曾经在软通动力写过的算法题其中有一小部分是参考网上的资料，现在拿出来给大家分享！   
第3章 控制语句   
  
/\* 1、打印出所有的“水仙花数”。所谓“水仙花数”是指一个三位数，其各位数字立方和等于该数本身。例如，153是一水仙花数，因为153 = 1^3 + 5^3 + 3^3。 \*/   
#include<iostream.h>   
void main()   
{ 

int i, a=0, b=0, c=0;

for(i=100;i<1000;i++)

{

a=i%10;

b=i/10%10;

c=i/100%10;

if(a\*a\*a+b\*b\*b+c\*c\*c==i)

cout<<"i="<<i<<endl;

}

}   
  
/\* 2、一个数如果恰好等于它的因子之和，这个数就称为“完数”。例如，6的因子为1、2、3，而6 = 1 + 2 + 3，因此6是“完数”。编程序找出1000之内的所有完数，并按下面的格式输出其因子：   
6 -〉1，2，3 \*/   
#include<iostream.h>   
void main()   
{ 

int i,j,sum=0,a[50],k,t;

for(i=1;i<=1000;i++)

{

sum=0;

for(j=1;j<i;j++)

{

if(i%j==0)

{

sum+=j;

a[k++]=j;

}

t=k;

if(sum==i)

{

cout<<i<<"->";

for(k=0;k<t;k++)

{

    cout<<a[k];

    if(k<t-1)cout<<",";

}

}

cout<<endl;

}

k=0;

}

}   
  
/\* 3、求Sn=a+aa+aaa+…+aa…a之值，其中a是一个数字。例如：2+22+222+…+22222（此时n=5），n由键盘输入。\*/   
#include<iostream.h>   
void main()   
{   
double a,sn=0.0,sum=0.0;   
int n,i;   
cout<<"please input a number";   
cin>>a;   
    cout<<"please input n number";   
cin>>n;   
sn=a;   
sum=a;   
    for(i=2;i<=n;i++)   
    {   
sum=sum\*10+a;   
sn+=sum;   
    }   
cout<<"Sn="<<sn<<endl;   
}   
  
/\* 4、一球从100米高度自由落下，每次落地后反跳回原高度的一半，再落下。求它在第10次落地时，共经过了多少米？第10次反弹多高？\*/   
#include<iostream.h>   
void main()   
{ 

double h1=100,h2=100,sum=0.0;

int i;

for(i=1;i<=10;i++)

{

sum+=h2;

h1=h1/2.0;

h2=h1\*2;

}

cout<<"sum="<<sum<<"   "<<"h1="<<h1<<endl;

}   
  
/\* 5、猴子吃桃问题。猴子第一天摘下若干个桃子，当即吃了一半，还不过瘾，又多吃了一个。第二天早上又将剩下的桃子吃掉了一半，又多吃了一个。以后每天早上都吃了前一天剩下的一半零一个。到第10天早上想再吃时，见只剩一个桃子了。求第一天共摘了多少桃子。\*/   
#include<iostream.h>   
void main()   
{ 

int number,i;

number=1;

for(i=10;i>1;i--)

number=(number+1)\*2;

cout<<"number="<<number<<endl;

}   
  
8.＋＋程序中使用流格式输入、输出，我们可以怎么做？   
答：在程序的开头包含头文件iostream.h   
     cin输入，cout输出。   
例如：   
  
#include<iostream.h>   
void main()   
{   
   int a;   
   cout<<"请输入a的值：";   
   cin>>a;   
   cout<<"a的值为："<<a<<endl;   
}   
  
  
第4章 函数   
  
/\* 1、写一函数用“气泡法”对输入的10个字符按由小到大的顺序排列。\*/   
#include<iostream.h>   
void main()   
{ 

int i,j,temp,a[10];

cout<<"please input ten numbers:\n";

for(i=0;i<10;i++)

cin>>a[i];

for(i=0;i<10;i++) //每循环一次确定数组中一个数的位置   
for(j=i+1;j<10;j++) //

for(j=i+1;j<10;j++) //每次循环对比一个数的大小   
{   
if(a[i]>a[j])   
{ 

{

if(a[i]>a[j])

{

temp=a[j];  
a[j]=a[i];

a[i]=temp;

}

}

cout<<"resort result=";

for(i=0;i<10;i++)

cout<<a[i]<<" ";

}   
  
/\* 2、用递归方法求n阶勒让得多项式的值，递归公式为   
         1 (n = 0)   
Pn(x) =  x (n = 1)   
         ((2n-1)\*x\*Pn-1(x)-(n-1)\*Pn-2(x))/n (n > 1) \*/   
#include<iostream.h>   
double fun (double,double);   
void main()   
{   
double n,x,sum;   
cout<<"input n and x"<<endl;   
cin>>n>>x;   
sum=fun(n,x);   
cout<<"P"<<n<<"("<<x<<")"<<"="<<sum<<endl;   
}   
double fun(double n1,double x1)   
{   
if (n1==0)   
return 1;   
else if (n1==1)   
return  x1;   
else if (n1>1)   
return ((2\*n1-1)\*x1\*fun(n1-1,x1)-(n1-1)\*fun(n1-2,x1))/n1;   
}   
  
/\* 3、编写一函数，由实参传来一字符串，统计此字符串中字母、数字、空格、和其它字符的个数，并在主函数中输入字符串以及输出上述结果。 \*/   
#include<iostream.h>   
void judge(char a[]);   
void main()   
{   
const int size=100;   
char a[size];   
cin.getline(a,size);   
judge(a);   
}   
void judge(char a[100])//判断字符类型   
{   
int letter=0,number=0,others=0,i=0;   
while(a[i]!='\0')   
{   
if ((a[i]>='a'&&a[i]<='z')||(a[i]>='A'&&a[i]<='z')) letter++;//统计字母个数   
else if (a[i]>='0' && a[i]<='9') number++;//统计数字个数   
else others++;//统计其他数个数   
i++;   
}   
cout<<"letter="<<letter<<"  number="<<number<<"  others="<<others<<endl;   
}   
  
/\* 4、给出年、月、日，计算该日是该年的第几天。 \*/   
#include<iostream.h>   
int lead(int);   
void main()   
{   
int ly,year,month,date,i,sum=0;   
cout<<"input year、month、date: ";   
cin>>year>>month>>date;   
int a[12]={31,0,31,30,31,30,31,31,30,31,30,31};   
ly=lead(year);   
if (ly==1)   
a[1]=29;//366天   
else a[1]=28;//365天   
for(i=0;i<month-1;i++) //当前月之前所有月天数累加和   
sum+=a[i];   
sum+=date; //加上当前月天数   
cout<<"你输入的日期是当年的第"<<sum<<"天";   
}   
int lead(int y)//判断闰年   
{   
if((y%4==0&&y%100!=0)||(y%400==0)) return 1;//是闰年   
else return 0;//不是闰年   
}   
  
/\* 5、写两个函数，分别求两个整数的最大公约数和最小公倍数，用主函数调用这两个函数，并输出结果，两个整数由键盘输入。 \*/   
#include<iostream.h>   
int cdivisor(int,int);   
int cmultiple(int,int,int);   
void main()   
{   
int x,y,d,m;   
cout<<"input two number: ";   
cin>>x>>y;   
d=cdivisor(x,y);   
m=cmultiple(x,y,d);   
cout<<"common divisor is "<<d<<endl<<"common multiple is "<<m<<endl;   
}   
int cdivisor(int x1,int y1)//最大公约数   
{   
int r,temp;   
if (x1<y1)   
{   
temp=x1;   
x1=y1;   
y1=temp;   
}   
while(x1%y1)//当较大数除以较小数余数等于0时，较小数为最大公约数   
{   
r=x1%y1;   
x1=y1;   
y1=r;   
}   
return y1;   
}   
int cmultiple(int x2,int y2,int d1)//最小公倍数   
{   
return x2\*y2/d1;//两数相乘结果除以它们的最大公约数为最小公倍数   
}   
  
/\* 6、写一函数，将两个字符串连接。 \*/   
#include<iostream.h>   
#include<string.h>   
void main()   
{   
const int size=100;   
char a[size],b[size];   
cout<<"input two string:"<<endl;   
cin.getline(a,size);   
cin.getline(b,size);   
strcat(a,b);   
cout<<"a="<<a<<endl;   
}   
  
/\* 7、写一函数，将一个字符串的元音字母复制到另一个字符串，然后输出。 \*/   
#include<iostream.h>   
#include<string.h>   
void scpy(char \*,char \*);   
void main()   
{   
const int size=100;   
char a[size]="Hello world";   
char b[size]="Net";   
cout<<"a= "<<a<<"b= "<<b<<endl;   
scpy(a,b);   
cout<<"a= "<<a<<endl;   
}   
void scpy(char \*p,char \*q)   
{   
while(\*q!='\0')   
{   
if (\*q=='a'||\*q=='A'||\*q=='e'||\*q=='E'||\*q=='i'||\*q=='I'||\*q=='o'||\*q=='O'||\*q=='u'||\*q=='U')   
\*p++=\*q;   
q++;   
}   
}   
  
/\* 8、写一函数，输入一个四位数字，要求输出这4个数字字符，但每两个数字间空一空格。如输入1990，应输出“1 9 9 0”。 \*/   
#include<iostream.h>   
#include<string.h>   
void outs(char a[]);   
void main()   
{   
const int size=10;   
char a[size];   
cin.getline(a,size);   
outs(a);   
}   
void outs(char a[10])   
{   
int i;   
if(strlen(a)<=4)   
{   
for(i=0;i<4;i++)   
cout<<a[i]<<" ";   
}   
else cout<<"input error."<<endl;   
}   
  
第5章   数组   
  
/\* 1、将一个数组中的值按逆序重新存放，例如，原来顺序为：a、b、c、d。要求改为：d、c、b、a。 \*/   
#include<iostream.h>   
void back(char \*);   
void main()   
{   
char a[50]="abcdefg";   
cout<<"a="<<a<<endl;   
back(a);   
}   
#include<iostream.h>   
void back(char \*p)   
{   
int i=0;   
while(\*p!='\0')   
{   
p++;//把指针定位到字符串末尾   
i++;//统计字符个数   
}   
cout<<"a=";   
for(;i>0;i--)//逆序输出   
{   
p--;   
cout<<\*p;   
}   
cout<<endl;   
}   
  
/\* 2、打印出杨辉三角形（要求打印出前15行）。(杨辉三角最本质的特征是，它的两条斜边都是由数字1组成的，而其余的数则是等于它肩上的两个数之和。) \*/   
#include<iostream.h>   
void tri(int a[][15]);   
void main()   
{   
int i,j,a[15][15];   
tri (a);   
cout<<"a= ";   
for(i=0;i<15;i++)//遍历整个数组   
{   
for(j=0;j<=i;j++)   
{   
cout<<a[i][j];   
if(a[i][j]>=1&&a[i][j]<=9)//当输出个位数之后输出4个空格保持整齐   
cout<<"    ";   
else if (a[i][j]>=10&&a[i][j]<=99)//当输出十位数之后输出3个空格保持整齐   
cout<<"   ";   
else if(a[i][j]>=100&&a[i][j]<=999)//当输出百位数之后输出2个空格保持整齐   
  cout<<"  ";   
  else cout<<" ";//当输出百位数之后输出1个空格保持整齐   
}   
cout<<endl<<"   ";//每行输出结束后换行   
}   
}   
void tri(int a[15][15])   
{   
int i,j;   
for(i=0;i<15;i++)   
for(j=0;j<=i;j++)   
{   
if(j==0||j==i)//三角形第一列和对角线被赋值为1   
a[i][j]=1;   
else a[i][j]=a[i-1][j-1]+a[i-1][j];//算出其余的数组元素   
}   
}   
  
/\* 3、编一程序，将两个字符串连接起来，不要用strcat函数。 \*/   
#include<iostream.h>   
#include<string.h>   
void scat(char \*,char \*);   
void main()   
{   
const int size=100;   
char a[size]="Hello";   
char b[size]="Bye";   
cout<<"a="<<a<<"   b="<<b<<endl;   
scat(a,b);   
cout<<"a="<<a<<" after link a and b"<<endl;   
}   
void scat(char \*p,char \*q)   
{   
while(\*p!='\0')//确定数组a的插入位置   
{   
p++;   
}   
while(\*q!='\0')   
{   
\*p=\*q;   
p++;   
q++;   
}   
}   
  
/\* 4、打印“魔方阵”。所谓魔方阵是指这样的方阵，它的每一行、每一列和对角线之和均相等。例如：三阶魔方阵：   
      8 1 6   
      3 5 7   
      4 9 2   
要求打印由1到n2的自然数构成的所有魔方阵。 \*/   
//方法一：输出N介魔方阵，但每介只输出一种。   
#include<iostream.h>   
void square(int a[][10],int k,int n);   
void main()   
{   
int n,i,j,k,a[10][10]={0};   
cout<<"input an odd number:"<<endl;   
cin>>n;   
k=n/2;//确定第一个数列数   
square(a,k,n);   
for(i=0;i<n;i++)   
{   
for(j=0;j<n;j++)   
cout<<"\t"<<a[i][j];   
cout<<endl;   
}   
}   
void square(int a[][10],int k,int n)   
{   
int i,j;   
for(i=1,j=0;i<=n\*n;i++,j--,k++)//n为阶数，从1开始给数组赋值   
{   
if(j<0&&k>=n)//当数组行列都越出范围时候，确定数组正确位置   
{   
j+=2;k-=1;   
}   
else if(j<0)//当数组行越出范围时候，确定数组正确位置   
j+=n;   
else if(k>=n)//当数组列越出范围时候，确定数组正确位置   
          k-=n;   
  else if(a[j][k]!=0)//当数组原位置有数时候，确定数组位置   
  {   
j+=2;k-=1;   
  }   
a[j][k]=i;   
}   
}   
//方法二：输出N介魔方阵所有魔方阵。   
#include <string>   
#include <iomanip>   
#include <iostream>   
  
using namespace std;   
  
void printA(int \*\*p,int n)//输出这个n阶魔方阵   
{   
cout<<endl<<"下面是一个"<<n<<"阶魔方阵:"<<endl;   
  
int i,j;   
  
for(i = 0;i < n;i++)   
{   
for(j = 0;j < n;j++)   
{   
cout<<setw(4)<<p[i][j];   
}   
  
cout<<endl;   
}   
  
cout<<endl<<endl;   
}   
bool Judge(int \*\*p,int n)//判断是否为n阶魔方阵   
{   
int i,j,sum = 0,NowSum = 0;   
bool YesOrNo = true;   
  
for(j = 0;j < n;j++)//第一行总和   
{   
sum += p[0][j];   
}   
  
for(i = 1;i < n;i++)//判断每行总和是否相等   
{   
NowSum = 0;   
for(j = 0;j < n;j++)   
{   
NowSum += p[i][j];   
}   
  
if(NowSum != sum)   
{   
YesOrNo = false;   
  
goto END;   
}   
}   
  
for(i = 0;i < n;i++)//每列是否相等   
{   
NowSum = 0;   
for(j = 0;j < n;j++)   
{   
NowSum += p[j][i];   
}   
  
if(NowSum != sum)   
{   
YesOrNo = false;   
  
goto END;   
}   
}   
  
NowSum = 0;   
for(i = 0,j = 0;i < n,j < n;i++,j++)//主对角线是否相等   
{   
NowSum += p[i][j];   
}   
if(NowSum != sum)   
{   
YesOrNo = false;   
  
goto END;   
}   
  
NowSum = 0;   
for(i = n-1,j = 0;i >= 0,j < n;i--,j++)//次对角线是否相等   
{   
NowSum += p[i][j];   
}   
if(NowSum != sum)   
{   
YesOrNo = false;   
  
goto END;   
}   
  
END:   
return YesOrNo;   
}   
void combination(int \*\*p,int n,int \*a)//求m = n\*n个数(1,2,3.....m)的全排列   
{   
int m = n\*n;   
static int Num = 0;   
int \*b\_val = new int[m];   
int c = 0,k,i,j;   
  
b\_val[c] = -1;//一维数组首地址的值赋-1 c[0]-1   
while(b\_val[0] < m) //-1   
{   
if(++b\_val[c] < m) //分别从0开始累加每个元素值，并限制不超出最大阶数   
                 //b[0]0 [1]0 [1]1 [2]0 [2]1 [2]2...[6]6...[7]0...[7]7 [8]0...[8]8 判断魔方 [8]9 012345678   
                      //                                               [7]8 [8]0...[8]7 判断魔方 [8]8 [8]9 012345687   
     //                                 [6]7...[7]0...[7]6        [8]8 012345768   
                                           //         012345786   
     //                                                                012345867 ...   
           //      876543210   
{     
for(k = 0;k < c;k++) //是否与前面数字重复，如有重复元素跳出，否则使K下标等于C下标                                    
if(b\_val[k] == b\_val[c])                                                       
break;   
      
if(k == c)//如果没有重复元素，就可以确定当前元素值，并继续排列下一个下标的数组元素   
{   
if(c+1 < m) //1 2 3...7 8   如果不满足条件，则生成了一组排列方式，否则继续排列下一个元素   
           //          8   
{      
++c; //1 2 3...7 8   
//          8   
b\_val[c] = -1;   
//   continue;   
}   
else  //生成了一组排列方式   
{   
k = -1;   
for(i = 0;i < n;i++)   
{   
for(j = 0;j < n;j++)   
{   
p[i][j] = a[b\_val[++k]];//a[0]-a[8]   
}   
}   
  
//判断是否为n阶魔方阵   
if(Judge(p,n))   
{   
printA(p,n);   
}   
}   
}   
}   
else   
{   
c--;   
}   
}   
  
delete []b\_val;   
}   
void evaluate(int \*\*p,int n)//给n阶方阵的元素赋值   
{   
int i;   
  
int \*AllNum = new int[3\*3];   
for(i = 1;i <= n\*n;i++)   
{   
AllNum[i - 1] = i;   
}   
  
combination(p,n,AllNum);   
  
delete []AllNum;   
}   
void main()   
{   
int i,n,\*\*a;   
string s;   
  
do   
{   
//输入n阶方阵的阶数n   
cout<<"请输入n阶方阵的阶数n(退出程序按e或E键)：";   
cin>>s;   
  
if(!strcmp(s.c\_str(),"e") || !strcmp(s.c\_str(),"E"))   
{   
break;   
}   
else if(s.find\_first\_not\_of("0123456789") != string::npos)   
{   
cout<<"请输入有效的数字，不能含有非数字的字符。"<<endl;   
  
continue;   
}   
else   
{   
n = atoi(s.c\_str());   
  
if(n < 1)   
{   
cout<<"请输入有效的数字，必须 >= 1。"<<endl;   
  
continue;   
}   
  
//分配内存   
a = new int\*[n];   
for(i = 0; i < n; i++)   
{   
a[i] = new int[n];   
}   
  
cout<<"正在运算，请等待。。。。。。"<<endl;   
  
//给n阶方阵的元素赋值   
evaluate(a,n);   
  
cout<<"运算结束！"<<endl;   
  
for(i = 0; i < n; i++)   
{   
delete []a[i];   
}   
delete []a;   
}   
}while(1);   
  
return;   
}   
  
/\* 5、求一个3x3矩阵对角线元素之和。\*/   
#include<iostream.h>   
int dia(int a[][3]);   
void main()   
{   
int i,j,sum,a[3][3]={2,3,5,6,2,3,1,9,0};   
cout<<"a= ";   
for(i=0;i<3;i++)   
{   
for(j=0;j<3;j++)   
cout<<a[i][j];   
cout<<endl<<"   ";   
}   
sum=dia(a);   
cout<<"\nsum="<<sum<<endl;   
}   
int dia(int a[3][3])   
{   
int i,j,sum=0;   
for(i=0;i<3;i++)//主对角线之和   
for(j=i;j<=i;j++)   
sum+=a[i][j];   
for(j=0;j<3;j++)//另一个对角线之和   
for(i=2-j;i<=2-j;i++)   
if(i!=j)//避免重复累加两个对焦向重合的元素   
sum+=a[i][j];   
return sum;//返回对角线员素之和   
}   
  
/\* 6、编写一个程序，将字符数组s2中的全部字符拷贝到字符数组s1中。不用strcpy函数。拷贝时，‘\0’也要拷贝过去。 ‘\0’后面的字符不拷贝。\*/   
#include<iostream.h>   
void scopy(char a[],char b[]);   
void main()   
{   
int i;   
char a[10];   
char b[10]="Hello";   
scopy(a,b);   
for(i=0;a[i]!='\0';i++)   
cout<<a[i];   
cout<<endl;   
}   
void scopy(char a[],char b[])   
{   
int i;   
for(i=0;b[i]!='\0';i++)   
{   
a[i]=b[i];   
}   
a[i]='\0';   
}   
  
/\* 7、用筛选法求100之内的素数。（所谓素数就是除了1和它本身以外，不能再被别的整数整除，这种数称作素数(也称质数)。）\*/   
#include<iostream.h>   
void main()   
{   
int i,j;   
for(i=1;i<=100;i++)   
{   
for(j=2;j<i;j++)//判断素数   
if(i%j!=0);   
else break;//不是素数   
if(i==j)//相等为素数   
cout<<" "<<i;   
}   
cout<<endl;   
}   
  
/\* 8、用选择法对10个整数排序。\*/   
#include<iostream.h>   
void csort(int a[10]);   
void main()   
{   
int i;   
int a[10]={6,4,2,7,9,0,1,6,3,0};   
for(i=0;i<10;i++)//输出原数组数据顺序   
cout<<a[i];   
cout<<endl;   
csort(a);   
for(i=0;i<10;i++)//输出排序后的顺序   
cout<<a[i];   
cout<<endl;   
}   
void csort(int a[10])//排序   
{   
int i,j,k,temp=0;   
for(i=1;i<10;i++)   
{   
k=i;   
for(j=k+1;j<10;j++)//找出最小数的数组下标   
if(a[k]>a[j])k=j;   
if(k!=i)   
{   
temp=a[i];//把数放到正确位置   
a[i]=a[k];   
a[k]=temp;   
}   
}   
}   
  
第6章   指针   
  
/\* 1、在主函数中输入10个字符串。用另一函数对它们排序。然后在主函数输出这10个已排好序的字符串。（用指针实现）\*/   
#include<iostream.h>   
void psort(int \*p);   
void main()   
{   
int i,a[10];   
cout<<"please input ten numbers:\n";   
for(i=0;i<10;i++)   
cin>>a[i];   
psort(a);   
cout<<"resort result=";   
for(i=0;i<10;i++)   
cout<<a[i]<<" ";   
}   
void psort(int \*p)   
{   
int i,j,temp;   
for(i=0;i<10;i++) //每循环一次确定数组中一个数的位置   
for(j=i+1;j<10;j++) //每次循环对比一个数的大小   
{   
if(p[i]>p[j])   
{   
temp=p[j];   
p[j]=p[i];   
p[i]=temp;   
}   
}   
  
}   
  
/\* 2、输入一个字符串，内有数字和非数字字符，如A123x456 1233?8997jhlkll   
将其中连续的数字作为一个整数，依次存放到一数组a中，统计共有多少个整数，并输出这些数。\*/   
#include<iostream.h>   
#include<string.h>   
#include <windows.h>   
int charge(int \*,char \*);   
void main()   
{   
int a[50],i,numb;   
char b[50];   
cout<<"please input a character string:"<<endl;   
cin.getline(b,50);   
system("cls");   
cout<<"your character string is ";   
cout.write(b,strlen(b))<<endl;   
numb=charge(a,b);   
for(i=0;i<numb;i++)   
cout<<" a["<<i<<"]="<<a[i];   
cout<<endl<<"total numbers="<<numb<<endl;   
}   
int charge(int \*q,char \*p)//\*q指向新数组，\*p指向初始数组   
{   
int numb=0;   
for(;\*p!='\0';p++)//判断每个字符   
{   
if(\*p>='0'&&\*p<='9')   
{   
\*q=(\*p)-'0';//将字符型整数转换成整型整数赋值给新数组   
p++;   
while(\*p>='0'&&\*p<='9')//判断是否有连续字符型整数   
{   
\*q=(\*q)\*10+((\*p)-'0');//将连续字符型整数转换成一个整型整数赋值给新数组   
p++;   
}   
q++;   
numb++;//统计整数的个数   
}   
}   
return numb;   
}   
  
/\* 3、用指向指针的指针的方法对5个字符串排序并输出。\*/   
#include <iostream.h>   
#include <string.h>   
void sort(char \*\*str);   
void main()   
{   
int i;   
char \*string[5];   
cout<<"输入5个字符串："<<endl;   
for(i=0;i<5;i++)   
{   
string[i] = new char[10];   
cin.getline(\*(string+i),50);   
}   
sort(string);   
for(i=0;i<5;i++)   
delete [] string[i];   
}   
void sort(char \*\*str)   
{   
int i=0,j;   
char \*p=0;   
for(i=0;i<4;i++)   
{   
for(j=i+1;j<5;j++)   
{   
if(strcmp(\*(str+i),\*(str+j))<0)   
{   
p=\*(str+i);   
\*(str+i)=\*(str+j);   
\*(str+j)=p;   
}   
}   
}   
cout<<"after sort the chars :"<<endl;   
for(i=0;i<5;i++)   
{   
cout<<\*(str+i)<<endl;   
}   
}   
  
/\* 4、统计一字符串在另一个字符串中出现的次数。\*/   
#include<iostream.h>   
#include<string.h>   
int change(char \*,char \*);   
void main()   
{   
int sum;   
char a[10]="dog";   
char b[20]="sdlkdogsddydodog";   
cout.write(a,10)<<endl;   
cout.write(b,20)<<endl;   
sum=change(a,b);   
cout<<"sum="<<sum<<endl;   
}   
int change(char \*p,char \*q)   
{   
    int sum=0,i=0;   
while(\*q!='\0')   
{   
while(\*p==\*q&&\*p!='\0')//对比是否含有相等字符串   
{   
\*p++;   
\*q++;   
i++;   
}   
if(\*p=='\0')   
{   
sum+=1;//含有字符串个数   
}   
p=p-i;//第一个字符串重新定位   
q=q-i;//第二个字符串重新定位   
i=0;//重新累加移动次数   
q++;   
}   
return sum;   
}   
  
/\* 5、有n个整数,使其前面各数顺序向后移m个位置,最后m个数变成最前面的m个数.n和m从键盘输入。\*/   
#include<iostream.h>   
#include<string.h>   
void charge(int a[],int,int);   
void main()   
{   
int i,n,m,a[50];   
cout<<"请输入n的值：";   
cin>>n;   
cout<<"请输入移动位数：";   
cin>>m;   
cout<<"请输入整数：";   
for(i=0;i<n;i++)   
{   
cin>>a[i];   
}   
cout<<"您输入的整数为：";   
for(i=0;i<n;i++)   
{   
cout<<a[i]<<" ";   
}   
cout<<endl;   
charge(a,n,m);   
cout<<"移动后的整数为：";   
for(i=0;i<n;i++)   
{   
cout<<a[i]<<" ";   
}   
cout<<endl;   
}   
void charge(int a[],int n, int m)//n为整数个数，向右移动m个位置   
{   
int i,j;   
for(j=0;j<m;j++)//移动m个位置   
{   
for(i=n-1;i>=0;i--)//移动一个位置就要移动每一个数组元素   
{   
a[i+1]=a[i];   
}   
a[0]=a[n];   
}   
}   
  
/\* 6、有n个人围成一圈，顺序排号。从第1个人开始报数（从1到3报数），凡报到3的人退出圈子，问最后留下的是原来第几号的那位。\*/   
#include<iostream.h>   
void change(int a[],int n);   
void main()   
{   
int i,a[50],n;   
cout<<"输入人数：";   
cin>>n;   
for(i=0;i<n;i++)   
a[i]=i+1;   
change(a,n);   
}   
void change(int a[],int n)   
{   
int qnumber=0,i=0,k=0;   
while(n-qnumber>1)//直到只剩下1人时   
{   
if(a[i]!=0)k++; //报数   
if(k==3)   
{   
a[i]=0; //退出圈子的人   
qnumber++; //退出的总人数   
k=0; //重新开始报数   
}   
i++;   
if(i==n)i=0; //当所有人都报过数之后重新每人再次报数   
}   
for(i=0;i<n;i++)   
if(a[i]!=0)cout<<a[i]<<" ";   
}   
  
/\* 7、写一函数，实现两个字符串的比较。即自己写一个strcmp函数。\*/   
#include<iostream.h>   
#include<string.h>   
int change(char \*,char \*);   
void main()   
{   
int result;   
char a[10]="dog";   
char b[20]="sdlkdogsddydodog";   
cout.write(a,10)<<endl;   
cout.write(b,20)<<endl;   
result=change(a,b);   
cout<<"result="<<result<<endl;   
}   
int change(char \*p,char \*q)   
{   
    int sum=0,i=0,flag=0;   
while(\*p!='\0'&&\*q!='\0')   
{   
if(\*p>\*q)   
{   
flag=1;//第一个字符串大于第二个字符串返回1   
break;   
}   
if(\*p<\*q)   
{   
flag=-1;//第二一个字符串大于第一个字符串返回-1   
break;   
}   
if((\*p==\*q)&&\*(p+1)=='\0'&&\*(q+1)!='\0')   
{   
flag=-1;//第二一个字符串大于第一个字符串返回-1   
break;   
}   
\*p++;   
\*q++;   
}   
return flag;   
}   
  
/\* 8、将一个5 x 5的矩阵中最大的元素放在中心，四个角分别放四个最小的元素（顺序为从左到右，从上到下顺序依次从小到大存放），写一函数实现之。用main函数调用。\*/   
#include<iostream.h>   
void psort(int a[]);   
void main()   
{   
int i;   
int a[25]={5,3,52,6,2,1,8,0,23,56,7,21,23,4,57,62,15,31,45,6,43,78,12,53,41};   
for(i=0;i<25;i++)   
{   
cout<<" "<<a[i];   
}   
cout<<endl;   
psort(a);   
}   
void psort(int a[])   
{   
int i,j,k=0,temp,b[5][5];   
for(i=0;i<25;i++) //每循环一次确定数组中一个数的位置   
for(j=i+1;j<25;j++) //每次循环对比一个数的大小   
{   
if(a[i]>a[j])   
{   
temp=a[j];   
a[j]=a[i];   
a[i]=temp;   
}   
}   
for(i=0;i<5;i++)   
for(j=0;j<5;j++)   
{   
b[i][j]=a[k++];   
}   
temp=b[4][4]; //确定5个位置的数值   
b[4][4]=b[2][2];   
b[2][2]=temp;   
temp=b[0][1];   
b[0][1]=b[0][4];   
b[0][4]=temp;   
temp=b[0][2];   
b[0][2]=b[4][0];   
b[4][0]=temp;   
temp=b[0][3];   
b[0][3]=b[4][4];   
b[4][4]=temp;   
for(i=0;i<5;i++)   
{   
for(j=0;j<5;j++)   
{   
cout<<"\t"<<b[i][j];   
}   
cout<<endl;   
}   
}   
  
第7章 结构体与共用体   
  
/\* 1、用指针和结构体实现一双向链表，并实现其相应的增、删、遍历功能，并在实例中应用它。\*/   
#include <iostream.h>   
  
typedef struct node   
{   
int number;   
  
struct node \*next;   
struct node \*parent;   
}Node,\*LinkNode;   
  
class LinkClass   
{   
public:   
LinkNode first,current;//头指针，当前指针   
void init();//初始化函数   
LinkNode Insert(int data,LinkNode cur);//插入函数   
void Remove(LinkNode p);//删除函数   
void Prior(LinkNode head);//遍历函数   
};   
  
void LinkClass::init()   
{   
struct node \*head=new struct node;   
struct node \*tail=new struct node;   
  
head->parent=NULL;   
head->next=tail;   
tail->parent=head;   
tail->next=NULL;   
first=current=head;   
}   
  
LinkNode LinkClass::Insert(int data,LinkNode cur)   
{   
struct node \*newNode=new struct node;   
newNode->number = data;   
newNode->next = cur->next;   
cur->next = newNode;   
newNode->parent = cur;   
newNode->next->parent = newNode;   
cur=newNode;   
return cur;   
}   
  
void LinkClass::Prior(LinkNode head)   
{   
LinkNode cur=head->next;   
while(cur->next!=NULL)   
{   
cout<<cur->number<<" ";   
cur=cur->next;   
}   
cout<<""<<endl;   
}   
  
void LinkClass::Remove(LinkNode cur)   
{   
LinkNode temp=cur;   
temp->parent->next=temp->next;   
temp->next->parent=temp->parent;   
delete(temp);   
}   
  
void main()   
{   
LinkClass lc;   
lc.init();   
LinkNode cur=lc.current;   
for(int i=0;i<=10;i++)//用循环来初始化结构体内的number   
{   
cur=lc.Insert(i,cur);//调用插入函数   
}   
LinkNode head=lc.first;   
cout<<"没调用删除函数的遍历:"<<endl;   
lc.Prior(head);//遍历函数   
for(int j=0;j<=3;j++)//删除元素6   
{   
cur=cur->parent;   
}   
lc.Remove(cur);//执行删除函数   
cout<<"调用删除函数后的遍历:"<<endl;   
lc.Prior(head);   
}   
  
/\* 2、用指针和结构体实现栈及其方法，并在实例中应用它。\*/   
#include <stdlib.h>   
#include <iostream.h>   
  
#define STACK\_INIT\_SIZE 100   
#define STACKIN 10   
  
struct stack   
{   
int \*top;   
int \*base;   
int stacksize;   
    int initstack(stack &s)   
{   
s.base=(int \*)malloc(STACK\_INIT\_SIZE \* sizeof(int));   
if(!s.base) return 0;   
s.top=s.base;   
s.stacksize =STACK\_INIT\_SIZE;   
return 1;   
  
}   
int gettop(stack s,int &e)   
{   
if(s.top ==s.base) return 0;   
e=\*(s.top-1);   
return 1;   
}   
int push(stack &s,int e)   
{   
if(s.top -s.base>=s.stacksize)   
{   
s.base=(int \*)realloc(s.base,(s.stacksize + STACKIN)\*sizeof(int));   
if(!s.base) return 0;   
s.top =s.base+s.stacksize;   
s.stacksize +=STACKIN;   
}   
\*s.top++=e;   
return 1;   
}   
int pop(stack &s,int &e)   
{   
if (s.top ==s.base)return 0;   
e=\*--s.top;   
return 1;   
}   
};   
  
void main()   
{   
stack s;   
int result,e=0;   
result=s.initstack(s);   
if(result==1)   
cout<<"建栈成功!"<<endl;   
else   
cout<<"建栈失败!"<<endl;   
for(int i=0;i<5;i++)   
result=s.push(s,i);   
if(result==1)   
cout<<"初始化栈成功!"<<endl;   
else   
cout<<"初始化栈失败!"<<endl;   
s.gettop(s,e);   
cout<<"栈顶元素为："<<e<<endl;   
result=s.pop(s,e);   
if(result==1)   
cout<<"删除栈顶元素成功!"<<endl;   
else   
cout<<"删除栈顶元素失败!"<<endl;   
s.pop(s,e);   
cout<<"删除的栈顶元素为："<<e<<endl;   
}   
  
/\* 3、编写一函数print，打印一个学生的成绩数组，该数组中有5个学生的数据记录，每个记录包括num、name、score[3]，用主函数输入这些记录，用print函数输出这些记录。   
   4、在上题的基础上，编写一个函数input，用来输入5个学生的数据记录。\*/   
#include <string.h>   
#include <iostream.h>   
  
struct student   
{   
public:   
int num;   
char name[20];   
int score[3];   
};   
  
class contral   
{   
public:   
void print(struct student \*p,int count);   
int input(struct student \*p,int count);   
};   
  
void contral::print(struct student \*p,int count)   
{   
for(int i=0;i<=count;i++)   
{   
cout<<"Num:"<<p->num<<endl;   
cout<<"Name:"<<p->name<<endl;   
cout<<"JAVA:"<<p->score[0]<<endl;   
cout<<"C++:"<<p->score[1]<<endl;   
cout<<"English:"<<p->score[2]<<endl;   
p++;   
}   
}   
  
int contral::input(struct student \*p,int count)   
{   
while(true)   
{   
cout<<"请输入学号："<<endl;   
cin>>p->num;   
cout<<"请输入姓名："<<endl;   
cin>>p->name;   
cout<<"请输入JAVA分数："<<endl;   
cin>>p->score[0];   
cout<<"请输入C++分数："<<endl;   
cin>>p->score[1];   
cout<<"请输入English分数："<<endl;   
cin>>p->score[2];   
p++;   
count++;   
cout<<"输入Q退出,输入其它继续"<<endl;   
char str[8];   
cin>>str;   
if(strcmp(str, "q")==0)   
{   
break;   
}   
}   
return count;   
}   
  
void main()   
{   
int count=0;   
contral con;   
student NewS[100];   
student \*p=NewS;   
count=con.input(p,count);   
cout<<count<<endl;   
con.print(p,count-1);   
}   
  
/\* 5、将一个链表按逆序排列，即将链头当链尾，链尾当链头。\*/   
#include<iostream.h>   
  
struct num   
{   
int data;   
struct num \*next;   
};   
  
void fan(struct num  node1[])   
{   
  
for(int i=9;i>=0;i--)   
{   
node1[i].next=&node1[i-1];   
}   
}   
  
void main()   
{   
     struct num  node[10];   
     for(int i=0;i<10;i++)   
{   
       node[i].data=i;   
}   
     for(int j=0;j<10;j++)   
cout<<node[j].data;   
     fan(node);   
cout<<endl;   
cout<<node[9].data;   
     for(int k=9;k>0;k--)   
cout<<(node[k].next)->data;   
}   
第8章   类和对象   
  
/\*1、自定义一个字符串处理类CString，且能实现取子串、删除子串的功能。\*/   
#include<iostream.h>   
  
#define MAX\_LENGTH 100   
  
class CString   
{   
public:   
void cpystring(char \*str);   
void getSubstring(int beginStr,int endStr);   
void delSubstring(int beginStr,int endStr);   
void print();   
private:   
char cstr[MAX\_LENGTH];   
int length;   
};   
  
#include"CString.h"   
#include"string.h"   
  
void CString::cpystring(char \*str)   
{   
int i=0;   
while(\*str!='\0')   
{   
cstr[i++]=\*str;   
str++;   
}   
cstr[i]='\0';   
}   
  
void CString::getSubstring(int beginStr,int endStr)   
{   
int i,j=0;   
char pstr[MAX\_LENGTH];   
if(beginStr<0||endStr>MAX\_LENGTH||beginStr>endStr)   
{   
cout<<"error!"<<endl;   
}   
for(i=beginStr;i<endStr;i++,j++)   
{   
pstr[j]=cstr[i];   
}   
pstr[j]='\0';   
cpystring(pstr);   
}   
  
void CString::delSubstring(int beginStr,int endStr)   
{   
int i,j=0;   
char pstr[MAX\_LENGTH];   
if(beginStr<0||endStr>MAX\_LENGTH||beginStr>endStr)   
{   
cout<<"error!"<<endl;   
}   
for(i=0;i<beginStr;i++,j++)   
{   
pstr[j]=cstr[i];   
}   
for(i=endStr+1;i<strlen(cstr);i++,j++)   
{   
pstr[j]=cstr[i];   
}   
pstr[j]='\0';   
cpystring(pstr);   
}   
  
void CString::print()   
{   
cout<<cstr<<endl;   
}   
  
#include<iostream.h>   
#include"CString.h"   
  
main ()   
{   
CString str1,str2,str3;   
str1.cpystring("Just like before, it's yesterday once more!");//初始化str1   
str2=str1;   
str3=str1;   
str2.getSubstring(5,9);//截取字符串   
str3.delSubstring(10,16);//删除字符串   
str1.print();   
str2.print();   
str3.print();   
}   
  
/\*2、定义一个循环队列类，且实现其相关的成员操作函数，并实例化调用之。\*/   
CircularQueue.h文件：   
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
#define MAX\_SIZE 101   
  
class CircularQueue   
{   
private:   
int queue[MAX\_SIZE];   
int front;   
int rear;   
  
public:   
CircularQueue();   
virtual ~CircularQueue();   
bool isEmpty();   
bool isFull();   
bool push(int);   
int pop();   
};   
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
  
CircularQueue.cpp文件：   
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
#include "CircularQueue.h"   
  
CircularQueue::CircularQueue()   
{   
front = 0;   
rear = 0;   
}   
  
CircularQueue::~CircularQueue()   
{   
  
}   
  
bool CircularQueue::isFull()   
{   
if ((rear+1)%MAX\_SIZE == front)   
{   
return true;   
}   
else   
{   
return false;   
}   
}   
  
bool CircularQueue::isEmpty()   
{   
if (rear == front)   
{   
return true;   
}   
else   
{   
return false;   
}   
}   
  
bool CircularQueue::push(int e)   
{   
if (isFull())   
{   
return false;   
}   
  
queue[rear] = e;   
rear = ++rear % MAX\_SIZE;   
return true;   
}   
  
CircularQueue::pop()   
{   
if (isEmpty())   
{   
return 0;   
}   
  
int e = queue[front];   
front = ++front % MAX\_SIZE;   
return e;   
}   
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
  
main.cpp文件   
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
#include "CircularQueue.h"   
#include <iostream>   
  
using namespace std;   
  
void main()   
{   
CircularQueue cQueue;   
for (int i=0 ; i<75; i++)   
{   
cQueue.push(i);   
}   
  
for (i=0; i<50; i++)   
{   
cout << cQueue.pop() << " ";   
}   
cout << endl;   
  
for (i=0; i<60; i++)   
{   
cQueue.push(i);   
}   
  
for (i=0; i<85; i++)   
{   
cout << cQueue.pop() << " ";   
}   
cout << endl;   
}   
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
第9章   运算符重载   
  
/\*1、定义一个二维向量类Vector，并在该类中用成员函数方式重载一元运算符++（前、后缀）和二元运算符+。\*/   
class Vector   
{   
public:   
Vector();   
Vector(int x,int y);   
operator ++();   
Vector operator ++(int);   
Vector operator +(const Vector &a);   
void display();   
private:   
int x,y;   
};   
  
#include<iostream.h>   
#include"Vector.h"   
  
Vector::Vector(){}   
Vector::Vector(int x,int y)   
{   
this->x=x;   
this->y=y;   
}   
Vector::operator ++() {++x;++y;}   
Vector Vector::operator ++(int)   
{   
Vector s;   
s.x=x++;   
s.y=y++;   
return s;   
}   
Vector Vector::operator +(const Vector &v)   
{   
Vector sum;   
sum.x=x+v.x;   
sum.y=y+v.y;   
return sum;   
}   
void Vector::display()   
{   
cout<<"("<<x<<","<<y<<")"<<endl;   
}   
  
#include<iostream.h>   
#include"Vector.h"   
  
void main()   
{   
Vector v1(3,4),v2(1,2),v3;   
cout<<"v1=";v1.display();   
cout<<"v2=";v2.display();   
++v1;   
cout<<"++v1=";v1.display();   
cout<<"v2++=";(v2++).display();   
cout<<"v2=";v2.display();   
v3=v1+v2;   
cout<<"v1+v2=";v3.display();   
}   
  
/\*2、将第一题中的相关重载项改为用友元函数的方式重载。\*/   
class Vector   
{   
public:   
Vector();   
Vector(int x,int y);   
friend void operator ++(Vector &v);   
friend void operator ++(Vector &v,int);   
friend Vector operator +(const Vector &v1,const Vector &v2);   
void display();   
private:   
int x,y;   
};   
  
#include<iostream.h>   
#include"Vector.h"   
  
Vector::Vector(){}   
Vector::Vector(int x,int y)   
{   
this->x=x;   
this->y=y;   
}   
void operator ++(Vector &v) {++v.x;++v.y;}   
Vector operator ++(Vector &v,int)   
{   
Vector before(v.x,v.y);   
v.x++;   
v.y++;   
return before;   
}   
Vector operator +(const Vector &v1,const Vector &v2)   
{   
Vector sum;   
sum.x=v1.x+v2.x;   
sum.y=v1.y+v2.y;   
return sum;   
}   
void Vector::display()   
{   
cout<<"("<<x<<","<<y<<")"<<endl;   
}   
  
#include<iostream.h>   
#include"Vector.h"   
  
void main()   
{   
Vector v1(3,4),v2(1,2),v3;   
cout<<"v1=";v1.display();   
cout<<"v2=";v2.display();   
++v1;   
cout<<"++v1=";v1.display();   
cout<<"v2++=";(v2++).display();   
cout<<"v2=";v2.display();   
v3=v1+v2;   
cout<<"v1+v2=";v3.display();   
}   
  
/\*3、重载字符串处理类CString的“=”号和“+”号运算符。\*/   
class Cstring   
{   
public:   
Cstring(char \*pn);   
~Cstring();   
Cstring& operator=(Cstring &c);   
Cstring& operator+(Cstring &c);   
void display();   
private:   
char \*p;   
};   
  
#include<iostream.h>   
#include<string.h>   
#include"Cstring.h"   
  
Cstring::Cstring(char \*pn)   
{   
p=new char[strlen(pn)+1];   
strcpy(p,pn);   
}   
  
Cstring::~Cstring()   
{   
delete []p;   
}   
  
Cstring& Cstring::operator=(Cstring &s)   
{   
delete []p;   
p=new char[strlen(s.p)+1];   
strcpy(p,s.p);   
return \*this;   
}   
  
Cstring& Cstring::operator+(Cstring &s)   
{   
char \*pp = new char[strlen(p)+strlen(s.p)+1];   
strcpy(pp,p);   
strcat(pp,s.p);   
delete []p;   
p = pp;   
return \*this;   
}   
  
void Cstring::display()   
{   
cout<<p<<endl;   
}   
  
#include<iostream.h>   
#include <string.h>   
#include"Cstring.h"   
  
void main()   
{   
Cstring s1("first hello");   
Cstring s2("second hello");   
cout<<"赋值之前:"<<endl;   
cout<<"s1=";   
s1.display();   
cout<<"s2=";   
s2.display();   
cout<<"赋值之后:"<<endl;   
s1=s2;   
cout<<"s1=";   
s1.display();   
cout<<"s2=";   
s2.display();   
cout<<"相加之后:"<<endl;   
cout<<"s1+s2"<<endl;   
(s1+s2).display();   
}   
第11章 继承和派生类   
  
/\*1、利用虚基类，消除“两性人”中的冗余数据：姓名、年龄，并编程实现之。\*/   
class Person    
{   
public:   
Person(char \*name,int age);   
Person();   
virtual ~Person();   
  
protected:   
int age;   
char name[20];   
};   
  
#include "Person.h"   
#include <string.h>   
  
Person::Person(char \*name, int age)   
{   
strcpy(this->name,name);   
this->age = age;   
}   
  
class Man: virtual public Person   
{   
public:   
Man(char \*name,int age,char \*sex);   
Man();   
virtual ~Man();   
  
protected:   
char sex[8];   
};   
  
#include "Man.h"   
#include "Person.h"   
#include <string.h>   
  
Man::Man(char \*name,int age,char \*sex):Person(name,age)   
{   
strcpy(this->sex,sex);   
}   
  
#include"Person.h"   
  
class Woman: virtual public Person    
{   
public:   
Woman(char \*name,int age,char \*sex);   
Woman();   
virtual ~Woman();   
  
protected:   
char sex[8];   
};   
  
#include "Woman.h"   
#include "Person.h"   
#include <string.h>   
  
Woman::Woman(char \*name,int age,char \*sex):Person(name,age)   
{   
strcpy(this->sex,sex);   
}   
  
class Transexual:public Man,public Woman   
{   
public:   
print();   
Transexual(char \*name, int age, char \*sex1,char \*sex2);   
Transexual();   
virtual ~Transexual();   
  
};   
  
#include "Transexual.h"   
#include "Man.h"   
#include "Woman.h"   
#include "Person.h"   
#include <iostream.h>   
  
Transexual::Transexual(char \*name, int age, char \*sex1,char \*sex2):Person(name,age),Man(name,age,sex1),Woman(name,age,sex2)   
{   
  
}   
  
Transexual::print()   
{   
cout<<"姓名："<<name<<endl;   
cout<<"年龄："<<age<<endl;   
cout<<"性别："<<Man::sex<<" "<<Woman::sex<<endl;   
}   
  
/\*2、通过Point类派生出Circle和Rectangle类，再通过Circle和Rectangle派生出“足球场图形”类Football，并实例化调用Football这个类，且用cout打印跟踪所有类的构造函数和析构函数的调用过程。（注：下面是这几个类的图形描述，且要求每个类都重载构造函数，且都不用默认构造函数）\*/   
#include<iostream.h>   
  
class Point{   
public:   
Point(double x){   
this->x=x;   
cout<<"Point Constructor called"<<endl;   
}   
~Point(){   
cout<<"Point Destructor called"<<endl;   
}   
protected:   
double x;   
};   
  
#include<iostream.h>   
#define PI 3.14159265   
  
class Circle:virtual public Point{   
public:   
Circle(double x):Point(x){   
cout<<"Circle Constructor called"<<endl;   
}   
~Circle(){   
cout<<"Circle Destructor called"<<endl;   
}   
void setCarea(){   
carea=PI\*x/2\*x/2;   
}   
double getCarea(){   
return carea;   
}   
protected:   
double carea;   
  
};   
  
#include<iostream.h>   
  
class Rectangle:virtual public Point{   
public:   
Rectangle(double x,double y):Point(x){   
this->y=y;   
cout<<"Rectangle Constructor called"<<endl;   
}   
~Rectangle(){   
cout<<"Rectangle Destructor called"<<endl;   
}   
void setRarea(){   
rarea=x\*y;   
}   
double getRarea(){   
return rarea;   
}   
protected:   
double y,rarea;   
};   
  
#include<iostream.h>   
#define PI 3.14159265   
  
class Football:public Circle,public Rectangle{   
public:   
Football(double x,double y):Point(x),Circle(x),Rectangle(x,y){   
cout<<"Football Constructor called"<<endl;   
}   
~Football(){   
cout<<"Football Destructor called"<<endl;   
}   
void setFarea(){   
farea=x\*y+PI\*x/2\*x/2;   
}   
double getFarea(){   
return farea;   
}   
protected:   
double farea;   
};   
  
#include<iostream.h>   
#include"Point.h"   
#include"Circle.h"   
#include"Rectangle.h"   
#include"Football.h"   
  
void main(){   
Circle c(10);   
Rectangle r(10,20);   
Football f(10,20);   
c.setCarea();   
r.setRarea();   
f.setFarea();   
cout<<"Circle area:"<<c.getCarea()<<endl;   
cout<<"Rectangle area:"<<r.getRarea()<<endl;   
cout<<"Football area:"<<f.getFarea()<<endl;   
}   
第12章   模板   
  
/\*1、利用C++的模板机制定义单向队列类模板、链表类模板，并实例化应用之。\*/   
/\*单向队列类模板\*/   
#define MAX\_SIZE 50   
  
template<class T>   
class Temqueue   
{   
private:   
T queue[MAX\_SIZE];   
T front;   
T rear;   
  
public:   
Temqueue();   
virtual ~Temqueue();   
bool isEmpty();   
bool isFull();   
bool push(T);   
T pop();   
};   
  
template<class T>   
Temqueue<T>::Temqueue()   
{   
front = 0;   
rear = 0;   
}   
  
template<class T>   
Temqueue<T>::~Temqueue()   
{   
  
}   
  
template<class T>   
bool Temqueue<T>::isFull()   
{   
if ((rear-MAX\_SIZE) == front)   
{   
return true;   
}   
else   
{   
return false;   
}   
}   
  
template<class T>   
bool Temqueue<T>::isEmpty()   
{   
if (rear == front)   
{   
return true;   
}   
else   
{   
return false;   
}   
}   
  
template<class T>   
bool Temqueue<T>::push(T e)   
{   
if (isFull())   
{   
return true;   
}   
queue[rear] = e;   
rear = ++rear;   
return true;   
}   
  
template<class T>   
T Temqueue<T>::pop()   
{   
if (isEmpty())   
{   
return true;   
}   
T e = queue[front];   
front = ++front;   
return e;   
}   
  
#include "Temqueue.h"   
#include <iostream>   
  
using namespace std;   
  
void main()   
{   
int k;   
cout<<"请输入要给数列初始化的长度，队列长度为50。"<<endl;   
cin>>k;   
Temqueue<int> cQueue;   
for (int i=0 ; i<k; i++)   
{   
cQueue.push(i);   
}   
if(k>0)   
{   
for (i=0; i<k; i++)   
{   
if (i>=MAX\_SIZE)   
{   
cout<<"队列已满!"<<endl;   
break;   
}   
else   
cout<<cQueue.pop()<< " ";   
}   
cout<<endl;   
}   
else   
cout<<"队列为空!"<<endl;   
}   
  
/\*链表类模板\*/   
template<typename T> class List;     //对List类的申明   
template<typename T> class Node{     //定义一个描述节点的类   
public:   
Node();   
friend class List<T>;   
private:   
T data;   
Node<T> \* next;   
};   
  
template<typename T>Node<T>::Node(){   
data=0;   
next=NULL;   
}   
  
template<typename T>class List{                     //定义一个描述链表的类   
public:   
List();                                         //空链表的构造（链表中只含表头结点）   
~List();                                        //析构函数   
void MakeEmpty();                               //清空链表（删除链表中除表头结点以外的所有结点）   
Node<T> \* Find(T d);                            //查找数据域与d相同的结点（返回所找到结点的指针）   
void PrintList();                               //输出链表中各结点的数据域   
void CreateList(); //初始化链表   
private:   
Node<T> \* head,\* rear;   
};   
  
template<typename T>List<T>::List(){   
head=rear=new Node<T>;   
}   
  
template<typename T>void List<T>::MakeEmpty(){   
Node<T> \*temp;   
while(head->next!=NULL){   
temp = head->next ;   
    head->next = temp->next ;   
   delete temp ;   
}   
rear = head;   
}   
  
template<typename T>List<T>::~List(){   
MakeEmpty();   
delete head;   
}   
  
template<typename T>void List<T>::PrintList(){   
rear = head->next;   
while(rear!=NULL){   
cout<<rear->data<<endl ;   
rear = rear->next ;   
}   
}   
  
template<typename T>void List<T>::CreateList(){   
T d;   
cout<<"现在开始创建链表，请依次输入数据（以Ctrl+Z结束）："<<endl;   
while(cout<<"请输入：" && cin>>d){   
rear->next=new Node<T>;   
   rear->next->data=d;   
   rear=rear->next;   
   rear->next=NULL;   
}   
}   
  
#include <iostream>   
#include "Node.h"   
#include "List.h"   
using namespace std;   
  
void main(){   
List <int>l;   
    l.CreateList();   
l.PrintList();   
cout<<"over!"<<endl;   
}   
  
/\*2、定义一个类模板，然后生成其模板类并定义该类模板的派生类模板和派生类。\*/   
using namespace std;   
  
template <class T>   
class Te{   
public:   
T te;   
};   
  
template <class T>   
class Tte:public Te<T>{   
public:   
T tte;   
};   
  
class Fte:public Te<int>{   
public:   
int fte;   
};   
  
#include<iostream>   
#include"Te.h"   
#include"Tte.h"   
#include"Fte.h"   
  
void main(){   
Te<int> te;   
te.te=10;   
cout<<"te="<<te.te<<endl;   
Tte<int> itte;   
itte.tte=20;   
cout<<"itte="<<itte.tte<<endl;   
Tte<double> dtte;   
dtte.tte=3.14;   
cout<<"dtte="<<dtte.tte<<endl;   
Fte fte;   
fte.fte=10;   
cout<<"fte="<<fte.fte<<endl;   
}   
第13章 多态性与虚函数   
  
/\*1、利用虚函数，实现triangle（三角形），square（矩形），circle（圆）的面积计算函数show\_area。\*/   
using namespace std;   
  
class Shape{   
public:   
virtual double show\_area()=0;   
virtual char\* shapeName()=0;   
};   
  
class Circle:public Shape{   
public:   
Circle(double radius){   
this->radius=radius;   
}   
virtual double show\_area(){   
return 3.14\*radius\*radius;   
}   
virtual char\* shapeName(){   
return "Circle: ";   
}   
protected:   
double radius;   
};   
  
class Square:public Shape{   
public:   
Square(double length,double width){   
this->length=length;   
this->width=width;   
}   
virtual double show\_area(){   
return length\*width;   
}   
virtual char\* shapeName(){   
return "Square: ";   
}   
protected:   
double length,width;   
};   
  
class Triangle:public Shape{   
public:   
Triangle(double length,double height){   
this->length=length;   
this->height=height;   
}   
virtual double show\_area(){   
return length\*height/2;   
}   
virtual char\* shapeName(){   
return "Triangle: ";   
}   
protected:   
double length,height;   
};   
  
#include<iostream>   
#include"Shape.h"   
#include"Circle.h"   
#include"Square.h"   
#include"Triangle.h"   
  
void main(){   
Shape \*ptr;   
Circle circle(10);   
Square square(10,15);   
Triangle triangle(10,15);   
ptr=&circle;   
cout<<ptr->shapeName()<<ptr->show\_area()<<endl;   
ptr=&square;   
cout<<ptr->shapeName()<<ptr->show\_area()<<endl;   
ptr=&triangle;   
cout<<ptr->shapeName()<<ptr->show\_area()<<endl;   
}   
  
/\*2、将圆类circle的计算面积和计算周长的功能分别抽象成class area和class perimeter，并在这两个新类中利用纯虚函数功能实现面积和周长的统一调用接口显示函数Show。\*/   
using namespace std;   
  
class Circle{   
public:   
Circle(double radius){   
this->radius=radius;   
}   
virtual double show()=0;   
virtual char\* shapeName()=0;   
protected:   
double radius;   
};   
  
class Area:public Circle{   
public:   
Area(double radius):Circle(radius){   
}   
virtual double show(){   
return 3.14\*radius\*radius;   
}   
virtual char\* shapeName(){   
return "Circle Area: ";   
}   
};   
  
class Perimeter:public Circle{   
public:   
Perimeter(double radius):Circle(radius){   
}   
virtual double show(){   
return 2\*3.14\*radius;   
}   
virtual char\* shapeName(){   
return "Circle Perimeter: ";   
}   
};   
  
#include<iostream>   
#include"Circle.h"   
#include"Area.h"   
#include"Perimeter.h"   
  
void main(){   
Circle \*ptr;   
Area area(10);   
Perimeter perimeter(10);   
ptr=&area;   
cout<<ptr->shapeName()<<ptr->show()<<endl;   
ptr=&perimeter;   
cout<<ptr->shapeName()<<ptr->show()<<endl;   
}   
第14章  I/O流   
  
1、建立一个二进制文件，并对其进行各种操作。   
struct i\_Data //每个学生记录含准考证号、姓名、英语成绩共3个字段   
{   
char ID[30];   
char name[30];   
int score;   
};   
  
#include "i\_Data.h"   
#define NUM 2   
  
class Io{   
public:   
void input(){   
fstream infile;   
int i;   
infile.open("score.dat",ios::out|ios::binary);   
if(!infile){   
cout<<"score.dat can't writed.\n";   
abort();   
}   
for(i=0;i<NUM;i++){   
cout<<"请输入第"<<i+1<<"位考生信息：\n";   
cout<<"input 准考证号:";   
cin>>data[i].ID;   
cout<<"input 姓名:";   
cin>>data[i].name;   
cout<<"input 英语成绩:";   
cin>>data[i].score;   
cout<<endl;   
infile.write((char \*) &data[i], sizeof(data[i]));   
}   
infile.close();   
system("cls");   
}   
  
void output(){   
fstream outfile;   
int i;   
outfile.open("score.dat",ios::in|ios::binary);   
if(!outfile){   
cout<<"score.dat can't opened.\n";   
abort();   
}   
for(i=0;i<NUM;i++){   
outfile.read((char \*) &data[i], sizeof(data[i]));   
cout<<"第"<<i+1<<"位考生信息为：\n";   
cout<<"准考证号："<<data[i].ID<<endl;   
cout<<"姓名："<<data[i].name<<endl;   
cout<<"英语成绩："<<data[i].score<<endl;   
cout<<endl;   
}   
outfile.close();   
}   
private:   
struct i\_Data data[NUM];   
};   
  
#include <iostream>   
#include <fstream.h>   
#include <stdlib.h>   
#include "Io.h"   
#define NUM 2   
  
void main() {   
Io io;   
io.input(); //写入文件   
io.output(); //读取文件   
}   
第15章   异常处理   
  
1、自己写一个程序，在某种条件下抛出各种类型异常（如：整数、字符串、类对象、引用等），再捕捉这些异常，并进行相关处理，保证自己的程序不被中断，让它继续执行。   
#include<iostream>   
#include<string>   
using namespace std;   
  
class String{   
public:   
String(char\*, int);   
class Range{ //异常类1   
public:Range(int j):index(j){}   
int index;   
};   
class Size{}; //异常类2   
char& operator[](int k){   
if(0<=k && k<len)   
return p[k];   
        throw Range(k);   
}   
private:char\* p;   
int len;   
static int max;   
};   
  
int String::max = 20;   
String::String(char\* str, int si){   
if(si<0 || max<si)   
   throw Size();   
p=new char[si];   
strncpy(p, str, si);   
len=si;   
}   
  
void g(String& str){   
int num=10;   
for(int n=0; n<num; n++)   
   cout <<str[n];   
cout <<endl;   
}   
  
void f()   
{   
//代码区1   
try{   
//代码区2   
String s("abcdefghijklmnop", 10);   
g(s);   
}   
catch(String::Range r)   
{   
cerr <<"->out of range: " <<r.index <<endl;   
//代码区3   
}   
catch(String::Size)   
{   
cerr <<"size illegal!\n";   
}   
cout <<"The program will be continued here.\n\n";   
//代码区4   
}   
  
void main()   
{   
//代码区5   
f();   
cout <<"These code is not effected by probably exception in f().\n";   
}   
/\*小张同学看到你的一本Java书上写着“Object无处不在”，真的是这样吗？他一眼扫到写字台上一盏可调亮度的台灯。可以用Java表现台灯吗？他请你试试看。基本要求：   
1、编写台灯类，台灯具有开关状态和“标准”、“增强”两档可调亮度；台灯具有打开、关闭功能，并可以设置亮度档位；   
2、编写主类，显示台灯控制菜单，用户可以根据菜单选项控制不断调整一盏台灯的状态；   
3、选择控制菜单的“退出”选项可以结束程序运行。\*/   
using namespace std;   
  
class Light{   
public:   
Light();   
void setStatus(bool status);   
bool getStatus();   
void setBright(int bright);   
int getBright();   
void setStyle(int style);   
int getStyle();   
int input();   
void output();   
private:   
bool status;//表状态:开启或者关闭   
int bright;//表亮度   
int style;//表亮度类型:普通或者增强   
};   
  
#include<iostream>   
#include"Light.h"   
  
void Light::setStatus(bool status){   
this->status=status;   
}   
  
Light::Light(){   
status=false;   
bright=3;   
style=0;   
}   
  
bool Light::getStatus(){   
return status;   
}   
  
void Light::setBright(int bright){   
this->bright=bright;   
}   
  
int Light::getBright(){   
return bright;   
}   
  
void Light::setStyle(int style){   
this->style=style;   
}   
  
int Light::getStyle(){   
return style;   
}   
  
int Light::input(){   
cout<<"请输入你要输入的值:";   
int res=-1;//返回输入的值,   
cin>>res;   
if(cin.fail()){   
cout<<"输入的数据超出范围!";   
exit(0);//程序退出   
}   
return res;   
}   
  
void Light::output(){   
cout<<"当前电灯系统状态为--"<<(this->getStatus()?"开启":"关闭")<<",亮度类型为--"   
<<(this->getStyle()==1?"增强":"标准")<<",亮度档位为--"<<this->bright<<endl;   
}   
  
#include<iostream>   
#include"Light.h"   
  
void main(){   
Light light;   
do{   
cout<<"---------------<Control Menu>-----------------"<<endl;   
cout<<"1-开灯,2-调节状态,3-调节亮度,4-关灯"<<endl;   
cout<<"----------------------------------------------"<<endl;   
int answer=light.input();   
if(answer==1){   
light.setStatus(true);   
cout<<"系统启用中。。。"<<endl;   
light.output();   
}   
else if(answer==2){   
if(!light.getStatus()){   
cout<<"警告!系统尚未启动!操作失败!"<<endl;   
continue;   
}   
do{   
cout<<"请输入你要调度的状态(0-标准,1-增强):"<<endl;   
int style=light.input();   
if(style==0 || style==1){   
cout<<"操作成功!"<<endl;   
light.setStyle(style);   
light.output();   
break;   
}   
else{   
cout<<"您输入的值超出了选择范围!请重新输入!"<<endl;   
continue;   
}   
}while(true);   
}   
else if(answer==3){   
if(!light.getStatus()){   
cout<<"警告!系统尚未启动!操作失败!"<<endl;   
continue;   
}   
do{   
cout<<"请输入你要调度的亮度(1--5):"<<endl;   
int bright=light.input();;   
if( bright>=0 &&  bright<=5){   
cout<<"操作成功!"<<endl;   
light.setBright(bright);   
light.output();   
break;   
}   
else{   
cout<<"您输入的值超出了选择范围!请重新输入!"<<endl;   
continue;   
}   
}while(true);   
}   
else if(answer==4){   
if(!light.getStatus()){   
cout<<"警告!系统尚未启动!操作失败!"<<endl;   
continue;   
}   
cout<<"系统即将关闭!"<<endl;   
light.setStatus(false);   
break;   
}   
else {   
cout<<"您输入的值超出了选择范围!请重新输入!"<<endl;   
}   
}while(true);   
}   
  
/\*万年历\*/   
#include<iostream.h>   
#include<stdlib.h>   
#include<iomanip.h>   
void main()   
{   
int year,month,startyear,today,day0,day1,k,j;   
char a,b,c;   
int M[12]={31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};   
int M0[12]={ 31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };   
  
    do{   
do{   
cout<<"请输入年份(1990-2020):";   
cin>>year;   
if(year<1990||year>2020){   
cout<<"您输入的年份越界，是否重新输入(Y/N||y/n)?";   
cin>>a;   
if(a=='Y'||a=='y'){   
continue;}   
else exit(0);   
}   
else{break;}   
}while(true);   
do{   
cout<<"请输入月份(1-12):";   
cin>>month;   
if(month<1||month>12){   
cout<<"您输入的月份越界，是否重新输入(Y/N||y/n)?";   
cin>>b;   
if(b=='Y'||b=='y'){   
continue;}   
else{exit(0);}   
}   
else{break;}   
}while(true);   
//计算有效天数   
day0=0;   
day1=0;   
j=0;   
startyear=1990;   
    while (startyear != year)   
{   
     if ((startyear%4==0&&startyear%100!=0)||(startyear%400==0))   
         day0 = day0 + 366;   
     else   
         day0 = day0 + 365;   
     startyear++;   
}   
if((year%4==0&&year%100!=0)||(year%400==0)){   
for (int i = 0; i < (month - 1); i++)   
            day1 = day1 + M0[i];   
            today = day0 + day1;   
}   
else{   
for (int i = 0; i < (month - 1); i++)   
            day1 = day1 + M[i];   
            today = day0 + day1;   
}   
//输出部分   
  
//闰年输出    
if((year%4==0&&year%100!=0)||(year%400==0)){   
cout<<year<<"年"<<month<<"月份的日历如下"<<endl;   
cout<<setw(30)<<endl;   
cout << setw(8) << "Sun";   
        cout << setw(8) << "Mon";   
        cout << setw(8) << "Tue";   
        cout << setw(8) << "Wed";   
        cout << setw(8) << "Thu";   
        cout << setw(8) << "Fri";   
        cout << setw(8) << "Sat";   
        cout << endl;   
k=(today%7+1)%7;   
for(int i=1; i<=k; i++)   
cout<<setw(8)<<"";   
for(int g=1; g<=(7-k); g++){   
cout<<setw(8)<<g;   
}   
cout<<endl;   
for(int h=8-k; h<=M0[month-1]; h++) {   
cout<<setw(8)<<h;   
j++;   
if(j==7){   
j=0;   
cout<<endl;   
}   
}   
}   
//平年输出   
else {   
cout<<year<<"年"<<month<<"月份的日历如下"<<endl;   
cout<<setw(30)<<""<<endl;   
cout << setw(8) << "Sun";   
        cout << setw(8) << "Mon";   
        cout << setw(8) << "Tue";   
        cout << setw(8) << "Wed";   
        cout << setw(8) << "Thu";   
        cout << setw(8) << "Fri";   
        cout << setw(8) << "Sat";   
        cout << endl;   
k=(today%7+1)%7;   
for(int i=1; i<=k; i++)   
cout<<setw(8)<<"";   
for(int g=1; g<=(7-k); g++){   
cout<<setw(8)<<g;   
}   
cout<<endl;   
for(int h=8-k; h<=M[month-1]; h++) {   
cout<<setw(8)<<h;   
j++;   
if(j==7){   
j=0;   
cout<<endl;   
}   
}   
}   
cout<<endl;   
cout<<"是否继续输入(Y/N||y/n)?";   
cin>>c;   
if(c=='Y'||c=='y') continue;   
else exit(0);   
}while(true);   
}