
            else
                break;
        }
        if( j == L2) {
            cout << "yes" << endl;
            return 0;
        }
    }
    cout << "no" <<endl;
    return 0;
}
```

9. 输入一个字符串，求其不含重复字符的最长的子串的长度。

解答 1:

```
#include <iostream>
#include <cstring>
using namespace std;
char s[1000];
int main()
{
    cin >> s ;
    int len = strlen(s);
    int maxLen = 1;
    for(int i = 0;i < len; ++i) { //枚举子串开始位置
        int k = i + 1;
        int tmpLen = 0;
        for(; k < len; ++k) { //从 i 开始往后找子串
            int n = i;
            for(; n < k; ++n) {
                if( s[n] == s[k])
                    break;
            }
            if( n == k) {
```

```
                if( maxLen < k-i+1)
                        maxLen = k-i+1;
                }
                else {//s[k]和 s[i]到 s[k-1]中间的字符重复，即和 s[n]  重复
                        if(n>i)
                                i = n;   //下个子串起点是 n+1 就可以了
                        break;
                }
        }
 }
 cout << maxLen;
 return 0;
}
```

解答 2:

```
#include <iostream>
#include <cstring>
using namespace std;
char s[1000];
int maxLen[1000]; //maxLen[i]表示以 s[i]为终点的最长不重复子串长度
int main()
{
 cin >> s ;
 int len = strlen(s);
 for(int i = 0;i < 1000; ++i)
        maxLen[i] = 1;
 for(int i = 1;i < len; ++i ) {
        int k = i - 1;
        for(; k >= i-maxLen[i-1];--k){
                if( s[i] == s[k] ) {
                        maxLen[i] = i - k;
                        break;
                }
        }
        if( k < i-maxLen[i-1]) //没有碰到和  s[i]重复的
                maxLen[i] = maxLen[i-1] + 1;
 }
```

```
 int maxL = 0;
 for(int i = 0;i < len; ++i)
        if( maxL < maxLen[i])
              maxL = maxLen[i];
 cout << maxL;
 return 0;
}
```

10. 输入两个字符串，判断第二个串在第一个串里出现了几次。比如，"aa"在串"aaaa"里应该算出现了 3 次。

解答:

```
#include <iostream>
#include <cstring>
using namespace std;
char s1[1000],s2[1000];
int main()
{
 cin >> s1 >> s2 ;
 int len1 = strlen(s1);
 int len2 = strlen(s2);
 if( len2 > len1) {
        cout << 0<< endl;
        return 0;
 }
 int times = 0;
 for(int i = 0;i < len1; ++i) {
        int j = 0;
        for(; j < len2; ++j) {
              if( s1[i+j] != s2[j])
                    break;
        }
        if( j == len2)
              ++times;
 }
 cout << times;
 return 0;
}
```

11. 输入一个串，求其最长上升子串的长度。上升串就是字符的 ASCII 码递增的字符串，比如"ades","abcd"。

"fghia"就不是上升串。

解答：

```
#include <iostream>
#include <cstring>
using namespace std;
char s[1000];
int maxLen[1000]; //maxLen[i]表示以 s[i]为终点的最长上升子串长度
int main()
{
 cin >> s ;
 int len = strlen(s);
 for(int i = 0;i < 1000; ++i)
        maxLen[i] = 1;
 for(int i = 1;i < len; ++i )
        if( s[i] > s[i-1])
                maxLen[i] = maxLen[i-1] + 1;
 int maxL = 0;
 for(int i = 0;i < len; ++i)
        if( maxL < maxLen[i])
                maxL = maxLen[i];
 cout << maxL;
 return 0;
}
```

12. 输入两个字符串，判断第二个串是不是第一个串的子序列。子序列和子串的差别在于子序列可以不连续，比如"eck"是"eacdbk"的子序列，但不是其子串。

解答:

```
#include <iostream>
#include <cstring>
using namespace std;
char s1[1000],s2[1000];
bool IsSubSeq(char s1[],char s2[])
{
 if( s2[0] == 0)
        return true;
```

```
 if( s1[0] == 0)
        return false;
 bool b1 = false,b2;
 if( s1[0] == s2[0])
        b1 =    IsSubSeq(s1+1,s2+1);
 b2 = IsSubSeq(s1+1,s2);
 return b1 || b2;
}
int main()
{
 cin >> s1 >> s2 ;
 cout << IsSubSeq(s1,s2);
 return 0;
}
```

第七章

1. 写出下面程序片段的输出结果

(1) char * p = "This";
cout << sizeof(p) << "," << *p << endl;
++p; cout << p[2] ;
解答：
4,T
s

(2) char s[] = "Hello,world";
char * p = s + 3;
string str = p; cout << str << endl;

解答:
lo,world

(3) int a[10][20];
int * p1 = a[2]; int * p2 = a[4] +2;
cout << p2 - p1 << endl;
解答:
42

(4) char s[4][20] =
{ "pig","dog","cat","sheep"};
char * p = s[1]; cout << * p << endl;
解答:
d

(5) char * p[3] =
{ "Toyota","Honda","BMW", };
char ** pp = p + 2; cout << p[2] ;
解答:
BMW

2. 以下初始化语句哪个是正确的？

A) string * p = "this";
B) string p[] = "that";
C) string p[] = { "What","this" };
D) char * p = { "Please" };
解答:C

3. 以下程序片段哪个是正确的?

A) char a[20]; char * p = & a[10];

B) int a[4]; int * p = a[5];

C) char c = 'a'; int * p = & c;

D) char * p1 = NULL ,* p2; p2 = p1[4];

解答:A

4. 下面程序片段的输出结果是 Hello ,请填空

```
char s[] = "Hello"; char * p;
for(_____________________)
    cout << * p ;
```

解答:p=s;*p;++p

5. 下面程序片段的输出结果是 kkkkk ,请填空。不能使用 strcpy 和 strncpy 函数。

```
    char s[6] = "12345";
    _____________________;
    cout << s;
```

解答: memset(s,'k',5);

6. 下面程序输出结果是 Lexus,请填空

```
#include <iostream>
using namespace std;
void Print(char * p1, char * p2)
{  for( ; _____; _____)    cout << * p1;}
int main()  {
    char * s = "Lexus";
    Print(s,s+5);
    return 0;
}
```

解答:

p1<p2

++p1

7. 编写一个 MyStrstr 函数,实现和 strstr 完全一样的功能。

解答:

```
char * MyStrstr(char * s1,char * s2)
```

```
{
    for(int i = 0; s1[i]; ++i) {
        int j = 0;
        for(; s2[j] ; ++j) {
            if( s1[i+j] != s2[j])
                break;
        }
            if( s2[j] == 0)
                return s1+i;
    }
    return NULL;
}
```

8. 编写一个 MySort 函数，实现和 qsort 完全一样的功能，具体排序的算法不限。

```
void  MySort(const void * a, int n,int size, int (*cmp)(const void * ,const void *))
{
    char * adr = (char *)a;
    for(int d = n-2; d >=0; --d) {   //冒泡排序
        for(int i = 0;i <= d; ++i) {
            char * p1 = adr + i * size;
            char * p2 = adr + i * size + size;
            int result = cmp(p1,p2) ;
            if( result > 0 ) { //交换两个元素
                for(int k = 0;k < size; ++k) {
                    char tmp = p1[k];
                    p1[k] = p2[k];
                    p2[k] = tmp;
                }
            }
        }
    }
}
```

第八章

1. 请写出下面程序片段的输出结果：

(1)
```
struct Student{
        int age;
        char name[20];
        double gpa;
};
cout << sizeof(Student);
```

解答: 32

(2)
```
struct Student{
        int age;
        char * name;
        double gpa;
};
Student s;
s.name = new char[20];
cout << sizeof(Student);
```

解答：16

(3)
```
enum Colors { Red ,Blue = 6,
Black,Green,Yellow = 3,Purple };
cout << Red << "," << Green << ","
   << Purple ;
```

解答: 0,8,4

(4)
```
union IP {
        unsigned int ip;
        unsigned char seg[4];
};
IP ipadr;
ipadr.ip = 0x12131415;
cout << (int)ipadr.seg[3] << ","
     << (int)ipadr.seg[0] ;
```

解答: 18,21

2. 下面程序的输出结果是 5 ，请填空：

```
#include <iostream>
using namespace std;
int Max(int a,int b) {
    return a>b?a:b;
}
int main()  {
    ____________________________;
    PFUN p = Max;
    cout << p(5,4);
    return 0;
}
```

解答: typedef int (*PFUN) (int ,int)

3. 下面的程序片段，输入一个字符串，则输出相同的字符串，请填空：

```
    struct Employee {
        string name;
        int age;
        int salary;
    };
    Employee * p;
    ____________________;
    cin >> p->name ;
    cout << p->name ;
```

解答：p = new Employee()

4. 若有以下定义语句：

```
  struct Employee {
    string name;
    int age;
    int salary;
  };
  Employee e, * p = & e;
```

则以下对 e 中的成员变量 name 的引用，不正确的是：

A) p.name

B) p->name

C) (*p).name

D) *p.name

解答: A

5. 若有以下定义语句：

```
struct Employee {
  char name[20];
  int age;
  int salary;
};
Employee e[5], * p = e;
```

则以下对结构变量成员引用方式不正确的是：

A) e[0].name

B) p->age

C) p[1]->age

D) int * p = & (p->age)

解答：C

6. 编写一个结构，用于表示一辆汽车。一辆汽车的信息包括：品牌、发动机排量、价格、车主姓名、颜色。

解答：

```
struct Car
{
    string brand;
    double engineVolume;
    int price;
    string owner;
    int color;
};
```

7. 请写出能表示 ip 地址的联合的定义，并说明每个成员变量对应于 ip 地址的哪一段。

解答：

```
union IpAddress
{
    unsigned int adr;
    unsigned char seg[4];
};
```

若 ip 地址是：192.168.0.1，则:

adr 的值是: 0x100a8c0

ip.seg[0]=192

ip.seg[1]=168

ip.seg[2]=0

ip.seg[3]=1

第九章

1. 给定四个整数 m,n,p,q，求大于 m 的最小的 n,p,q 的公倍数。

解答：

```
#include <iostream>
using namespace std;
int gcd(int a,int b)
{ //求最大公约数
    if( b == 0)
        return a;
    else
        return gcd(b,a%b);
  }
int main()
{
    int m,n,p,q;
    cin >> m >> n >> p >> q;
    int b = n * p / gcd(n,p); //b 是 n,p 的最小公倍数
    int b2 = b * q / gcd(b,q);
    int x = m / b2;
    cout << (x + 1) * b2;
    return 0;
}
```

2. 九宫图问题。将 1-9 这九个数字填入 3×3 的方格,使得每一行，每一列，每条对角线上的数字之和都相等。编程求该问题的一个解。

解答：

```
#include <iostream>
using namespace std;
int d[10];
```

```
int used[10] ={0};
bool judge()
{
int sum1 = d[0] + d[1] + d[2];
int sum2 = d[3] + d[4] + d[5];
int sum3 = d[6] + d[7] + d[8];
if( sum1 != sum2 || sum1 != sum3 || sum2 != sum3)
    return false;
int sum0 = sum1;
sum1 = d[0] + d[3] + d[6];
sum2 = d[1] + d[4] + d[7];
sum3 = d[2] + d[5] + d[8];
if( sum0 != sum1 || sum2 != sum0 || sum3 != sum0)
    return false;
sum1 = d[0] + d[4] + d[8];
sum2 = d[2] + d[4] + d[6];
if( sum1 != sum0 || sum2 != sum0)
    return false;
return true;
}
bool put(int n)
//已经放好了第 0 个到第 n-1 个数，现在第 n 个以及以后的数，返回值为 true 表示找到解
{
if( n == 9) {
    if( judge()) {
        for(int i = 0;i < 3; ++i) {
            for(int j = 0; j < 3; ++j) {
                cout << d[i*3+j] << " ";
            }
            cout << endl;
        }
        return true;
    }
    return false;
}
for(int i = 1;i <= 9; ++i ) {
    if( ! used[i] ) {
```

```
            used[i] = 1;
            d[n] = i;
            if( put(n+1))
                return true;
            else
                used[i] = 0;
        }
    }
    return false;
    }
    int main()
    {
        put(0);
        return 0;
    }
```

3. 给定 n 个不同的字母（n<10）,按字典序输出其全部排列。比如，给定三个字母 A,B,C，则应输出：

ABC,ACB,BAC,BCA,CAB,CBA

解答：

```
//全排列
#include <iostream>
#include <algorithm>
#include <cstring>
#include <string>
using namespace std;
const int M = 11;
char str[M];
char permutation[M];
bool used[M] = {0};
int L = 0;
void Permutation(int n)
{
if( n == L ) {
    permutation[L] = 0;
    cout << permutation << endl;
    return ;
}
```

```
for(int i = 0;i < L; ++i) {
    if( !used[i]) {
        used[i] = true;
        permutation[n] = str[i];
        Permutation(n+1);
        used[i] = false;
    }
}
}
int main()
{
cin >> str; //假设输入形式是：cbacef 这样
L = strlen(str);
sort(str,str+L); //排序
Permutation(0);
return 0;
}
```

4. 编写不用递归的分苹果程序。

解答：

```
#include <iostream>
#include <cstring>
using namespace std;
int ways[20][20];
//ways[i][j]表示 i 个苹果放在 j 个盘子里的放法数目
int main()
{
    int t,m,n;
    cin >> t; //测试数据组数
    while(t--)  {
        cin >> m >> n; //苹果数 m 和盘子数 n，假定都 <= 10
        memset(ways,0,sizeof(ways));
        for(int i = 0;i <= n; ++i)
            ways[0][i] = 1;
        for(int i = 1;i <= m; ++i)
            for(int j = 1; j <= n; ++j)
                if( j > i )
                    ways[i][j] = ways[i][i];
```

```
                else
                    ways[i][j] = ways[i][j-1] + ways[i-j][j];
        cout << ways[m][n] << endl;

    }
    return 0;
}
```

5. 共有 n 级台阶，上台阶时可以每步走一级、二级或三级，编程求一共有多少种走法。

解法 1(慢)：

```
#include <iostream>
using namespace std;
int steps(int n)
{
    if( n < 0)
        return 0;
    else if( n == 0)
        return 1;
    else
        return steps(n-1) + steps(n-2) + steps(n-3);
}
int main()
{
    int n;
    cin >> n;
    cout << steps(n);
    return 0;
}
```

解法 2：

```
#include <iostream>
using namespace std;
int main()
{
    int n;
    cin >> n;
    int w[3] = {1,1,2};
    //0 级台阶 1 种走法，1 级台阶 1 种走法，2 级台阶 2 种走法
```

```
    if( n <= 2)
        cout << w[n];
    int ans;
    for(int i = 3; i <= n; ++i) {
        ans = w[0] + w[1] + w[2];
        w[0] = w[1];
        w[1] = w[2];
        w[2] = ans;
    }
    cout << ans ;
    return 0;
}
```

6. 编程求方程 $2^x+3x-7=0$ 的一个解。

解答：

```
#include <iostream>
#include <cmath>
using namespace std;
double value(double x)
{
    return pow(2,x) + 3*x - 7;
}
int main()
{
    const double eps = 1e-6;
    double L = 1,R = 2;
    double x = (L+R)/2;
    double v ;
    while(true) {
        v = value(x);
        if( fabs(v) < eps)
            break;
        if( v > 0)
            R = x;
        else
            L = x;
        x = (L+R)/2;
    }
```

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

第十章

1. 写出下面程序片段的输出结果：

(1)
```
int n ;
int main()  {
      n = 5;
      int n = 12;
      cout << n << endl;
      cout << ::n << endl;
      return 0;
}
```
解答：

12

5

(2)
```
void Func() {
    static int a = 12;
    cout << a++ << endl;
}
int main()  {
      for( int i = 0;i < 3; ++ i )
          Func();
      return 0;
}
```
解答：

12

13

14

(3)
```
#define MAX(x,y) x * y
cout << MAX(2+3,3+4);
```

解答：

15
分析：等价于 2+3*3+4

(4) int main() {
```
     #define DEBUG
     #ifdef DEBUG
     #undef DEBUG
         cout << "debuging " << endl;
     #endif
         cout << "running" << endl;
    #ifdef DEBUG
         cout << "step1" << endl;
    #else
         cout << "step2" << endl;
     #endif
     return 0;
  }
```
解答:
debuging
running
step2

2. 编写一个程序 sort.exe，由可命令行输入若干个单词，程序将这些单词排序后输出。例如：
sort bike about take fan
程序输出结果是：
about bike fan take

解答:
```
#include <iostream>
#include <cstdlib>
#include <cstring>
using namespace std;
int Cmp(const void * e1,const void * e2)
{
 char ** p1 = (char **)e1;
 char ** p2 = (char **)e2;
 return strcmp(*p1,*p2);
}
```

```
int main(int argc, char * argv[])  {
     qsort(argv+1,argc-1,sizeof(char*),Cmp);
     for(int i = 1;i < argc; ++i)
          cout << argv[i] << " ";
     return 0;
}
```

3. 编写一个由两个文件组成的C++程序,一个文件里定义的函数和全局变量,在另一个文件里使用。

解答:

File1.cpp:

```
int a = 10;
int mysum(int x,int y)
{
     return x + y;
}
```

File2.cpp:

```
#include <iostream>
using namespace std;
extern int a;
int mysum(int x,int y);
int main()
{
     cout << mysum(a,4);
     return 0;
}
```

《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
*/
```