**八、程序设计**

1. 编一程序求出满足不等式1+1/2+1/3+...+1/n≥5的最小n值。

2. 计算1+3+32+...+310的值并输出，假定分别用i,p,s作为循环变量、累乘变量和累加变量的标识符。

3. 求满足不等式22+42+...+n2<1000的最大n值，假定分别用i和s作为取偶数值和累加值的变量，并限定使用do循环编程。

4. 已知，求出并显示当x分别取-3.8, 6.4, 2.3, -4.2,8.9, 3.5, -5.0, 4.5时所对应的y值，要求把a定义为常量，其值设定为10.2，x的每个值由键盘输入，并假定用-100作为终止标志，求平方根函数为sqrt(x)。

5. 求出从键盘上输入的10个整数中的最大值，要求输入变量用x表示，存储最大值的变量用max表示。

6. 已知6≤a≤30，15≤b≤36，求满足不定方程2a+5b=126的全部整数组解。如(13, 20)就是一个整数组解，并以所给的样式输出每个解。

7. 某班级学生进行百米跑测试，规定成绩在12秒以内（含12秒）为优秀，在12秒以上至15秒为达标，在15秒以上为不达标，编一程序，从键盘上输入每个人的成绩，以x作为输入变量，并以小于0的任何数作为终止标志，分别用变量c1,c2和c3统计出成绩为优秀、达标和不达标的人数。

8. 编写一个函数，分别求出由指针a所指向的字符串中包含的每种十进制数字出现的次数，把统计结果保存在数组b的相应元素中。

9. 按照下面函数原型语句编写一个函数，返回二维数组a[m][n]中所有元素的平均值，假定采用变量v存放平均值。

10. 按照下面函数原型语句编写一个递归函数计算出数组a中n个元素的平方和并返回。

int f(int a[],int n);

11. 按照函数原型语句“void p(int n);”编写一个递归函数显示出如下图形，此图形是n=5的情况。

55555

4444

333

22

1

12. 按照函数原型语句“void p(int n);”编写一个递归函数显示出如下图形，此图形是n=5的情况。

1

22

333

4444

55555

13. 根据下面类中Count函数成员的原型和注释写出它的类外定义。

class AA {

int\* a;

int n;

int MS;

public:

void InitAA(int aa[], int nn, int ms) {

if(nn>ms) {cout<<"Error!"<<endl; exit(1);}

MS=ms;

n=nn;

a=new int[MS];

for(int i=0; i<MS; i++) a[i]=aa[i];

}

int Count(int x); //从数组a的前n个元素中统计出其

//值等于x的个数并返回。

};

14. 根据下面类中Search函数成员的原型和注释写出它的类外定义。

class AA {

int\* a;

int n;

int MS;

public:

void InitAA(int aa[], int nn, int ms) {

if(nn>ms) {cout<<"Error!"<<endl; exit(1);}

MS=ms;

n=nn;

a=new int[MS];

for(int i=0; i<MS; i++) a[i]=aa[i];

}

int Search(int x); //从数组a的前n个元素中顺序查找值为x的元素，

//若查找成功则返回元素的下标，否则返回-1。

};

15. 根据下面类中MaxMin 函数成员的原型和注释写出它的类外定义。

class AA {

int\* a;

int n;

int MS;

public:

void InitAA(int aa[], int nn, int ms) {

if(nn>ms) {cout<<"Error!"<<endl; exit(1);}

MS=ms;

n=nn;

a=new int[MS];

for(int i=0; i<MS; i++) a[i]=aa[i];

}

int MaxMin(int& x, int& y); //从数组a的前n个元素中求出

//最大值和最小值，并分别由引用参数x和y带回，

//同时若n大于0则返回1，否则返回0。

};

16. 根据下面类中Compare 函数成员的原型和注释写出它的类外定义。

class AA {

int\* a;

int n;

int MS;

public:

void InitAA(int aa[], int nn, int ms) {

if(nn>ms) {cout<<"Error!"<<endl; exit(1);}

MS=ms;

n=nn;

a=new int[MS];

for(int i=0; i<MS; i++) a[i]=aa[i];

}

int Compare(AA b); //比较\*this与b的大小，若两者中

//的n值相同，并且数组中前n个元素值对应

//相同，则认为两者相等返回1，否则返回0。

};

17. 根据下面类中CompareBig 函数成员的原型和注释写出它的类外定义。

class AA {

int\* a;

int n;

int MS;

public:

void InitAA(int aa[], int nn, int ms) {

if(nn>ms) {cout<<"Error!"<<endl; exit(1);}

MS=ms;

n=nn;

a=new int[MS];

for(int i=0; i<MS; i++) a[i]=aa[i];

}

int CompareBig(AA b); //比较\*this与b的大小，从前向后按两数组

//中的对应元素比较，若\*this中元素值大则返回1，若b中

//元素值大则返回-1，若相等则继续比较下一个元素，直到

//一个数组中无元素比较，此时若两者的n值相同则返回0，

//否则若\*this中的n值大则返回1，若b中的n值大则返回-1。

};

18. 根据下面类中Reverse 函数成员的原型和注释写出它的类外定义。

class AA {

int\* a;

int n;

int MS;

public:

void InitAA(int aa[], int nn, int ms) {

if(nn>ms) {cout<<"Error!"<<endl; exit(1);}

MS=ms;

n=nn;

a=new int[MS];

for(int i=0; i<MS; i++) a[i]=aa[i];

}

AA\* Reverse(); //对于调用该函数的对象，将其a数组中前n个

//元素值按相反的次序排列，返回指向该对象的指针。

};

19. 根据下面类中Reverse1函数成员的原型和注释写出它的类外定义。

class AA {

int\* a;

int n;

int MS;

public:

void InitAA(int aa[], int nn, int ms) {

if(nn>ms) {cout<<"Error!"<<endl; exit(1);}

MS=ms;

n=nn;

a=new int[MS];

for(int i=0; i<MS; i++) a[i]=aa[i];

}

AA\* Reverse1(); //通过动态存储分配得到一个对象，并动态分配

//a[MS]数组空间,要求该对象中的n和MS的值与\*this中的

//对应成员的值相同，数组元素的值是按照\*this中数组元

//素的相反次序排列得到的，要求该函数返回动态对象的地址。

};

20. 根据下面类中构造函数的原型和注释写出它的类外定义。

class Array {

int \*a; //指向动态分配的整型数组空间

int n; //记录数组长度

public:

Array(int aa[], int nn); //构造函数，利用aa数组长度nn初始化n，

//利用aa数组初始化a所指向的数组空间

Array(Array& aa); //拷贝构造函数

Array& Give(Array& aa); //实现aa赋值给\*this的功能并返回\*this

Array Uion(Array& aa); //实现\*this和aa中的数组合并的

//功能，把合并结果存入临时对象并返回

int Lenth() {return n;} //返回数组长度

void Print() { //输出数组

for(int i=0; i<n; i++)

cout<<a[i]<<' ';

cout<<endl;

}

};

21. 根据下面类中拷贝构造函数的原型写出它的类外定义。

class Array {

int \*a; //指向动态分配的整型数组空间

int n; //记录数组长度

public:

Array(int aa[], int nn); //构造函数，利用aa数组长度nn初始化n，

//利用aa数组初始化a所指向的数组空间

Array(Array& aa); //拷贝构造函数

Array& Give(Array& aa); //实现aa赋值给\*this的功能并返回\*this

Array Uion(Array& aa); //实现\*this和aa中的数组合并的

//功能，把合并结果存入临时对象并返回

int Lenth() {return n;} //返回数组长度

void Print() { //输出数组

for(int i=0; i<n; i++)

cout<<a[i]<<' ';

cout<<endl;

}

};

22. 根据下面类中Give函数的原型和注释写出它的类外定义。

class Array {

int \*a; //指向动态分配的整型数组空间

int n; //记录数组长度

public:

Array(int aa[], int nn); //构造函数，利用aa数组长度nn初始化n，

//利用aa数组初始化a所指向的数组空间

Array(Array& aa); //拷贝构造函数

Array& Give(Array& aa); //实现aa赋值给\*this的功能并返回\*this

Array Uion(Array& aa); //实现\*this和aa中的数组合并的

//功能，把合并结果存入临时对象并返回

int Lenth() {return n;} //返回数组长度

void Print() { //输出数组

for(int i=0; i<n; i++)

cout<<a[i]<<' ';

cout<<endl;

}

};

23. 根据下面类中Uion 函数的原型和注释写出它的类外定义。

class Array {

int \*a; //指向动态分配的整型数组空间

int n; //记录数组长度

public:

Array(int aa[], int nn); //构造函数，利用aa数组长度nn初始化n，

//利用aa数组初始化a所指向的数组空间

Array(Array& aa); //拷贝构造函数

Array& Give(Array& aa); //实现aa赋值给\*this的功能并返回\*this

Array Uion(Array& aa); //实现\*this和aa中的数组合并的功能，把合并

//结果(其长度为两数组长度之和)存入临时对象并返回

int Lenth() {return n;} //返回数组长度

void Print() { //输出数组

for(int i=0; i<n; i++)

cout<<a[i]<<' ';

cout<<endl;

}

};

24. 根据下面类中构造函数的原型和注释写出它的类外定义。

class Strings {

char \*s; //指向动态分配的字符串数组空间

int n; //记录字符串长度

public:

Strings(char\*str); //构造函数，利用str字符串长度初始化n，

//利用str字符串初始化s所指的字符串空间

Strings(Strings& str); //拷贝构造函数

Strings& Give(Strings& str); //实现str赋值给\*this的功能

Strings Uion(Strings& str); //实现\*this和str中的字符串合并的

//功能，把合并结果存入临时对象并返回

int Lenth() {return n;} //返回字符串长度

void Print() {cout<<s<<endl;} //输出字符串

};

25. 根据下面类中拷贝构造函数的原型写出它的类外定义。

class Strings {

char \*s; //指向动态分配的字符串数组空间

int n; //记录字符串长度

public:

Strings(char\*str); //构造函数，利用str字符串长度初始化n，

//利用str字符串初始化s所指的字符串空间

Strings(Strings& str); //拷贝构造函数

Strings& Give(Strings& str); //实现str赋值给\*this的功能

Strings Uion(Strings& str); //实现\*this和str中的字符串合并的

//功能，把合并结果存入临时对象并返回

int Lenth() {return n;} //返回字符串长度

void Print() {cout<<s<<endl;} //输出字符串

};

26. 根据下面类中Give函数的原型和注释写出它的类外定义。

class Strings {

char \*s; //指向动态分配的字符串数组空间

int n; //记录字符串长度

public:

Strings(char\*str); //构造函数，利用str字符串长度初始化n，

//利用str字符串初始化s所指的字符串空间

Strings(Strings& str); //拷贝构造函数

Strings& Give(Strings& str); //实现str赋值给\*this的功能并返回\*this

Strings Uion(Strings& str); //实现\*this和str中的字符串合并的

//功能，把合并结果存入临时对象并返回

int Lenth() {return n;} //返回字符串长度

void Print() {cout<<s<<endl;} //输出字符串

};

27. 根据下面类中Uion函数的原型和注释写出它的类外定义。

class Strings {

char \*s; //指向动态分配的字符串数组空间

int n; //记录字符串长度

public:

Strings(char\*str); //构造函数，利用str字符串长度初始化n，

//利用str字符串初始化s所指的字符串空间

Strings(Strings& str); //拷贝构造函数

Strings& Give(Strings& str); //实现str赋值给\*this的功能并返回\*this

Strings Uion(Strings& str); //实现\*this和str中的字符串连接的功能，

//把连接结果存入临时对象并返回

int Lenth() {return n;} //返回字符串长度

void Print() {cout<<s<<endl;} //输出字符串

};

28. 下列程序段中，A\_class的成员函数Variance()可求出两数的平方差，请改写该程序段，把Variance()函数从A\_class类中分离出来，用友元函数来实现该函数的功能。

class A\_class {

private:

int x,y,t;

public:

A\_class(int i,int j):x(i),y(j) {

if(y>x){t=x;x=y;y=t;}

}

int Variance(){return x\*x-y\*y;}

//其它函数从略

};

void main() {

A\_class A\_obj(3,5);

cout<<"Result:"<<A\_obj.Variance()<<endl;

}

29. 下面给出了矩阵类Matrix定义。为了求两个矩阵对象的乘积，需要定义一个Matrix的友元函数Multiply()。请按照友元函数Multiply（）的声明编写出该函数的定义。

class Matrix {

public:

Matrix(int row,int col); //构造一个具有row行col列的矩阵

~Matrix() {delete []mem;} //析构函数

friend bool Multiply(Matrix &m1, Matrix &m2, Matrix &m3);

//定义Multiply()为友元函数，该函数把m1×m2的值赋给m3

//其他成员函数从略

private:

int \*mem; //动态申请矩阵空间

const int rows,cols; //矩阵的行数和列数

};

Matrix::Matrix(int row,int col):rows(row),cols(col)

{

mem = new int[row\*col];

}

bool Multiply(Matrix &m1, Matrix &m2, Matrix &m3)

{

//确定矩阵是否能够进行相乘

if(m1.rows != m3.rows ||m2.cols != m3.cols || m1.cols != m2.rows) return false;

//定义sum变量，用于计算乘积矩阵m3中每个元素的值

int sum;

//请在下面编写剩余部分

}

30. 已知类定义如下，其中Shape为基类，Circle和Rectangle分别Shape的直接派生类，Square为Rectangle的直接派生类和Shape的间接派生类。请模仿Circle类，写出Rectangle类的所有成员函数。

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*文件shape.h\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

const float PI=3.14159f; //定义圆周率常量

class Shape //几何图形抽象类

{

public:

virtual float GetPerimeter()=0; //纯虚函数，计算周长

virtual float GetAre()=0; //纯虚函数，计算面积

};

class Circle: public Shape //圆类

{

public:

Circle(float rad):rad(rad){}

~Circle(){}

float GetPerimeter() {return 2\*PI\*rad;} //计算圆形周长

float GetAre() {return PI\*rad \*rad;} //计算圆形面积

private:

float rad; //圆的半径

};

class Rectangle: public Shape //矩形类

{

public: //在下面编写每个成员函数

private:

float length, width; //矩形的长和宽

};

class Square: public Rectangle //正方形类

{

public:

Square(float len): Rectangle(len,len){}

~Square(){}

};

31. 已知类定义如下，其中Shape为基类，Circle和Rectangle分别Shape的直接派生类，Square为Rectangle的直接派生类和Shape的间接派生类。请模仿Rectangle类，写出Circle类的所有成员函数。

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*文件shape.h\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

const float PI=3.14159f; //定义圆周率常量

class Shape //几何图形抽象类

{

public:

virtual float GetPerimeter()=0; //纯虚函数，计算周长

virtual float GetAre()=0; //纯虚函数，计算面积

};

class Rectangle: public Shape //矩形类

{

public:

Rectangle (float len,float wid):length(len),width(wid){}

~Rectangle (){}

float GetPerimeter() {return 2\*(length+width);} //计算矩形周长

float GetAre() {return length\*width;} //计算矩形面积

private:

float length, width; //矩形的长和宽

};

class Circle: public Shape //圆类

{

public: //在下面编写每个成员函数

private:

float rad; //圆的半径

};

class Square: public Rectangle //正方形类

{

public:

Square(float len): Rectangle(len,len){}

~Square(){}

};