//根据半径计算圆的周长和面积

#include <iostream.h>

const float PI=3.1416; //声明常量(只读变量)PI为3.1416

float fCir\_L(float); //声明自定义函数fCir\_L()的原型

float fCir\_S(float); //声明自定义函数fCir\_S()的原型

//以下是main()函数

main()

{

float r,l,s; //声明3个变量

cout<<"R="; //显示字符串

cin>>r; //键盘输入

l=fCir\_L(r); //计算圆的周长，赋值给变量l

s=fCir\_S(r); //计算圆的面积，赋值给变量s

cout<<"l="<<l; //显示计算结果

cout<<"\ns="<<s;

}

//定义计算圆的周长的函数fCir\_L()

float fCir\_L(float x)

{

float z=-1.0; //声明局部变量

if (x>=0.0) //如果参数大于0，则计算圆的周长

z=2\*PI\*x;

return(z); //返回函数值

}

//定义计算圆的面积的函数fCir\_S()

float fCir\_S(float x)

{

float z=-1.0; //声明局部变量

if (x>=0.0) //如果参数大于0，则计算圆的面积

z=PI\*x\*x;

return(z); //返回函数值

}

/\* Program: P1-2.CPP

Written by: Hap

Date written: 02:11:10

\*/

#include <iostream.h>

void main(void)

{

double s1,s2,s3;

s1=1.5; /\* 对变量s1赋值\*/

cout<<"s1="<<s1<<endl;

/\* 对变量s2赋值\*/ s2=2.5;

cout<<"s2="<<s2<<endl;

s3= /\* 对变量s3赋值\*/ 3.5;

cout<<"s3="<<s3<<endl;

cout<<"s1+s2+s3="<<s1+s2+s3<<endl; //计算并显示

//计算并显示 cout<<"s1+s2+s3="<<s1+s2+s3<<endl;

}

#include <iostream.h>

main()

{

double r=1.0;

cout<<"r="<<r<<endl;

double l;

l=2\*3.1416\*r; //计算圆的周长，赋值给变量l

cout<<"l="<<l<<endl; //显示圆的周长

double s=3.1416\*r\*r; //计算圆的面积，赋值给变量s

cout<<"s="<<s<<endl; //显示圆的面积

cout<<"R="; //显示提示输入的信息

cin>>r; //键盘输入

l=2\*3.1416\*r; //计算圆的周长，赋值给变量l

cout<<"l="<<l<<endl; //显示圆的周长

s=3.1416\*r\*r;

cout<<"s="<<s<<endl; //显示圆的面积

}

#include <iostream.h> //包含iostream.h头文件

void main()

{

//输出字符常量、变量和字符串

char c1='A';

cout<<'W';

cout<<c1<<endl;

cout<<"This is a test."<<endl;

cout<<"------------------"<<endl;

//输出整型常量、变量和表达式

int n=100;

cout<<10;

cout<<n;

cout<<2\*n<<endl; //输出整型表达式

cout<<"------------------"<<endl;

//输出浮点型常量、变量和表达式

double pi=3.1415926,r=10.0,s=pi\*r\*r;

cout<<pi<<endl;

cout<<r;

cout<<s;

cout<<2\*r\*pi<<endl; //输出浮点型表达式

cout<<"------------------"<<endl;

//一个cout可以输出多项数据

cout<<'W'<<" "<<c1<<endl;

cout<<"This is a test."<<endl;

cout<<"pi="<<pi<<" r="<<r<<" s="<<s<<endl;

}

#include <iostream.h> //包含iostream.h头文件

main()

{

//输入输出字符

char c;

cin>>c;

cout<<"c="<<c<<endl;

//输入输出整型数据

int n;

cin>>n;

cout<<"n="<<n<<endl;

//输入输出浮点型数据

double x;

cin>>x;

cout<<"x="<<x<<endl;

//输入提示

cout<<"n=";

cin>>n;

cout<<"n="<<n<<endl;

//多项输入

cout<<"c n x"<<endl;

cin>>c>>n>>x;

cout<<"c="<<c<<" n="<<n<<" x="<<x<<endl;

}

#include <iostream.h> //包含iostream.h头文件

main()

{

//声明整型变量

int a,b;

//从键盘上为整型变量赋值

cout<<"a=";

cin>>a;

cout<<"b=";

cin>>b;

//整型数的算术运算

cout<<a<<"+"<<b<<"="<<a+b<<endl;

cout<<a<<"-"<<b<<"="<<a-b<<endl;

cout<<a<<"\*"<<b<<"="<<a\*b<<endl;

cout<<a<<"/"<<b<<"="<<a/b<<endl;

cout<<a<<"%"<<b<<"="<<a%b<<endl;

//测试溢出

short n=32767,m; //n取short类型的最大值

cout<<"n="<<n<<endl;

m=n+1; //引起溢出

cout<<"n+1="<<m<<endl;

}

#include <iostream.h> //包含iostream.h头文件

main()

{

//声明变量，并初始化

int a=010,b=10,c=0X10;

//以十进制形式显示数据

cout<<"DEC:";

cout<<" a="<<a;

cout<<" b="<<b;

cout<<" c="<<c<<endl;

//以八进制形式显示数据

cout<<"OCT:";

cout<<oct; //指定八进制输出

cout<<" a="<<a;

cout<<" b="<<b;

cout<<" c="<<c<<endl;

//以十六进制形式显示数据

cout<<"HEX:";

cout<<hex; //指定十六进制输出

cout<<" a="<<a;

cout<<" b="<<b;

cout<<" c="<<c<<endl;

//八、十和十六进制数混合运算并输出

cout<<"a+b+c=";

cout<<dec; //恢复十进制输出

cout<<a+b+c<<endl;

//测试八、十和十六进制输入

cout<<"DEC:a="; cin>>a;

cout<<"OCT:b="; cin>>b;

cout<<"HEX:a="; cin>>c;

cout<<"DEC:"<<dec<<endl; //指定十进制输出

cout<<"a="<<a<<endl;

cout<<"b="<<b<<endl;

cout<<"c="<<c<<endl;

}

#include <iostream.h> //包含iostream.h头文件

#include<iomanip.h> // iomanip.h头文件包含setprecision()的定义

main()

{

//float型变量的声明、输入、计算和输出

float fx,fy;

cout<<"fx=";

cin>>fx;

cout<<"fy=";

cin>>fy;

cout<<fx<<"+"<<fy<<"="<<fx+fy<<endl;

cout<<fx<<"-"<<fy<<"="<<fx-fy<<endl;

cout<<fx<<"\*"<<fy<<"="<<fx\*fy<<endl;

cout<<fx<<"/"<<fy<<"="<<fx/fy<<endl<<endl;

//cout<<fx<<"%"<<fy<<"="<<fx%fy<<endl; Error!

//double型变量的声明、输入、计算和输出

float dx,dy;

cout<<"dx=";

cin>>dx;

cout<<"dy=";

cin>>dy;

cout<<dx<<"+"<<dy<<"="<<dx+dy<<endl;

cout<<dx<<"-"<<dy<<"="<<dx-dy<<endl;

cout<<dx<<"\*"<<dy<<"="<<dx\*dy<<endl;

cout<<dx<<"/"<<dy<<"="<<dx/dy<<endl<<endl;

//cout<<fx<<"%"<<fy<<"="<<fx%fy<<endl; Error!

//测试float和double类型数据的有效位

fx=10.0;fy=6.0;

float fz=fx/fy;

dx=10.0;dy=6.0;

double dz=dx/dy;

cout<<"fz=";

cout<<setprecision(20)<<fx<<"/"<<fy<<"="<<fz<<endl;

cout<<"dz=";

cout<<setprecision(20)<<dx<<"/"<<dy<<"="<<dz<<endl<<endl;;

//float型溢出

float x=3.5e14;

cout<<"x="<<x<<endl;

cout<<"x\*x="<<x\*x<<endl;

cout<<"x\*x\*x="<<x\*x\*x<<endl;

}

#include <iostream.h> //包含iostream.h头文件

main()

{

//字符类型变量的声明

char c1='A';

char c2;

//字符数据的运算及输出

c2=c1+32;

cout<<"c1="<<c1<<endl;

cout<<"c2="<<c2<<endl;

//输出字符及ASCII码

cout<<c1<<" : "<<int(c1)<<endl;

cout<<c2<<" : "<<int(c2)<<endl;

cout<<'$'<<" : "<<int('$')<<endl;

//输入字符

cout<<"c1 c2"<<endl;

cin>>c1>>c2;

cout<<"c1="<<c1<<" c2="<<c2<<endl;

}

#include <iostream.h> //包含iostream.h头文件

main()

{

char c1='\a',TAB='\t';

//阵铃一声

cout<<c1<<endl;

//使用水平制表符

cout<<1<<TAB<<2<<TAB<<3<<TAB<<4<<endl;

//使用双引号

cout<<"He said \"Thank you\"."<<endl;

//使用回车换行

cout<<"abc\n"<<"def"<<'\n';

}

#include <iostream.h> //包含iostream.h头文件

main()

{

//声明bool变量，并初始化

bool flag1=false,flag2=true;

//输出布尔常量和变量

cout<<"false:"<<false<<endl;

cout<<"true: "<<true<<endl;

cout<<"flag1="<<flag1<<endl;

cout<<"flag2="<<flag2<<endl;

//布尔变量的赋值和输出

int x=1;

flag1=x>0; //存放关系运算结果

cout<<"flag1="<<flag1<<endl;

flag2=flag1; //bool类型变量相互赋值

cout<<"flag2="<<flag2<<endl;

//布尔变量超界处理

flag1=100;

cout<<"flag1="<<flag1<<endl;

flag2=-100;

cout<<"flag2="<<flag2<<endl;

}

#include <iostream.h>

const double PI=3.1416; //声明常量(const变量)PI为3.1416

main()

{

//声明3个变量

double r,l,s;

//输入圆的半径

cout<<"r=";

cin>>r;

//计算圆的周长

l=2\*PI\*r;

cout<<"l="<<l<<endl;

//计算圆的面积

s=PI\*r\*r;

cout<<"s="<<s<<endl;

}

#include<iostream.h>

main()

{

//定义枚举类型，并指定其枚举元素的值

enum color {

RED=3,

YELLOW=6,

BLUE=9

};

//声明枚举变量a和b,并为枚举变量a赋初值

enum color a=RED;

color b; //合法，与C语言不同

// 输出枚举常量

cout<<"RED="<<RED<<endl;

cout<<"YELLOW="<<YELLOW<<endl;

cout<<"BLUE="<<BLUE<<endl;

//枚举变量的赋值和输出

b=a;

a=BLUE;

cout<<"a="<<a<<endl;

cout<<"b="<<b<<endl;

//a=100; 错误!

//a=6 也错误!

//枚举变量的关系运算

b=BLUE; // 枚举变量的赋值运算

cout<<"a<b="<<(a<b)<<endl;

}

#include <iostream.h>

const double PI=3.1416; //声明常量(const变量)PI为3.1416

main()

{

//声明3个变量

double r=3,l,s;

//计算圆的周长

l=2\*PI\*r;

cout<<"l="<<l<<endl;

//计算圆的面积

s=PI\*r\*r;

cout<<"s="<<s<<endl;

//验证赋值误差

int il,is;

il=l;

is=s;

cout<<"il="<<il<<endl;

cout<<"is="<<is<<endl;

}

#include <iostream.h>

main()

{

//变量声明

char c;

double x,y;

//测试自增

cout<<"++E and E++ :"<<endl;

c='B';

cout<<"c="<<++c<<endl; //输出c=C

c='B';

cout<<"c="<<c++<<endl; //输出c=B

x=1.5;

y=5+ ++x; //加号后的空格不能少

cout<<"y="<<y<<endl; //输出y=7.5

x=1.5;

y=5+x++;

cout<<"y="<<y<<endl; //输出y=6.5

cout<<"--------------------"<<endl;

//测试自减

cout<<"--E and E-- :"<<endl;

c='B';

cout<<"c="<<--c<<endl; //输出c=A

c='B';

cout<<"c="<<c--<<endl; //输出c=B

x=1.5;

y=5+--x;

cout<<"y="<<y<<endl; //输出y=5.5

x=1.5;

y=5+x--;

cout<<"y="<<y<<endl; //输出y=6.5

}

#include <iostream.h>

main()

{

int a=3, b=2;

//输出关系表达式

cout<<a<b<<endl;

cout<<(a<b)<<(a>b)<<(a>=b)<<(a==b)<<(a!=b)<<endl;

bool flag=2\*a<b+10;

cout<<"flag="<<flag;

}

#include <iostream.h>

main()

{

float a=3.5,b=2.1,c=0;

cout<<"a="<<a<<" b="<<b<<" c="<<c<<endl;

//与运算

cout<<"a&&b="<<(a&&b)<<endl;//输出1

cout<<"a&&c="<<(a&&c)<<endl;//输出0

//或运算

cout<<"a||b="<<(a||b)<<endl;//输出1

cout<<"a||c="<<(a||c)<<endl;//输出1

//非运算

cout<<"!a="<<!a<<endl<<"!c="<<!c<<endl;//输出0 1

//关系运算和逻辑运算

bool flag=a>=0 && a<=5; //变量a在[0,5]区间内

cout<<"a=>0 && a<=5="<<flag<<endl;//输出1

//算术运算、关系运算和逻辑运算

cout<<"a+5>2\*b+2||a<b+3="<<(a+5>2\*b+2||a<b+3)<<endl;//输出1

}

#include <iostream.h>

main()

{

//按位与运算

cout<<"24&12="<<(24&12)<<endl;

//按位异或运算

cout<<"24^12="<<(24^12)<<endl;

//按位或运算

cout<<"24|12="<<(24|12)<<endl;

//按位取反运算

cout<<"~24="<<(~24)<<endl;

//左移位运算

cout<<"5<<3="<<(5<<3)<<endl;

cout<<"-5<<3="<<(-5<<3)<<endl;

//右移位运算

cout<<"5>>3="<<(5>>3)<<endl;

cout<<"-5>>3="<<(-5>>3)<<endl;

}

#include <iostream.h>

main()

{

int a=1,b=1,c=3;

//显示a,b,c的值

cout<<"a="<<a<<" b="<<b<<" c="<<c<<endl;

//计算显示(1) b+=a+2\*c%5; 的结果

b+=a+2\*c%5; //相当于表达式语句 b=b+(a+2\*c%5);

cout<<"(1) b="<<b<<endl;

//计算显示(2) a<<=c-2\*b; 的结果

a=1,b=1,c=3;

a<<=c-2\*b; // 相当于表达式语句 a=a<<(c-2\*b);

cout<<"(2) a="<<a<<endl;

//计算显示(3) a\*=b=c=3;的结果

a=1,b=1,c=3;

a\*=b=c=3; //相当于语句组 c=3;b=c;a=a\*b;

cout<<"(3) a="<<a<<" b="<<b<<" c="<<c<<endl;

//计算显示(4) a+=b+=c;的结果

a=1,b=1,c=3;

a+=b+=c; //相当于语句组 b=b+c; a=a+b;

cout<<"(4) a="<<a<<" b="<<b<<" c="<<c<<endl;

//计算显示(5) a-=b=++c+2;的结果

a=1,b=1,c=3;

a-=b=++c+2; //相当于语句组 ++c;b=b+c+2;a=a-b;

cout<<"(5) a="<<a<<" b="<<b<<" c="<<c<<endl;

}

#include <iostream.h>

main()

{

//用 sizeof 计算各类种常量的字节长度

cout<<"sizeof('$')="<<sizeof('$')<<endl;

cout<<"sizeof(1)="<<sizeof(1)<<endl;

cout<<"sizeof(1.5)="<<sizeof(1.5)<<endl;

cout<<"sizeof(\"Good!\")="<<sizeof("Good!")<<endl;

//用sizeof 计算各类型变量的字节长度

int i=100;

char c='A';

float x=3.1416;

double p=0.1;

cout<<"sizeof(i)="<<sizeof(i)<<endl;

cout<<"sizeof(c)="<<sizeof(c)<<endl;

cout<<"sizeof(x)="<<sizeof(x)<<endl;

cout<<"sizeof(p)="<<sizeof(p)<<endl;

//用sizeof 计算表达式的字节长度

cout<<"sizeof(x+1.732)="<<sizeof(x+1.732)<<endl;

//用 sizeof 计算各类型的字节长度

cout<<"sizeof(char)="<<sizeof(char)<<endl;

cout<<"sizeof(int)="<<sizeof(int)<<endl;

cout<<"sizeof(float)="<<sizeof(float)<<endl;

cout<<"sizeof(double)="<<sizeof(double)<<endl;

//用sizeof 计算数组的字节长度

char str[]="This is a test.";

int a[10];

double xy[10];

cout<<"sizeof(str)="<<sizeof(str)<<endl;

cout<<"sizeof(a)="<<sizeof(a)<<endl;

cout<<"sizeof(xy)="<<sizeof(xy)<<endl;

//用sizeof 计算自定义类型的长度

struct st {

short num;

float math\_grade;

float Chinese\_grade;

float sum\_grade;

};

st student1;

cout<<"sizeof(st)="<<sizeof(st)<<endl;

cout<<"sizeof(student1)="<<sizeof(student1)<<endl;

}

#include <iostream.h>

main()

{

//声明变量语句中使用顺序运算

int x, y;

//计算中使用顺序运算

x=50;

y=(x=x-5, x/5);

cout<<"x="<<x<<endl;

cout<<"y="<<y<<endl;

}

#include <iostream.h>

main()

{

//测试表达式类型的转换

int n=100,m;

double x=3.791,y;

cout<<"n\*x="<<n\*x<<endl;

//赋值类型转换

m=x;

y=n;

cout<<"m="<<m<<endl;

cout<<"y="<<y<<endl;

//强制类型转换

cout<<"int(x)="<<int(x)<<endl;

cout<<"(int)x="<<(int)x<<endl;

cout<<"int(1.732+x)="<<int(1.732+x)<<endl;

cout<<"(int)1.732+x="<<(int)1.723+x<<endl;

cout<<"double(100)="<<double(100)<<endl;

}

#include <iostream.h>

main()

{

float a,b,s;

cout<<"a b"<<endl;

cin>>a>>b; //利用cin从键盘上为变量 a,b 赋值

s=a;

if (a<b) {

s=b; //if语句中只有这一个语句，可省略花括号

}

s=s\*s; //变量s中保存a,b中较大的一个数的平方

cout<<"s="<<s;

}

#include <iostream.h>

main()

{

int x,y;

cout<<"x=";

cin>>x;

if (x<=0) { //满足条件执行

y=2\*x;

cout<<"y="<<y; //输出结果

}

else { //不满足条件执行

y=x\*x;

cout<<"y="<<y; //输出结果

}

}

#include <iostream.h>

main()

{

int a,b,c;

int smallest;

cout<<"a b c"<<endl;

cin>>a>>b>>c;

if (a<=b) //外层条件语句

{

if (a<=c) //内层条件语句

smallest=a;

else

smallest=c;

}

else

{

if (b<=c) //内层条件语句

smallest=b;

else

smallest=c;

}

cout<<"Smallest="<<smallest<<endl;

}

#include <iostream.h>

main()

{

int score;

//从键盘上输入分数

cout<<"score=";

cin>>score;

//用带else if的条件语句判断处理

if (score<0 || score>100)

{

cout<<"The score is out of range!"<<endl;

}

else if (score>=90)

cout<<"Your grade is a A."<<endl;

else if (score>=80)

cout<<"Your grade is a B."<<endl;

else if (score>=70)

cout<<"Your grade is a C."<<endl;

else if (score>=60)

cout<<"Your grade is a D."<<endl;

else

cout<<"Your grade is a E."<<endl;

}

#include <iostream.h>

main()

{

int n;

cout<<"n=";

cin>>n;

if (n>=0 && n<=100 &&n%2==0)

cout<<"n="<<n<<endl;

else

cout<<"The "<<n<<" is out of range!"<<endl;

}

#include <iostream.h>

main()

{

int a,b,Max;

//输入数据

cout<<"a=";

cin>>a;

cout<<"b=";

cin>>b;

//找出较大值

Max=a>b?a:b;

cout<<"Max="<<Max<<endl;

}

#include <iostream.h>

main()

{

int a,b;

//输入数据

cout<<"a=";

cin>>a;

cout<<"b=";

cin>>b;

//除法判断

if (b!=0 && a%b==0) {

cout<<b<<" divides "<<a<<endl;

cout<<"a/b="<<a/b<<endl;

}

else

cout<<b<<" does not divide "<<a<<endl;

}

#include <iostream.h>

main()

{

//x,y 为操作数，c为运算符

int x,y,z;

char c1;

cin>>x>>c1>>y; //c1

//多路选择语句选择不同表达式计算语句

switch(c1) {

case '+':cout<<x<<"+"<<y<<"="<<x+y<<endl;

break;

case '-':cout<<x<<"-"<<y<<"="<<x-y<<endl;

break;

case '\*':cout<<x<<"\*"<<y<<"="<<x\*y<<endl;

break;

case '/':cout<<x<<"/"<<y<<"="<<x/y<<endl;

break;

case '%':cout<<x<<"%"<<y<<"="<<x%y<<endl;

break;

default :cout<<"Wrong !"<<endl; //当不符合上述情况时执行本子句

}

}

#include<iostream.h>

float x=365.5; //声明全局变量

main() {

int x=1,y=2;

double w=x+y;

{

double x=1.414,y=1.732,z=3.14;

cout<<"inner:x="<<x<<endl;

cout<<"inner:y="<<y<<endl;

cout<<"inner:z="<<z<<endl;

cout<<"outer:w="<<w<<endl;

cout<<"::x="<<::x<<endl; //访问重名的全局变量

}

cout<<"outer:x="<<x<<endl;

cout<<"outer:y="<<y<<endl;

cout<<"outer:w="<<w<<endl;

//cout<<"inner:z="<<z<<endl;无效

cout<<"::x="<<::x<<endl; //访问重名的全局变量

}

#include<iostream.h>

main() {

//显示1,2,3...10

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

cout<<i<<" ";

cout<<endl;

//显示10,9,8...1

for(int j=10;j>=1;j--)

cout<<j<<" ";

cout<<endl;

//显示1,3,5...9

for(int k=1;k<=10;k=k+2)

cout<<k<<" ";

cout<<endl;

//显示ABC...Z

for(char c='A';c<='Z';c++)

cout<<c;

cout<<endl;

//显示0,0.1,0.2...1.0

for(float x=0;x<=1.0;x=x+0.1)

cout<<x<<" ";

cout<<endl;

//显示0,0.1,0.2...1.0

for(float x1=0;x1<=1.0+0.1/2;x1=x1+0.1)

cout<<x1<<" ";

cout<<endl;

//计算s=1+2+3...+100

int s=0;

for(int n=1;n<=100;n++)

s=s+n;

cout<<"s="<<s<<endl;

}

#include<iostream.h>

main()

{

//计算s=1+2+3...+100

int s=0,n=1;

while(n<=100) {

s=s+n;

n++;

}

cout<<"s="<<s<<endl;

//累加键盘输入的数据

double x,sum=0.0;

cout<<"x=";

cin>>x;

while(x!=0) {

sum+=x;

cout<<"x=";

cin>>x;

}

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

}

#include<iostream.h>

main()

{

//计算s=1+2+3...+100

int s=0,n=0;

do {

n++;

s+=n;

}while(n<100);

cout<<"s="<<s<<endl;

//累加键盘输入的数据

double x,sum=0.0;

do {

cout<<"x=";

cin>>x;

sum+=x;

} while(x!=0);

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

}

#include<iostream.h>

main()

{

//计算和打印打印乘法九九表

for (int i=1;i<=9;i++) {

cout<<i;

for (int j=1;j<=9;j++)

cout<<'\t'<<i<<"\*"<<j<<"="<<i\*j;

cout<<endl;

}

}

#include<iostream.h>

main()

{

int x,sum=0;

//定义标号L1

L1: cout<<"x=";

cin>>x;

if (x==-1)

goto L2; //无条件转移语句，转到L2语句处

else

sum+=x;

goto L1; //无条件转移语句，转到L1语句处

//定义标号L2

L2: cout<<"sum="<<sum<<endl;

}

#include<iostream.h>

main()

{

//累加键盘输入的数据

double x,sum=0.0;

while(1) {

cout<<"x=";

cin>>x;

if (x<=0) break;

sum+=x;

}

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

}

#include<iostream.h>

main()

{

int i;

for (i=1;i<=20;i++)

{

if (i%3==0) //能被 3 整除的整数，返回进行下次循环

continue;

cout<<i<<" ";

}

cout<<endl;

}

#include<iostream.h>

main()

{

//声明数组和变量

int a[5],i,sum;

double avg;

//从键盘上循环为数组赋值

for (i=0;i<5;i++) {

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

cin>>a[i];

}

//直接显示数组元素

cout<<a[0]<<a[1]<<a[2]<<a[3]<<a[4]<<endl;

//利用for循环显示数组各元素的值

for (i=0;i<5;i++)

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

cout<<endl;

//计算数组元素之和,并显示计算结果

sum=a[0]+a[1]+a[2]+a[3]+a[4];

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

//利用循环计算数组的累加和

for (sum=0,i=0;i<5;i++)

sum+=a[i];

//显示累加和及平均值

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

avg=sum/5.0;

cout<<"avg="<<avg<<endl;

}

#include<iostream.h>

main()

{

int i,max,index,a[5];

//从键盘上为数组赋值

for (i=0;i<=4;i++)

{

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

cin>>a[i];

}

// 利用循环遍历数组，找出最大值的元素及其下标

max=a[0];

for (i=0;i<=4;i++)

{

if (max<a[i])

{

max=a[i];

index=i;

}

}

cout<<"\nMax="<<max<<" index="<<index;

}

#include<iostream.h>

#define size 5

main()

{

//声明变量

int i,j;

float t,a[size];

//从键盘上为数组赋值

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

{

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

cin>>a[i];

}

//对数组按从小到大顺序排序

for (i=0;i<size-1;i++)

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

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

{

t=a[i];

a[i]=a[j];

a[j]=t;

}

//显示排序结果

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

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

cout<<endl;

//输入要查找的数据

int value;

int found; //找到为1，否则为0

int low,high,mid;

for (i=1;i<=3;i++) {

cout<<"value=";

cin>>value;

//二分法查找数组a

found=0;

low=0;

high=size-1;

while(low<=high)

{

mid=(high+low)/2;

if (a[mid]==value)

{

found=1;

break;

}

if (a[mid]<value)

low=mid+1;

else

high=mid-1;

}

if (found)

cout<<"The valu found at:a["<<mid<<"]="<<a[mid]<<endl;

else

cout<<"The "<<value<<" is not found!"<<endl;

}

}

#include<iostream.h>

main()

{

//声明变量

int i,j;

float t,a[5];

//从键盘上为数组赋值

for (i=0;i<=4;i++)

{

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

cin>>a[i];

}

//对数组按从大到小顺序排序

for (i=0;i<=3;i++)

for (j=i+1;j<=4;j++)

if (a[i]<=a[j])

{

t=a[i];

a[i]=a[j];

a[j]=t;

}

//显示排序结果

for (i=0;i<=4;i++)

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

}

#include<iostream.h>

main()

{

//声明二维数组及变量

int a[2][3],i,j;

//从键盘上为数组a赋值

for (i=0;i<2;i++)

for (j=0;j<3;j++)

{

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

cin>>a[i][j];

}

//显示数组a

for (i=0;i<2;i++) {

for (j=0;j<3;j++)

{

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

}

cout<<endl;

}

//找出该数组的最大元素及其下标

int h,l,Max=a[0][0];

for (i=0;i<2;i++) {

for (j=0;j<3;j++)

{

if (Max<a[i][j]) {

Max=a[i][j];

h=i;

l=j;

}

}

}

cout<<"Max:"<<"a["<<h<<"]["<<l<<"]="<<a[h][l]<<endl;

}

#include<iostream.h>

main()

{

//声明字符数组和变量

char str[6];

int i;

//从键盘上输入字符串

cout<<"str=";

cin>>str;

cout<<str<<endl;

//按数组和下标变量两种方式显示字符数组

cout<<str<<endl;

for (i=0;i<6;i++)

cout<<str[i];

cout<<endl;

//字符串反向输出

for (i=5;i>=0;i--)

cout<<str[i];

cout<<endl;

//将字符数组变成大写字母后输出

for (i=0;i<=5;i++)

str[i]-=32; //小写字母转换成大写字母

cout<<str<<endl; //显示字符串

}

#include<iostream.h>

main()

{

//声明变量和指针变量

int a,b,c,\*ip;

//指针变量ip指向变量a

a=100;

ip=&a; //使指针变量 ip 指向变量a

cout<<"a="<<a<<endl;

cout<<"\*ip="<<\*ip<<endl;

cout<<"ip="<<ip<<endl;

//指针变量ip指向变量b

ip=&b; //使指针变量 ip 指向变量b

b=200;

cout<<"b="<<b<<endl;

cout<<"\*ip="<<\*ip<<endl;

cout<<"ip="<<ip<<endl;

//指针变量ip指向变量c

ip=&c; //使指针变量 ip 指向变量b

\*ip=a+b;

cout<<"c="<<c<<endl;

cout<<"\*ip="<<\*ip<<endl;

cout<<"ip="<<ip<<endl;

}

#include<iostream.h>

main()

{

//声明数组、变量和指针变量

int a[2][3],i,j;

int\* ip;

//从键盘上为数组a赋值

for (i=0;i<2;i++) //为数组a赋值

for (j=0;j<3;j++)

{

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

cin>>a[i][j];

}

//利用下标变量显示数组a

for (i=0;i<2;i++) {

for (j=0;j<3;j++)

{

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

}

cout<<endl;

}

//利用指针变量显示数组a

ip=&a[0][0];

for (i=0;i<2;i++) {

for (j=0;j<3;j++)

{

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

cout<<ip<<" ";

cout<<\*ip<<endl;

ip++;

}

}

}

#include<iostream.h>

main()

{

//声明数组、变量和指针变量

int a[]={1,2,3,4,5,6};

int \*ip1,\*ip2;

//测试指针的赋值运算

ip1=a;

ip2=ip1;

cout<<"\*ip1="<<(\*ip1)<<endl;

cout<<"\*ip2="<<(\*ip2)<<endl;

//测试指针的自增自减运算和组合运算

ip1++;

ip2+=4;

cout<<"\*ip1="<<(\*ip1)<<endl;

cout<<"\*ip2="<<(\*ip2)<<endl;

//测试指针变量之间的关系运算

int n=ip2>ip1;

cout<<"ip2>ip1="<<n<<endl;

cout<<"ip2!=NULL="<<(ip2!=NULL)<<endl;

//指针变量之间的减法

n=ip2-ip1;

cout<<"ip2-ip1="<<n<<endl;

}

#include<iostream.h>

main()

{

//声明字符型数组和指针变量

char str[10];

char \*strip=str;

//输入输出

cout<<"str=";

cin>>str; //用字符数组输入字符串

cout<<"str="<<str<<endl;

cout<<"strip="<<strip<<endl;

cout<<"strip=";

cin>>strip; //用字符指针变量输入字符串

cout<<"str="<<str<<endl;

cout<<"strip="<<strip<<endl;

//利用指针变量改变其指向字符串的内容

\*(strip+2)='l';

cout<<"str="<<str<<endl;

cout<<"strip="<<strip<<endl;

//动态为字符型指针变量分配内存

strip=new char(100);

cout<<"strip=";

cin>>strip; //用字符指针变量输入字符串

cout<<"str="<<str<<endl;

cout<<"strip="<<strip<<endl;

}

#include<iostream.h>

main()

{

// 声明用于存放运动员号码的数组

int h[]={1001,1002,1003,1004};

// 声明用于存放运动员成绩的数组

float x[]={12.3,13.1,11.9,12.1};

//声明用于存放运动姓名的字符型指针数组

char \*p[]={"Wang hua","Zhang jian","Li wei","Hua ming"};

//i,j,it是用做循环控制变量和临时变量

int i,j,it;

//ft 用做暂存变量

float ft;

//pt为字符型指针变量用做暂存指针变量

char \*pt;

//用选择法对数组x进行排序，并相应调整数组h和p中的数据

for (i=0;i<=3;i++)

for (j=i+1;j<=3;j++)

if (x[i]>=x[j]) {

ft=x[i],x[i]=x[j],x[j]=ft;

it=h[i],h[i]=h[j],h[j]=it;

pt=p[i],p[i]=p[j],p[j]=pt;

}

//以下打印排序结果

for (i=0;i<=3;i++)

cout<<h[i]<<" ,"<<p[i]<<" ,"<<x[i]<<endl;

}

#include<iostream.h>

main()

{

//声明指针数组

char \*colors[]={"Red","Blue","Yellow","Green"};

//指向指针的指针变量

char \*\*pt;

//通过指向指针的变量访问其指向的内容

pt=colors;

for (int i=0;i<=3;i++) {

cout<<"pt="<<pt<<endl;

cout<<"\*pt="<<\*pt<<endl;

cout<<"\*\*pt="<<\*\*pt<<endl;

pt++;

}

}

#include<iostream.h>

main()

{

//定义结构类型

struct books

{

char title[20];

char author[15];

int pages;

float price;

} ;

//声明结构变量

struct books Zbk={"VC++ ","Zhang",295,35.5};

books Wbk;

//对结构变量的输出

cout<<"Zbk:"<<endl;

cout<<Zbk.title <<endl;

cout<<Zbk.author<<endl;

cout<<Zbk.pages<<endl;

cout<<Zbk.price<<endl;

cout<<"--------------------"<<endl;

//对结构成员的运算

Zbk.pages+=10;

Zbk.price+=0.5;

cout<<"Zbk.pages="<<Zbk.pages<<endl;

cout<<"Zbk.price="<<Zbk.price<<endl;

cout<<"--------------------"<<endl;

//对结构变量的输入输出

cout<<"Wbk.title =";

cin>>Wbk.title;

cout<<"Wbk.author=";

cin>>Wbk.author;

cout<<"Wbk.pages=";

cin>>Wbk.pages;

cout<<"Wbk.price=";

cin>>Wbk.price;

cout<<"Wbk:"<<endl;

cout<<Wbk.title <<endl;

cout<<Wbk.author<<endl;

cout<<Wbk.pages<<endl;

cout<<Wbk.price<<endl;

cout<<"--------------------"<<endl;

//结构变量之间的相互赋值

books temp;

temp=Wbk;

cout<<"temp:"<<endl;

cout<<temp.title<<endl;

cout<<temp.author<<endl;

cout<<temp.pages<<endl;

cout<<temp.price<<endl;

}

#include<iostream.h>

main()

{

int i;

//定义结构类型

struct student {

int num;

char name[10];

float maths;

float physics;

float chemistry;

double total;

};

//声明结构数组st

student st[3];

//从键盘上为结构数组输入值

cout<<" num name maths physics chemistry "<<endl;

for (i=0;i<3;i++)

{

cout<<i+1<<" ";

cin>>st[i].num;

cin>>st[i].name;

cin>>st[i].maths;

cin>>st[i].physics;

cin>>st[i].chemistry;

}

//计算每个学生的总成绩

for (i=0;i<3;i++)

st[i].total=st[i].maths+st[i].physics+st[i].chemistry;

//输出结构数组各元素的值

for (i=0;i<3;i++)

{

cout<<"st["<<i<<"]: ";

cout<<st[i].num<<'\t';

cout<<st[i].name<<'\t';

cout<<st[i].maths<<'\t';

cout<<st[i].physics<<'\t';

cout<<st[i].chemistry<<'\t';

cout<<st[i].total<<endl;

}

}

#include<iostream.h>

main()

{

//定义结构类型

struct human {

char name[10];

int sex;

int age;

};

//声明结构变量和结构指针变量,并初始化

struct human x={"WangPing",1,30},\*p=NULL;

//结构指针变量指向对象

p=&x;

//显示结构变量的值

cout<<"x.name="<<x.name<<endl;

cout<<"x.sex="<<x.sex<<endl;

cout<<"x.age="<<x.age<<endl;

//利用结构指针显示结构对象中的数据

cout<<"(\*p).name="<<(\*p).name<<endl;

cout<<"(\*p).sex="<<(\*p).sex<<endl;

cout<<"(\*p).age="<<(\*p).age<<endl;

cout<<"p->name="<<p->name<<endl;

cout<<"p->sex="<<p->sex<<endl;

cout<<"p->age="<<p->age<<endl;

//通过结构指针为结构对象输入数据

cout<<"name:";

cin>>(\*p).name;

cout<<"sex:";

cin>>(\*p).sex;

cout<<"age:";

cin>>(\*p).age;

//显示结构变量的值

cout<<"x.name="<<x.name<<endl;

cout<<"x.sex="<<x.sex<<endl;

cout<<"x.age="<<x.age<<endl;

}

include<iostream.h>

main()

{

//定义结构类型

struct human {

char name[10];

int sex;

int age;

};

//声明结构变量和结构指针,并初始化

struct human x={"WangPing",1,30},\*p=&x;

//利用结构指针显示结构中的数据

cout<<"(\*p).name="<<(\*p).name<<endl;

cout<<"(\*p).sex="<<(\*p).sex<<endl;

cout<<"(\*p).age="<<(\*p).age<<endl;

cout<<"-------------------------"<<endl;

//利用new运算符为p分配内存

p=new human;

//从键盘上为p指向的结构对象赋值

cout<<"p->name=";

cin>>p->name;

cout<<"p->sex=";

cin>>p->sex;

cout<<"p->age=";

cin>>p->age;

cout<<"-------------------------"<<endl;

//显示p所指结构对象的值

cout<<"p->name="<<p->name<<endl;

cout<<"p->sex="<<p->sex<<endl;

cout<<"p->age="<<p->age<<endl;

cout<<"-------------------------"<<endl;

//显示结构变量的值

cout<<"x.name="<<x.name<<endl;

cout<<"x.sex="<<x.sex<<endl;

cout<<"x.age="<<x.age<<endl;

//释放p指向的内存

delete p;

}

#include<iostream.h>

main()

{

//定义结构类型

struct human {

char name[10];

int sex;

int age;

};

//声明结构数组和结构指针变量,并初始化

human x[]={{"WeiPing",1,30},{"LiHua",1,25},{"LiuMin",0,23}},\*p=NULL;

//用下标变量的输出结构数组的元素

for (int i=0;i<3;i++)

{

cout<<x[i].name<<'\t';

cout<<x[i].sex<<'\t';

cout<<x[i].age<<endl;

}

cout<<"----------------"<<endl;

//用结构指针输出结构数组的元素

for (p=x;p<=&x[2];p++)

{

cout<<p->name<<'\t';

cout<<p->sex<<'\t';

cout<<p->age<<endl;

}

}

#include<iostream.h>

main()

{

//定义一个包含指针成员的结构类型

struct test {

char \*str;

int \*ip;

} x;

//使用结构变量x中的整型指针ip

x.ip=new int; //分配1个单元

\*(x.ip)=100;

cout<<"x.ip:"<<x.ip<<'\t'<<\*(x.ip)<<endl;

cout<<"---------------"<<endl;

delete x.ip;

x.ip=new int[5]; //分配5个单元

for(int i=0;i<5;i++)

\*(x.ip+i)=100+i;

cout<<"x.ip:"<<endl;

for(i=0;i<5;i++)

cout<<x.ip+i<<'\t'<<(\*(x.ip+i))<<endl;

delete x.ip;

cout<<"---------------"<<endl;

//使用结构变量x中的字符型指针str

x.str=new char('A'); //分配1个单元

cout<<"x.str:"<<(\*x.str)<<endl;

cout<<"---------------"<<endl;

delete x.str;

x.str=new char[5]; //分配多个单元

\*x.str='G';

\*(x.str+1)='o';

\*(x.str+2)='o';

\*(x.str+3)='d';

\*(x.str+4)='\0';

cout<<"x.str:"<<x.str<<endl;

delete x.str;

cout<<"---------------"<<endl;

//在声明结构变量时初始化

test y={"Very Good!",NULL};

cout<<"y.str:"<<y.str<<endl;

cout<<"y.ip:"<<y.ip<<endl;

}

#include<iostream.h>

main()

{

//定义date结构

struct date

{

int year;

int month;

int day;

};

//定义baby结构

struct baby {

int num;

float weight;

date birthday; // date为结构类型

};

//声明baby结构变量并初始化

baby b1={10001,10,{2002,12,25}};

//下列是baby结构变量b1的引用。

cout<<"b1.num="<<b1.num<<endl;

cout<<"b1.weight="<<b1.weight<<endl;

cout<<"b1.birthday.year="<<b1.birthday.year<<endl;

cout<<"b1.birthday.month="<<b1.birthday.month<<endl;

cout<<"b1.birthday.day="<<b1.birthday.day<<endl;

cout<<"--------------------------"<<endl;

//声明baby结构变量temp,并进行赋值运算

baby temp;

temp=b1;

cout<<"temp.num="<<temp.num<<endl;

cout<<"temp.weight="<<temp.weight<<endl;

cout<<"temp.birthday.year="<<temp.birthday.year<<endl;

cout<<"temp.birthday.month="<<temp.birthday.month<<endl;

cout<<"temp.birthday.day="<<temp.birthday.day<<endl;

}

#include<iostream.h>

main()

{

//定义名为list的递归结构

struct list {

char name[10];

int sex;

int age;

list \*next; //成员next为指向其自身结构的指针

};

//使用递归结构变量

list L1={"WeiPing",1,35.5,NULL};

cout<<"L1:"<<endl;

cout<<"name\t"<<L1.name<<endl;

cout<<"sex\t"<<L1.sex<<endl;

cout<<"age\t"<<L1.age<<endl;

cout<<"next\t"<<L1.next<<endl;

}

#include<iostream.h>

main()

{

int i;

//定义名为student的递归结构

struct student {

char name[10];

int math;

int computer;

float sum;

student \*next; //next成员是指向自身的结构指针

};

//用student声明3个结构指针变量

struct student \*head,\*tail,\*temp;

//申请第1块数据，并设置各结构指针的初值

temp=new struct student; //申请内存

head=temp; // 头指针

tail=head; // 尾指针

//循环为链表输入数据

cout<<"\tname Math Computer"<<endl;

for (i=1;;i++) {

cout<<i<<"\t";

cin>>temp->name;

if (temp->name[0]!='\*')

{

cin>>temp->math>>temp->computer;

temp->sum=temp->math+temp->computer;

temp->next=NULL;

tail=temp; //设置链表尾指针

}

else

{

// 以下是输入结束处理

delete temp;

tail->next=NULL;

break;

}

//为下一个学生申请内存

temp->next=new struct student;

temp=temp->next; // 使处理指针temp指向新内存块

}

//将链表数据从头到尾打印出来

cout<<"--------------------"<<endl;

temp=head;

while (temp!=NULL) {

cout<<temp->name<<","<<temp->math<<",";

cout<<temp->computer<<","<<temp->sum<<endl;

temp=temp->next;

}

}

#include<iostream.h>

main()

{

int i;

//定义名为student的递归结构

struct student {

char name[10];

int math;

int computer;

float sum;

student \*forw; //forw成员是前指针

student \*next; //next成员是后指针

};

//用student声明3个结构指针变量

struct student \*head,\*tail,\*temp;

//申请第1块数据，并设置各结构指针的初值

temp=new struct student; //申请内存

head=temp; // 头指针

tail=head; // 尾指针

head->forw=NULL;

//循环为链表记录输入数据

cout<<"\tname Math Computer"<<endl;

for (i=1;;i++) {

cout<<i<<"\t";

cin>>temp->name;

if (temp->name[0]!='\*')

{

cin>>temp->math>>temp->computer;

temp->sum=temp->math+temp->computer;

temp->next=NULL;

tail=temp; //设置链表尾指针

}

else

{

// 以下是输入结束处理

delete temp;

tail->next=NULL;

break;

}

//为下一个学生申请内存

temp->next=new struct student;

temp->next->forw=temp; //设置前指针

temp=temp->next; //使处理指针temp指向新内存块

}

// 将链表数据从头到尾打印出来

cout<<"head------>tail:"<<endl;

temp=head;

while (temp!=NULL) {

cout<<temp->name<<","<<temp->math<<",";

cout<<temp->computer<<","<<temp->sum<<endl;

temp=temp->next;

}

// 将链表数据从尾到头打印出来

cout<<"tail------>head:"<<endl;

temp=tail;

while (temp!=NULL) {

cout<<temp->name<<","<<temp->math<<",";

cout<<temp->computer<<","<<temp->sum<<endl;

temp=temp->forw;

}

}

#include<iostream.h>

main()

{

int i;

//定义联合类型

union utag {

char c;

int k;

float x;

};

//声明联合变量

union utag u;

// 使用联合变量中的字符型成员

u.c='\*';

cout<<"u.c="<<u.c<<endl;

// 使用联合变量中的整型成员

u.k=1000;

cout<<"u.k="<<u.k<<endl;

// 使用联合变量中的浮点型成员

u.x=3.1416;

cout<<"u.x="<<u.x<<endl;

//声明联合变量时初始化

utag u1={'A'};

//同时引用联合变量的各成员

cout<<"u1.c="<<u1.c<<endl;

cout<<"u1.k="<<u1.k<<endl;

cout<<"u1.x="<<u1.x<<endl;

}

#include<iostream.h>

main()

{

//定义结构类型，并为声明的结构变量赋初值

struct s\_tag {

short i;

float x;

} sx={100,3.1416};

//定义联合类型，并为声明的联合变量赋初值

union u\_tag {

short i;

float x;

} ux={1000};

//输出结构类型和结构变量的有关信息

cout<<"sizeof(struct s\_tag)="<<sizeof(struct s\_tag)<<endl;

cout<<"sx.i="<<sx.i<<endl;

cout<<"sx.x="<<sx.x<<endl;

cout<<"sizeof(sx)="<<sizeof(sx)<<endl;

cout<<"------------------------------"<<endl;

//输出联合类型和联合变量的有关信息

cout<<"sizeof(union u\_tag)="<<sizeof(union u\_tag)<<endl;

ux.i=200;

cout<<"ux.i="<<ux.i<<endl; //输出联合变量ux 的i成员

ux.x=123.456;

cout<<"ux.x="<<ux.x<<endl; //输出联合变量ux 的x成员

cout<<"sizeof(ux)="<<sizeof(ux)<<endl;

}

#include<iostream.h>

main()

{

//自定义类型

typedef int ARRAY\_INT[50];

int i;

ARRAY\_INT a; //用自定义类型声明数组变量a

//以下为数组a赋值，并打印

for (i=0;i<50;i++) {

if (i%10==0) //每10个数换一次行

cout<<endl;

a[i]=i;

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

}

cout<<endl;

}

#include<iostream.h>

//定义结构类型

struct student

{

int num;

char name[20];

float grade;

};

void main(void)

{

//声明数组

int i,size;

char str[]="This is a string.";

int int\_values[] = {51, 23, 2, 44, 45,0,11};

float float\_values[] = {15.1, 13.3, 22.2, 10.4, 1.5};

student st\_arr[]={101,"WangLin",92,102,"LiPing",85,103,"ZhaoMin",88};

//显示char类型数组元素及其大小

size=sizeof(str) / sizeof(char);

cout<<"Number of elements in str: ";

cout<<size<<endl;

for(i=0;i<size;i++) {

cout<<str[i];

}

cout<<endl;

//显示int类型数组元素及其大小

size=sizeof(int\_values) / sizeof(int);

cout<<"Number of elements in int\_values: ";

cout<<size<<endl;

for(i=0;i<size;i++) {

cout<<int\_values[i]<<" ";

}

cout<<endl;

//显示float类型数组元素及其大小

size=sizeof(float\_values) / sizeof(float);

cout<<"Number of elements in float\_values: ";

cout<<size<<endl;

for(i=0;i<size;i++) {

cout<<float\_values[i]<<" ";

}

cout<<endl;

//显示student类型数组元素及其大小

size=sizeof(st\_arr) / sizeof(student);

cout<<"Number of elements in st\_arr: ";

cout<<size<<endl;

for(i=0;i<size;i++) {

cout<<st\_arr[i].num<<" ";

cout<<st\_arr[i].name<<" ";

cout<<st\_arr[i].grade<<endl;

}

}

#include<iostream.h>

//add()函数的定义，其有返回值

double add(double x,double y)

{

double z;

z=x+y;

cout<<x<<"+"<<y<<"="<<z<<endl;

return(z);

}

main()

{

double a=0.5,b=1.0;

//以不同参数形式调用函数add()

cout<<"add(1.5,2.5)="<<add(1.5,2.5)<<endl;

cout<<"add(a,b)="<<add(a,b)<<endl;

cout<<"add(2\*a,a+b)="<<add(2\*a,a+b)<<endl;

cout<<"----------------------"<<endl;

//以表达式方式调用函数add()

double c=2\*add(a,b);

cout<<"c="<<c<<endl;

cout<<"----------------------"<<endl;

//以语句式方式调用函数add()

add(2\*a,b);

cout<<"----------------------"<<endl;

//用其他类型参数调用函数add()

int n=1,m=2;

cout<<"add("<<n<<","<<m<<")="<<add(n,m)<<endl;

}

#include<iostream.h>

//定义符号函数sgn(),其返回值为int类型

int sgn(double x)

{

if (x>0) return(1); //返回出口1

if (x<0) return(-1); //返回出口2

return(0); //返回出口3

}

//main()函数定义

main()

{

double x;

int i;

for (i=0;i<=2;i++) {

cout<<"x=";

cin>>x;

cout<<"sgn("<<x<<")="<<sgn(x)<<endl;

}

}

#include<iostream.h>

//函数原型语句可以在这里

//定义main()函数

main()

{

//max()函数原型声明语句

float max(float,float);

//变量声明语句

float a,b,Max;

//输入参数并计算

cout<<"a=";

cin>>a;

cout<<"b=";

cin>>b;

Max=max(a,b); //调用max()函数

cout<<"max("<<a<<","<<b<<")="<<Max<<endl;

}

//定义max()函数

float max(float x,float y) //max()返回值类型为浮点型

{

float z;

z=(x>y)?x:y;

return(z);

}

#include<iostream.h>

//定义f()函数

f(int x,int y) //f()的参数以值方式传递

{

++x;

--y;

cout<<"x="<<x<<",y="<<y<<endl;

}

main() {

int a,b;

//设置实际参数的值

a=b=10;

//以变量为参数调用f()函数

f(a,b);

//验证实际参数的值

cout<<"a="<<a<<",b="<<b<<endl;

//以表达式参数形式调用f()函数

f(2\*a,a+b);

}

#include<iostream.h>

//定义公共结构类型

struct student {

int num;

char name[10];

float maths;

float physics;

float chemistry;

double total;

};

//定义结构输入函数

input\_Rec(struct student \*p) //参数为student类型的结构指针变量

{

cin>>p->num;

cin>>p->name;

cin>>p->maths;

cin>>p->physics;

cin>>p->chemistry;

}

//定义结构数据交换函数

swap\_Rec(struct student \*p1,struct student \*p2)

{

struct student x;

//交换两个记录的数据

x=\*p1;

\*p1=\*p2;

\*p2=x;

}

//输出结构的值

put\_Rec(struct student \*p)

{

cout<<p->num<<'\t';

cout<<p->name<<'\t';

cout<<p->maths<<'\t';

cout<<p->physics<<'\t';

cout<<p->chemistry<<'\t';

cout<<p->total<<endl;

}

//定义main()函数

main()

{

int i,j;

// 声明结构指针变量和结构数组

struct student \*p1,a[3];

//输入3个学生的数据并计算总成绩

cout<<"num\tname\tmaths\tphysics\tchemistry"<<endl;

for (p1=a;p1<=a+2;p1++) {

input\_Rec(p1);

p1->total=p1->maths+p1->physics+p1->chemistry;

}

//对3个学生的数据排序

for (i=0;i<=2;i++)

for (j=i+1;j<=2;j++)

if (a[i].total<a[j].total)

swap\_Rec(&a[i],&a[j]); //交换两个结构变量中的数据

cout<<"-------------------"<<endl; //输出一分界线

//输出排序后的结构数组

cout<<"num\tname\tmaths\tphysics\tchemistry\ttotal"<<endl;

for (p1=a;p1<=a+2;p1++)

put\_Rec(p1);

}

#include<iostream.h>

//定义结构

struct student {

char name[10];

float grade;

};

//交换student类型的数据

void swap(student &x,student &y) //swap的参数为引用传递方式

{

student temp;

temp=x;

x=y;

y=temp;

}

//返回student类型的引用，求优者

student& max(student &x,student &y) //swap的参数为引用传递方式

{

return (x.grade>y.grade?x:y);

}

//显示student类型的数据

void show(student &x) //show的参数为引用传递方式

{

cout<<x.name<<" "<<x.grade<<endl;

}

void main()

{

student a={"ZhangHua",351.5},b={"WangJun",385};

//显示a和b的数据

cout<<"a:";

show(a);

cout<<"b:";

show(b);

cout<<"------------------"<<endl;

//交换a和b的数据,并显示

swap(a,b);

cout<<"a:";

show(a);

cout<<"b:";

show(b);

cout<<"------------------"<<endl;

//计算和显示成绩高者

student t=max(a,b);

cout<<"Max:";

show(t);

}

#include <iostream.h>

//参数带有默认值的函数

disp(int x=1,int y=1,int z=1)

{

cout<<"参数1: "<<x<<endl;

cout<<"参数2: "<<y<<endl;

cout<<"参数3: "<<z<<endl;

cout<<"------------------"<<endl;

}

//main()函数中测试参数带有默认值的函数disp()

void main()

{

disp();

disp(10);

disp(10,20);

disp(10,20,30);

int a=1,b=2,c=3;

disp(a,b,c);

}

#include <iostream.h>

//计算字符串长度的函数

int str\_len(const char \*string)

{

//char \*temp=string; 编译报错！

//\*string='x'; 编译报错！

int i=0;

while (\*(string+i)!=NULL)

i++;

return i;

}

//main()函数中测试str\_len()

void main()

{

char a[]="ABCDE";

cout<<a<<"\t"<<str\_len(a)<<endl;

char \*str="Hello!";

cout<<str<<"\t"<<str\_len(str)<<endl;

cout<<"This is a test."<<"\t"<<str\_len("This is a test.")<<endl;

}

#include<iostream.h>

void disp(void); //这个函数声明语句不能少

//定义main()函数的参数和返回值类型是void类型

void main(void)

{

//调用void类型函数

disp();

}

//以下定义disp()函数

void disp(void) {

cout<<" You are welcome."<<endl;

}

#include<iostream.h>

//函数原型语句

int abs(int x);

long abs(long x);

float abs(float x);

//main()函数的定义

void main(void)

{

//声明变量

int i1=32767,i2=-32767;

long l1=456789,l2=-456789;

float x1=1.1234,x2=-1.1234;

//直接在cout输出中调用函数

cout<<abs(i1)<<","<<abs(i2)<<endl;

cout<<abs(l1)<<","<<abs(l2)<<endl;

cout<<abs(x1)<<","<<abs(x2)<<endl;

}

//定义int型的abs()函数

int abs(int x) {

if (x<0)

return(-x);

else

return(x);

}

//定义long型的abs()函数

long abs(long x) {

if (x<0)

return(-x);

else

return(x);

}

//定义float型 abs函数

float abs(float x) {

if (x<0.0)

return(-x);

else

return(x);

}

#include<iostream.h>

//max()为内联函数

inline int max(int x,int y) //注意inline关键字

{

return x>y?x:y;

}

//定义main()函数

main()

{

int a=3,b=5,c;

c=max(a,b);

cout<<"max("<<a<<","<<b<<")="<<c<<endl;

cout<<"max("<<15<<","<<11<<")="<<max(15,11)<<endl;

}

#include<iostream.h>

main()

{

//函数原型声明

int fact(int x);

int n,sn;

//依次从键盘上输入3个正整型数据计算它们的阶乘

for (int i=1;i<=3;i++)

{

cout<<i<<" n=";

cin>>n;

sn=fact(n);

cout<<n<<"!="<<sn<<endl;

}

}

//以下是采用递归方法定义的fact()函数

int fact(int x)

{

if (x==0) return(1);

return(x\*fact(x-1)); //此处又调用了它自身

}

#include<iostream.h>

//带参数的main()函数

int main(int argc,char \*argv[])

{

int i;

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

cout<<i<<":"<<argv[i]<<endl;

return 0;

}

#include<iostream.h>

//用函数原型声明要使用的函数

void show\_array1(int\*,int);

void show\_array2(int a[],int);

void sort(int\*,int);

main()

{

//声明数组并初始化

int a[]={2,4,6,1,3,5};

int b[3][3]={{2,4,6},{1,3,5},{0,1,2}};

//显示数组的值

cout<<"show\_array1(int\*,int):"<<endl;

show\_array1(a,6);

show\_array1(&b[0][0],3\*3);

//用sort1排序并显示

cout<<"sort(int\*,int) and show\_array1(int\*,int): "<<endl;

sort(a,6);

show\_array1(a,6);

sort(&b[0][0],3\*3);

show\_array1(&b[0][0],9);

//显示数组的值

cout<<"show\_array2(int a[],int):"<<endl;

show\_array2(a,6);

show\_array2(&b[0][0],3\*3);

}

//显示数组,用指针当参数

void show\_array1(int \*p,int size) {

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

cout<<\*(p+i)<<" ";

cout<<endl;

}

//显示数组,用数组当参数

void show\_array2(int a[],int size) {

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

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

cout<<endl;

}

//对数组按从大到小顺序排序

void sort(int \*p,int size) {

int t;

for (int i=0;i<size-1;i++)

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

if (\*(p+i)<=\*(p+j))

{

t=\*(p+i);

\*(p+i)=\*(p+j);

\*(p+j)=t;

}

}

#include<iostream.h>

//定义结构

struct student {

char name[10];

float grade;

};

//更改student数据的grade成员,参数形式为引用

void change(student &x,float grade)

{

x.grade=grade;

}

//更改student数据的grade成员,参数形式为指针

void change1(student \*p,float grade)

{

p->grade=grade;

}

//更改student类型的数据,普通参数形式

void change2(student x,float grade)

{

x.grade=grade;

}

//显示student类型的数据,参数形式为引用

void show(student &x)

{

cout<<x.name<<" "<<x.grade<<endl;

}

//在main()函数中，测试对结构的处理函数

void main()

{

student a={"ZhangHua",351.5};

//显示a的数据

show(a);

//用change修改分数,并显示

cout<<"change(student &x,float grade):"<<endl;

change(a,360);

show(a);

//用change1修改分数,并显示

cout<<"change1(student \*p,float grade):"<<endl;

change1(&a,375);

show(a);

//用change2修改分数,并显示

cout<<"change2(student x,float grade):"<<endl;

change2(a,380.5);

show(a);

}

#include<iostream.h>

//定义函数计算数组的和和平均值

void calculate(int a[],int size,int& sum,float& average)

{

sum=0;

for (int i=0;i<size;i++) {

sum+=a[i];

}

average=sum/size;

}

//定义显示数组的函数

void put\_arr(int a[],int size)

{

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

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

cout<<endl;

}

main()

{

//声明数组并初始化

int asize,bsize;

int a[]={2,4,6,1,3,5};

int b[]={1,3,5,7,9,11,13,15};

//显示数组的值

asize=sizeof(a)/sizeof(int);

cout<<"put\_arr(a,asize):"<<endl;

put\_arr(a,asize);

bsize=sizeof(b)/sizeof(int);

cout<<"put\_arr(b,bsize):"<<endl;

put\_arr(b,bsize);

//计算数组的和和平均值

float a\_ave,b\_ave;

int a\_sum,b\_sum;

cout<<"calculate(a,asize,a\_sum,a\_ave):"<<endl;

calculate(a,asize,a\_sum,a\_ave);

cout<<"a\_sum="<<a\_sum;

cout<<" a\_ave="<<a\_ave<<endl;

cout<<"calculate(b,bsize,b\_sum,b\_ave):"<<endl;

calculate(b,bsize,b\_sum,b\_ave);

cout<<"b\_sum="<<b\_sum;

cout<<" b\_ave="<<b\_ave<<endl;

}

#include<iostream.h>

//参数为函数指针的函数

int get\_result(int a, int b, int (\*sub)(int,int))

{

int r;

r=sub(a,b);

return r;

}

//计算最大值

int max(int a, int b)

{

cout<<"In max"<<endl;

return((a > b) ? a: b);

}

//计算最小值

int min(int a, int b)

{

cout<<"In min"<<endl;

return((a < b) ? a: b);

}

//求和

int sum(int a, int b)

{

cout<<"In sum"<<endl;

return(a+b);

}

//测试指向函数的指针

void main(void)

{

int a,b,result;

//测试3次

for (int i=1;i<=3;i++) {

cout<<"Input a and b :";

cin>>a>>b;

cout<<i<<"\tget\_result("<<a<<","<<b<<", &max):"<<endl;

result =get\_result(a, b, &max);

cout<<"Max of "<<a<<" and "<<b<<" is "<<result<<endl;

result = get\_result(a, b, &min);

cout<<"Min of "<<a<<" and "<<b<<" is "<<result<<endl;

result = get\_result(a, b, &sum);

cout<<"Sum of "<<a<<" and "<<b<<" is "<<result<<endl;

}

}

#include<iostream.h>

#include<stdio.h>

#define size 3

//定义book结构类型

struct book

{

char title[20];

char author[15];

int pages;

float price;

};

//book结构的输入函数

input\_book(book& bk,char \*name)

{

cout<<name<<":"<<endl;

cout<<"title:";

cin>>bk.title;

cout<<"author:";

cin>>bk.author;

cout<<"pages:";

cin>>bk.pages;

cout<<"price:";

cin>>bk.price;

}

//book结构的输出函数

output\_book(book& bk,char \*name)

{

cout<<name<<": ";

cout<<bk.title<<" ";

cout<<bk.author<<" ";

cout<<bk.pages<<" ";

cout<<bk.price<<endl;

}

void main(void)

{

//声明变量和结构数组

int i;

char str[20];

book bk[size];

//输入结构数组

for(i=0;i<size;i++) {

sprintf(str,"bk[%d]",i+1);

input\_book(bk[i],str);

}

//显示结构数组

for(i=0;i<size;i++) {

sprintf(str,"bk[%d]",i+1);

output\_book(bk[i],str);

}

}

#include<iostream.h>

//声明全局变量并初始化

extern int a[]={1,2,3};

extern float p=3.14;

//在show()函数中使用外部变量

show() {

int i;

cout<<"In show():"<<endl;

cout<<"p="<<p<<endl;

cout<<"a[]: ";

for (i=0;i<=2;i++)

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

cout<<endl;

//cout<<"y="<<y<<endl; 编译出错！

}

//声明外部变量并初始化

int y=5678;

//在main()函数中使用外部变量

main()

{

//声明局部变量

int i,p=100;

//显示重名变量

cout<<"In main():"<<endl;

cout<<"p="<<p<<endl;

//显示全局变量

cout<<"::p="<<::p<<endl;

cout<<"a[]: ";

for (i=0;i<=2;i++)

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

cout<<endl;

cout<<"y="<<y<<endl; //编译正确！

show(); //调用函数

}

#include <iostream.h>

//使用静态变量的计数器函数

count1()

{

//声明静态变量i，并置初值为0。i在count()中局部可见

static int i=0;

return(++i);

}

//使用局部变量的计数器函数

count2()

{

int i=0;

return(++i);

}

//在main()函数中调用count()函数

main()

{

int i;

//调用count1()10次

cout<<"count1():"<<endl;

for (i=1;i<=12;i++)

cout<<count1()<<" ";

cout<<endl;

//调用count2()10次

cout<<"count2():"<<endl;

for (i=1;i<=12;i++)

cout<<count2()<<" ";

cout<<endl;

}

// p1-851.cpp 为main()函数文件

#include<iostream.h>

main()

{

int i,s=0;

extern int fact(int x);

for (i=2;i<=6;i=i+2)

s+=fact(i);

cout<<"s="<<s<<endl;

}

// p1-852.cpp为计算阶乘函数文件

//定义fact()函数为外部(extern)函数

extern int fact(int x)

{

int i,t=1;

if(x==0) return(1);

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

t\*=i;

return(t);

}

#include<iostream.h>

#include<stdio.h>

#include<string.h>

#include<process.h>

main() {

//声明变量

FILE \*fp1;

char str[80];

//从键盘上任意输入一个字符串

cout<<"Inupt a string:";

cin.getline(str,80);

//以写入方式打开d.dat文件

if ((fp1=fopen("d.dat","w"))==NULL)

{

cout<<"\nCould not open the file."<<endl;

cout<<"Exiting program."<<endl;

exit(1); //结束程序执行

}

// 写"流"文件

fputs(str,fp1);

fputs("\n",fp1);

fclose(fp1); //关闭文件

// 以读方式打开d.dat文件

if ((fp1=fopen("d.dat","r"))==NULL)

{

cout<<"\nCould not open the file."<<endl;

cout<<"Exiting program."<<endl;

exit(1); //结束程序执行

}

// 循环从"流"文件读取字符,并显示

char ch;

while ((ch=fgetc(fp1))!=EOF)

cout<<ch;

cout<<endl;

fclose(fp1); //关闭文件

}

#include<iostream.h>

#include <process.h>

#include<stdio.h>

#include<conio.h>

void main(void) {

//变量声明

char ch;

FILE \*fp1;

//以写入方式打开d.dat文件

if ((fp1=fopen("d.dat","w"))==NULL) {

cout<<"\nCould not open the file."<<endl;

cout<<"Exiting program."<<endl;

exit(1); //结束程序执行

}

//循环从键盘上读取字符,写入"流"文件

cout<<"char:"<<endl;

cin>>ch;

while (ch!='\*') {

fputc(ch,fp1); //将字符写到fp1指向的"流"文件中

cin>>ch;

}

fclose(fp1); //关闭文件

// 以读方式打开d.dat文件

if ((fp1=fopen("d.dat","r"))==NULL)

{

cout<<"\nCould not open the file."<<endl;

cout<<"Exiting program."<<endl;

exit(1); //结束程序执行

}

// 循环从"流"文件读取字符,并显示

while ((ch=fgetc(fp1))!=EOF)

cout<<ch<<" ";

cout<<endl;

fclose(fp1); //关闭文件

}

#include<iostream.h>

#include<stdio.h>

#include<string.h>

#include<process.h>

main() {

//声明变量

int i=0;

char p[100]; // 声明输入缓冲区

FILE \*fp1; // 声明文件指针变量

//以写入方式打开d.dat文件

if ((fp1=fopen("d.dat","w"))==NULL)

{

cout<<"\nCould not open the file."<<endl;

cout<<"Exiting program."<<endl;

exit(1); //结束程序执行

}

// 写文件操作

for (i=1;;i++) { //无条件循环

cout<<i<<" string:";

cin>>p; //从键盘上输入数据

if (stricmp(p,"end")) { //如果输入的字符串为end，则结束循环

fputs(p,fp1); //写入文件操作

fputs("\n",fp1);

}

else

break; //退出循环

}

fclose(fp1); //关闭文件

// 以读方式打开d.dat文件

if ((fp1=fopen("d.dat","r"))==NULL)

{

cout<<"\nCould not open the file."<<endl;

cout<<"Exiting program."<<endl;

exit(1); //结束程序执行

}

// 循环从文件读取字符,并显示

while (fgets(p,100,fp1)!=NULL)

cout<<p;

fclose(fp1); //关闭文件

}

#include<iostream.h>

#include<stdio.h>

#include<string.h>

#include<process.h>

#include<stdlib.h>

#define MAX 10

main() {

//声明变量

int i,n;

FILE \*fp1; // 声明文件指针变量

//以写入方式打开d.dat文件

if ((fp1=fopen("d.dat","w"))==NULL)

{

cout<<"\nCould not open the file."<<endl;

cout<<"Exiting program."<<endl;

exit(1); //结束程序执行

}

// 写文件操作

for (i=1;i<=MAX;i++) {

n=rand(); //产生1个整数随机数

putw(n,fp1);

cout<<n<<" ";

}

cout<<endl<<"--------------------"<<endl;

fclose(fp1); //关闭文件

// 以读方式打开d.dat文件

if ((fp1=fopen("d.dat","r"))==NULL)

{

cout<<"\nCould not open the file."<<endl;

cout<<"Exiting program."<<endl;

exit(1); //结束程序执行

}

// 循环从"流"文件读取字符,并显示

while ((n=getw(fp1))!=EOF)

cout<<n<<" ";

fclose(fp1); //关闭文件

}

#include<iostream.h>

#include<stdio.h>

#include<string.h>

#include<process.h>

#include<stdlib.h>

#define MAX 3

main() {

//定义结构类型

struct student {

int num;

char name[10];

float grade;

};

//声明数组和变量

student st[3];

int i;

FILE \*fp1; // 声明文件指针变量

//以写入方式打开d.dat文件

if ((fp1=fopen("d.dat","w"))==NULL)

{

cout<<"\nCould not open the file."<<endl;

cout<<"Exiting program."<<endl;

exit(1); //结束程序执行

}

//从键盘上读数据,写入文件

cout<<" num name grade"<<endl;

for (i=0;i<MAX;i++) {

cout<<i+1<<" ";

cin>>st[i].num;

cin>>st[i].name;

cin>>st[i].grade;

fprintf(fp1,"%d %s %f\n",st[i].num,st[i].name,st[i].grade);

}

fclose(fp1); //关闭文件

// 以读方式打开d.dat文件

if ((fp1=fopen("d.dat","r"))==NULL)

{

cout<<"\nCould not open the file."<<endl;

cout<<"Exiting program."<<endl;

exit(1); //结束程序执行

}

// 循环从"流"文件读取字符,并显示

student t;

while ((fscanf(fp1, "%d %s %f",&t.num,t.name,&t.grade))!=EOF) {

cout<<t.num<<" ";

cout<<t.name<<" ";

cout<<t.grade<<endl;

}

fclose(fp1); //关闭文件

}

#include<iostream.h>

#include <process.h>

#include <stdlib.h>

#include <stdio.h>

int main(void)

{

FILE \*fpd,\*fpw; // 声明FILE结构指针变量

unsigned char dw;

int i=0;

//以二进制读方式打开Calc.exe文件

if((fpd=fopen("C:\WINDOWS\Calc.exe", "rb"))==NULL)

{

cout<<"\nCould not open the file."<<endl;

cout<<"Exiting program."<<endl;

exit(1); //结束程序执行

}

// 以二进制写方式打开test.exe文件

if((fpw=fopen("test.exe", "wb+"))==NULL)

{

cout<<"\nCould not open the file."<<endl;

cout<<"Exiting program."<<endl;

exit(1); //结束程序执行

}

// 二进制文件读写操作，每次指定读写1个字节

while(!feof(fpd)) { //使用feof()判断文件尾

fread(&dw, 1, 1, fpd);

fwrite(&dw, 1, 1, fpw);

}

// 关闭文件

fclose(fpd);

fclose(fpw);

//执行Calc.exe和Calc.exe文件

cout<<"1 Run C:\WINDOWS\Calc.exe"<<endl;

system("C:\WINDOWS\Calc.exe");

cout<<"-------------------"<<endl;

cout<<"2 Run test.exe!"<<endl;

system("test.exe");

}

#include<iostream.h>

#include <process.h>

#include<stdio.h>

#include<conio.h>

void main(void) {

//声明变量

int i;

char ch;

FILE \*fp1;

//以写入方式打开d.dat文件

if ((fp1=fopen("d.dat","w"))==NULL) {

cout<<"\nCould not open the file."<<endl;

cout<<"Exiting program."<<endl;

exit(1); //结束程序执行

}

//循环从键盘上读取字符,写入文件

cout<<"char:";

cin>>ch;

while (ch!='\*') {

fputc(ch,fp1); //将字符写到fp1指向的"流"文件中

cin>>ch;

}

cout<<"--------------------"<<endl;

fclose(fp1); //关闭文件

//以读方式打开d.dat文件

if ((fp1=fopen("d.dat","r"))==NULL)

{

cout<<"\nCould not open the file."<<endl;

cout<<"Exiting program."<<endl;

exit(1); //结束程序执行

}

//循环从文件读取字符,并显示

while ((ch=fgetc(fp1))!=EOF)

cout<<ch;

cout<<endl<<"--------------------"<<endl;

//以下按倒序方式读取文件中的字符，并显示

for (i=-1;;i--) {

fseek(fp1,i,2); //设置文件指针，偏移量为i,相对文件尾

if ((ch=fgetc(fp1))!=EOF)

cout<<ch;

else

break;

}

cout<<endl<<"--------------------"<<endl;

//以下读取"流"文件中偶数位置上的字符，并打印

long position;

for (i=0;;i=i+2) {

fseek(fp1,i,0); //设置文件指针，偏移量为i,相对文件头

position=ftell(fp1);

if ((ch=fgetc(fp1))==EOF) //遇到文件尾，则退出，否则打印读取的字符

break;

else {

cout<<position<<" :"<<ch<<endl;

}

}

cout<<endl;

fclose(fp1); //关闭文件

}

#include<iostream.h>

#include<stdio.h>

#include<process.h>

#include<stdlib.h>

#define MAX 5

//显示数组的数据

void show\_array(double x[],int size) {

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

cout<<x[i]<<" ";

cout<<endl;

}

//main函数测试数组数据的文件读写

int main(void)

{

//声明变量

FILE \*fp; // 声明FILE结构指针变量

int i;

double a[MAX]={1.0,1.2,1.4,1.6,1.8};

//显示数组a的数据

cout<<"a:";

show\_array(a,MAX);

//打开d.dat文件

if ((fp=fopen("d.dat","wb+"))==NULL)

{

cout<<"\nCould not open the file."<<endl;

cout<<"Exiting program."<<endl;

exit(1); //结束程序执行

}

//以单个元素对数组进行文件读操作

for(i=0;i<MAX;i++) {

fwrite(&a[i], sizeof(double), 1, fp);

}

rewind(fp); //恢复读写指针的位置

//以单个元素对数组进行文件读操作

double b[MAX];

for(i=0;i<MAX;i++) {

if (!feof(fp)) //使用feof()判断文件尾

fread(&b[i], sizeof(double), 1, fp);

else

break;

}

cout<<"b:";

show\_array(b,MAX);//显示数组b的数据

fclose(fp); // 关闭文件

//打开d1.dat文件

if ((fp=fopen("d1.dat","wb+"))==NULL)

{

cout<<"\nCould not open the file."<<endl;

cout<<"Exiting program."<<endl;

exit(1); //结束程序执行

}

//将数组当成数据块写入文件

fwrite(&a, sizeof(double), MAX, fp);

rewind(fp); //恢复读写指针的位置

//将数组当成数据块从文件中读取

double c[MAX];

if (!feof(fp)) //使用feof()判断文件尾

fread(&c, sizeof(double),MAX,fp);

cout<<"c:";

show\_array(c,MAX); //显示数组c的数据

fclose(fp); // 关闭文件

}

#include<iostream.h>

#include<stdio.h>

#include<process.h>

#include<stdlib.h>

#define MAX 5

//定义结构类型

struct student {

int num;

char name[20];

float grade;

};

//显示student结构数据

void show\_str(student a,char \*name) {

cout<<name<<":"<<endl;

cout<<a.num<<" "<<a.name<<" "<<a.grade;

cout<<endl;

}

//main函数测试结构数据的文件读写

int main(void)

{

//声明变量

FILE \*fp;

//声明FILE结构指针变量

student st={1001,"ZhangBin",85.5};

//显示st结构数据

show\_str(st,"st");

//打开d.dat文件

if ((fp=fopen("d.dat","wb+"))==NULL)

{

cout<<"\nCould not open the file."<<endl;

cout<<"Exiting program."<<endl;

exit(1); //结束程序执行

}

//用fprintf()函数写结构数据到文件

fprintf(fp,"%d %s %f",st.num,st.name,st.grade);

rewind(fp); //恢复读写指针的位置

//用fscanf()函数读文件中的数据赋值给结构并显示

student temp;

fscanf(fp, "%d %s %f",&temp.num,temp.name,&temp.grade);

show\_str(temp,"temp");

cout<<"-----------------------"<<endl;

fclose(fp); // 关闭文件

//将结构数据当成数据块进行读写

if ((fp=fopen("d1.dat","wb+"))==NULL) //打开d1.dat文件

{

cout<<"\nCould not open the file."<<endl;

cout<<"Exiting program."<<endl;

exit(1); //结束程序执行

}

//声明结构数组并初始化

int i;

student starr[3]={{101,"WangPing",92},{102,"Li",85},{103,"LiuMin",97}};

//显示结构数组

for(i=0;i<3;i++)

show\_str(starr[i],"starr");

//将结构数组当成数据块写入文件

fwrite(starr, sizeof(student), 3, fp);

rewind(fp); //恢复读写指针的位置

//按数据块从文件中读取数据赋值给结构数组

student temp\_arr[3];

if (!feof(fp)) //使用feof()判断文件尾

fread(temp\_arr, sizeof(student),3,fp);

for(i=0;i<3;i++)

show\_str(temp\_arr[i],"temp\_arr");

fclose(fp); // 关闭文件

}

#include<stdio.h>

#include<stdlib.h>

#include<iostream.h>

int main(void)

{

//声明变量

char ch;

char str[20];

int n;

float x;

//用stdin从键盘上输入数据

fprintf(stdout,"ch str\n");

fscanf(stdin,"%c %s",&ch,str);

fprintf(stdout,"n x \n");

fscanf(stdin,"%d %f",&n,&x);

cout<<"----------------"<<endl;

//输出显示

fprintf(stdout,"ch=%c str=%s",ch,str);

fprintf(stdout,"\nn=%d x=%f",n,x);

cout<<endl;

}

#include <stdio.h>

void main( void )

{

int c;

/\* Create an error by writing to standard input. \*/

putc( 'A', stdin );

if( ferror( stdin ) )

{

perror( "Write error" );

clearerr( stdin );

}

/\* See if read causes an error. \*/

printf( "Will input cause an error? " );

c = getc( stdin );

if( ferror( stdin ) )

{

perror( "Read error" );

clearerr( stdin );

}

}

#include<iostream.h>

#include<math.h> //此预处理指令不可少

const double HD=3.1415926/180;

main() {

cout<<"x\tsin(x)"<<endl;

for (int i=0;i<=180;i=i+30)

cout<<i<<"\t"<<sin(i\*HD)<<endl;

}

#include<iostream.h>

//以下是几个简单宏替换预处理指令

#define YES 1

#define PI 3.1415926

#define RAD PI/180

#define MESG "This is a string."

//以下是主程序

main() {

//以下各语句使用了宏替换

cout<<"YES="<<YES<<endl;

if (YES)

cout<<"PI="<<PI<<endl;

cout<<"RAD="<<RAD<<endl;

cout<<MESG<<endl;

}

#include<iostream.h>

//以下为带参数宏替换的预处理指令

#define PRINT(k) cout<<(k)<<endl;

#define MAX(a,b) ((a)>(b) ? (a):(b))

main()

{

int i=3,j=2;

//MAX(a,b)宏替换的使用

cout<<"MAX(10,12)="<<MAX(10,12)<<endl;

cout<<"MAX(i,j)="<<MAX(i,j)<<endl;

cout<<"MAX(2\*i,j+3)="<<MAX(2\*i,j+3)<<endl;

//PRINT(k)宏替换的使用

PRINT(5);

PRINT(MAX(7,i\*j));

}

#include<iostream.h>

#define PI 3.1416

main() {

int i=100;

#if 1

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

#endif

#ifdef PI

cout<<"1 PI="<<PI<<endl;

#endif

#ifndef PI

cout<<"2 PI="<<PI<<endl; //此语句不被编译执行

#endif

}

#include<iostream.h>

const int MAX=5; //假定栈中最多保存5个数据

//定义名为stack的类，其具有栈功能

class stack {

//数据成员

float num[MAX]; //存放栈数据的数组

int top; //指示栈顶位置的变量

public:

//成员函数

void init(void) { top=0; } //初始化函数

void push(float x) //入栈函数

{

if (top==MAX){

cout<<"Stack is full !"<<endl;

return;

};

num[top]=x;

top++;

}

float pop(void) //出栈函数

{

top--;

if (top<0){

cout<<"Stack is underflow !"<<endl;

return 0;

};

return num[top];

}

}

//以下是main()函数，其用stack类创建栈对象，并使用了这些对象

main(void)

{

//声明变量和对象

int i;

float x;

stack a,b; //声明(创建)栈对象

//以下对栈对象初始化

a.init();

b.init();

//以下利用循环和push()成员函数将2,4,6,8,10依次入a栈对象

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

a.push(2\*i);

//以下利用循环和pop()成员函数依次弹出a栈中的数据并显示

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

cout<<a.pop()<<" ";

cout<<endl;

//以下利用循环和push()成员函数将键盘输入的数据依次入b栈

cout<<"Please input five numbers."<<endl;

for (i=1; i<=MAX; i++) {

cin>>x;

b.push(x);

}

//以下利用循环和pop()成员函数依次弹出b栈中的数据并显示

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

cout<<b.pop()<<" ";

}

#include<iostream.h>

const int MAX=5; //假定栈中最多保存5个数据

//定义名为stack的具有栈功能的类

class stack {

//数据成员

float num[MAX]; //存放栈数据的数组

int top; //指示栈顶位置的变量

public:

//成员函数

stack(void) //初始化函数

{

top=0;

cout<<"Stack initialized."<<endl;

}

void push(float x) //入栈函数

{

if (top==MAX){

cout<<"Stack is full !"<<endl;

return;

};

num[top]=x;

top++;

}

float pop(void) //出栈函数

{

top--;

if (top<0){

cout<<"Stack is underflow !"<<endl;

return 0;

};

return num[top];

}

}

//以下是main()函数，其用stack类创建栈对象，并使用了这些对象

main(void)

{

//声明变量和对象

int i;

float x;

stack a,b; //声明(创建)栈对象并初始化

//以下利用循环和push()成员函数将2,4,6,8,10依次入a栈

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

a.push(2.0\*i);

//以下利用循环和pop()成员函数依次弹出a栈中的数据并显示

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

cout<<a.pop()<<" ";

cout<<endl;

//以下利用循环和push()成员函数将键盘输入的数据依次入b栈

cout<<"Please input five numbers."<<endl;

for (i=1; i<=MAX; i++) {

cin>>x;

b.push(x);

}

//以下利用循环和pop()成员函数依次弹出b栈中的数据并显示

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

cout<<b.pop()<<" ";

cout<<endl;

}

#include<iostream.h>

const int MAX=5; //假定栈中最多保存5个数据

//定义名为stack的具有栈功能的类

class stack {

//数据成员

float num[MAX]; //存放栈数据的数组

int top; //指示栈顶位置的变量

public:

//成员函数

stack(char c) //初始化函数

{

top=0;

cout<<"Stack "<<c<<" initialized."<<endl;

}

void push(float x) //入栈函数

{

if (top==MAX){

cout<<"Stack is full !"<<endl;

return;

};

num[top]=x;

top++;

}

float pop(void) //出栈函数

{

top--;

if (top<0){

cout<<"Stack is underflow !"<<endl;

return 0;

};

return num[top];

}

}

//以下是main()函数，其用stack类创建栈对象，并使用了这些对象

main(void)

{

//声明变量和对象

int i;

float x;

stack a('a'),b('b'); //声明(创建)栈对象并初始化

//以下利用循环和push()成员函数将2,4,6,8,10依次入a栈

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

a.push(2.0\*i);

//以下利用循环和pop()成员函数依次弹出a栈中的数据并显示

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

cout<<a.pop()<<" ";

cout<<endl;

}

#include<iostream.h>

main()

{

//定义一个名为student的类

class student {

int num;

char \*name;

float grade;

public:

//定义构造函数

student(int n,char \*p,float g): num(n),name(p),grade(g){}

display(void) {

cout<<num<<" ,"<<name<<","<<grade<<endl;

}

};

student a(1001,"Liming",95),b(1002,"ZhangHua",96.5); //创建对象，并初始化

//student c; 错误，没提供参数

a.display(); //显示对象a中的数据

b.display(); //显示对象b中的数据

}

#include <iostream.h>

#include <stdlib.h>

//定义timer类

class timer{

long minutes;

public:

//无参数构造函数

timer(void) {

minutes =0;

};

//字符指针参数的构造函数

timer(char \*m) {

minutes = atoi(m);

};

//整数类型的构造函数

timer(int h, int m) {

minutes = 60\*h+m ;

};

//双精度浮点型构造函数

timer(double h) {

minutes = (int) 60\*h ;

};

long getminutes(void) { return minutes ; };

};

//main()函数的定义

main(void)

{

//使用double类型的构造函数创建对象

timer start(8.30),finish(17.30);

cout<<"finish(17.30)-start(8.30)=";

cout<<finish.getminutes()-start.getminutes()<<endl;

//使用char指针类型的构造函数创建对象

timer start0("500"),finish0("800"); //创建对象

cout<<"finish0(\"800\")-start0(\"500\")=";

cout<<finish0.getminutes()-start0.getminutes()<<endl;

//使用无参数构造函数和整型构造函数创建对象

timer start1;

timer finish1(3,30);

cout<<"finish1(3,30)-start1=";

cout<<finish1.getminutes()-start1.getminutes()<<endl;

return 0;

}

#include <iostream.h>

//定义rect类

class rect {

int length;

int width;

int area;

public:

rect(int l=1,int w=1)

{

length=l;

width=w;

area=length\*width;

}

void show\_rect(char \*name)

{

cout<<name<<":"<<endl;

cout<<"length="<<length<<endl;

cout<<"width="<<width<<endl;

cout<<"area="<<area<<endl;

}

};

//测试使用rect类

void main(void)

{

//用rect类创建对象

rect a;

rect b(2);

rect c(2,3);

//调用对象的函数显示对象中的数据

a.show\_rect("a");

b.show\_rect("b(2)");

c.show\_rect("c(2,3)");

}

#include<iostream.h>

const int MAX=5; //假定栈中最多保存5个数据

//定义名为stack的具有栈功能的类

class stack {

//数据成员

double num[MAX]; //存放栈数据的数组

int top; //指示栈顶位置的变量

public:

//成员函数

stack(char \*name) //构造函数

{

top=0;

cout<<"Stack "<<name<<" initialized."<<endl;

}

~stack(void) //析构函数

{

cout << "Stack destroyed." << endl; //显示信息

}

void push(double x) //入栈函数

{

if (top==MAX){

cout<<"Stack is full !"<<endl;

return;

};

num[top]=x;

top++;

}

double pop(void) //出栈函数

{

top--;

if (top<0){

cout<<"Stack is underflow !"<<endl;

return 0;

};

return num[top];

}

}

//以下是main()函数，其用stack类创建栈对象，并使用了这些对象

main(void)

{

double x;

//声明(创建)栈对象并初始化

stack a("a"),b("b");

//以下利用循环和push()成员函数将2,4,6,8,10依次入a栈

for (x=1; x<=MAX; x++)

a.push(2.0\*x);

//以下利用循环和pop()成员函数依次弹出a栈中的数据并显示

cout<<"a: ";

for (int i=1; i<=MAX; i++)

cout<<a.pop()<<" ";

cout<<endl;

//从键盘上为b栈输入数据,并显示

for(i=1;i<=MAX;i++) {

cout<<i<<" b:";

cin>>x;

b.push(x);

}

cout<<"b: ";

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

cout<<b.pop()<<" ";

cout<<endl;

}

#include<iostream.h>

#define MAX 5

//定义stack类接口

class stack{

int num[MAX];

int top;

public:

stack(char \*name); //构造函数原型

~stack(void); //析构函数原型

void push(int n);

int pop(void);

};

//main()函数测试stack类

main(void)

{

int i,n;

//声明对象

stack a("a"),b("b");

//以下利用循环和push()成员函数将2,4,6,8,10依次入a栈

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

a.push(2\*i);

//以下利用循环和pop()成员函数依次弹出a栈中的数据，并显示

cout<<"a: ";

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

cout<<a.pop()<<" ";

cout<<endl;

//从键盘上为b栈输入数据,并显示

for(i=1;i<=MAX;i++) {

cout<<i<<" b:";

cin>>n;

b.push(n);

}

cout<<"b: ";

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

cout<<b.pop()<<" ";

cout<<endl;

return 0;

}

//-------------------------

// stack成员函数的定义

//-------------------------

//定义构造函数

stack::stack(char \*name)

{

top=0;

cout << "Stack "<<name<<" initialized." << endl;

}

//定义析构函数

stack::~stack(void)

{

cout << "stack destroyed." << endl; //显示信息

}

//入栈成员函数

void stack::push(int n)

{

if (top==MAX){

cout<<"Stack is full !"<<endl;

return;

};

num[top]=n;

top++;

}

//出栈成员函数

int stack::pop(void)

{

top--;

if (top<0){

cout<<"Stack is underflow !"<<endl;

return 0;

};

return num[top];

}

#include<iostream.h>

//定义一个全部为public:模式的类

class ex

{

public:

int value;

void set(int n) {

value=n;

}

int get(void) {

return value;

}

};

//测试使用ex类

main()

{

ex a; //创建对象

//以下通过成员函数访问对象数据

a.set(100);

cout<<"a.get()=";

cout<<a.get()<<endl;

//以下直接访问对象的数据成员

a.value=200;

cout<<"a.value=";

cout<<a.value<<endl;

}

#include <iostream.h>

// ex\_class类接口定义

class ex\_class

{

private:

int iv;

double dv;

public:

ex\_class(void);

ex\_class(int n,double x);

void set\_ex\_class(int n,double x);

void show\_ex\_class(char\*);

};

//定义ex\_class类的构造函数

ex\_class::ex\_class(void):iv(1), dv(1.0) { }

ex\_class::ex\_class(int n,double x):iv(n), dv(x) { }

//定义ex\_class类的成员函数

void ex\_class::set\_ex\_class(int n,double x)

{

iv=n;

dv=x;

}

void ex\_class::show\_ex\_class(char \*name)

{

cout<<name<<": "<<endl;

cout <<"iv=" <<iv<< endl;

cout <<"dv=" <<dv<< endl;

}

//使用ex\_class类

void main(void)

{

ex\_class obj1;

obj1.show\_ex\_class("obj1");

obj1.set\_ex\_class(5,5.5);

obj1.show\_ex\_class("obj1");

ex\_class obj2(100,3.14);

obj2.show\_ex\_class("obj2");

obj2.set\_ex\_class(2000,1.732);

obj2.show\_ex\_class("obj2");

}

#include<iostream.h>

//定义一个含有static数据成员的类

class ex

{

static int num; //static数据成员

public:

ex() {num++;}

~ex() {num--;}

disp\_count() {

cout<<"The current instances count:";

cout<<num<<endl;

}

};

int ex::num=0; //设置static数据成员的初值

//main()函数测试ex类

main()

{

ex a;

a.disp\_count();

ex \*p;

p=new ex;

p->disp\_count();

ex x[10];

x[0].disp\_count();

delete p;

a.disp\_count();

}

#include<iostream.h>

//定义一个含有static数据成员的类

class ex

{

static int num; //static数据成员

public:

ex() {num++;}

~ex() {num--;}

static disp\_count(void) //static成员函数

{

cout<<"The current instances count:";

cout<<num<<endl;

}

};

int ex::num=0; //设置static数据成员的初值

//main()函数测试ex类

main()

{

ex a;

a.disp\_count();

ex \*p;

p=new ex;

p->disp\_count();

ex x[10];

ex::disp\_count(); //直接用类作用域符引用静态成员函数

delete p;

ex::disp\_count(); //直接用类作用域符引用静态成员函数

}

#include <iostream.h>

class ex\_class {

int value;

public:

ex\_class(int n) {

value=n;

cout << "Stack initialized." << endl;

}

~ex\_class() {

cout << "The Object destroyed." <<endl;

}

void set\_value(int n);

void show\_val(char \*name);

} ;

//在类外定义内联成员函数

inline void ex\_class::set\_value(int n) {

value=n;

}

//在类外定义非内联成员函数

void ex\_class::show\_val(char \*name) {

cout<<name<<": ";

cout<<value<<endl;

}

//在main()函数中测试ex\_class类

main(void)

{

//创建对象x和y

ex\_class x(100),y(200);

//显示对象的数据

x.show\_val("x");

y.show\_val("y");

//设置新值给对象

x.set\_value(1);

y.set\_value(2);

//显示对象的数据

x.show\_val("x");

y.show\_val("y");

return 0;

}

#include <iostream.h>

//定义空类empty

class empty

{

};

//在main()函数中用空类创建对象

main()

{

empty a,\*p; //编译通过

cout<<"Test a empty class."<<endl;

}

#include<iostream.h>

//用struct关键字定义ex\_class类

struct ex\_class {

ex\_class(int n=1): value(n) {}

void set\_value(int n) {

value=n;

}

show\_obj(char \*name) {

cout<<name<<": "<<value<<endl;

}

private:

int value;

}

//测试 ex\_class类

main()

{

//用ex\_class创建对象

ex\_class a,b(3);

a.show\_obj("a");

b.show\_obj("b");

a.set\_value(100);

b.set\_value(200);

a.show\_obj("a");

b.show\_obj("b");

}

#include <iostream.h>

#include<string.h>

//定义双亲（parent）类

class parent {

char f\_name[20];

char m\_name[20];

char tel[10];

public:

// parent类的构造函数，其带有缺省值

parent(char \*p1="",char \*p2="",char \*p3="") {

strcpy(f\_name,p1);

strcpy(m\_name,p2);

strcpy(tel,p3);

}

//显示parent对象的数据

show\_parent(void) {

cout<<"The parent:"<<endl;

cout<<" father's name:"<<f\_name<<endl;

cout<<" mother's name:"<<m\_name<<endl;

cout<<" tel:"<<tel<<endl;

}

};

//定义student类

class student {

int num;

char name[20];

float grade;

parent pt;

public:

// student类的构造函数

student(int n,char \*str,float g,class parent t) {

num=n;

strcpy(name,str);

grade=g;

pt=t;

}

//显示student对象的数据

show\_student(void) {

cout<<"num:"<<num<<endl;

cout<<"name:"<<name<<endl;

cout<<"grade:"<<grade<<endl;

pt.show\_parent();

}

};

//main()函数测试student类的对象

main(void)

{

//创建双亲对象

parent p1("ZhangHua","LiLan","83665215");

//创建学生对象

student st(10001,"ZhangHui",91.5,p1);

//显示学生信息

cout<<"p1:"<<endl;

p1.show\_parent();

//显示学生信息

cout<<"st:"<<endl;

st.show\_student();

}

#include <iostream.h>

#include <stdlib.h>

//定义timer类

class timer{

long minutes;

public:

//定义重载成员函数

settimer(char \*m) {

minutes = atoi(m);

};

//定义重载成员函数

settimer(int h, int m) {

minutes = 60\*h+m ;

};

//定义重载成员函数

settimer(double h) {

minutes = (int) 60\*h ;

};

long getminutes(void) { return minutes; };

};

//main()函数的定义

main(void){

timer start,finish; //创建对象

//使用重载成员函数

start.settimer(8,30);

finish.settimer(9,40);

cout<<"finish.settimer(9,40)-start.settimer(8,30):";

cout<<finish.getminutes()-start.getminutes()<<endl;

//使用重载成员函数

start.settimer(2.0);

finish.settimer("180");

cout<<"finish.settimer(\"180\")-start.settimer(2.0):";

cout<<finish.getminutes()-start.getminutes()<<endl;

return 0;

}

#include <iostream.h>

//定义复数类

class complex{

float real; //实部

float image; //虚部

public:

//重载的运算符"+"的原型

complex operator+ (complex right);

//重载赋值运算符"="的定义

complex operator= (complex right);

void set\_complex(float re, float im);

void put\_complex(char \*name);

};

//重载加法运算符"+"的定义

complex complex::operator+ (complex right) {

complex temp;

temp.real = this->real + right.real;

temp.image = this->image + right.image;

return temp;

}

//重载加赋值运算符"="的定义

complex complex::operator= (complex right) {

this->real = right.real;

this->image = right.image;

return \*this;

}

//定义set\_complex()成员函数

void complex::set\_complex(float re, float im) {

real = re;

image = im;

}

//定义put\_complex()成员函数

void complex::put\_complex(char \*name) {

cout<<name<<": ";

cout << real << ' ';

if (image >= 0.0 ) cout << '+';

cout << image << "i\n";

}

//在main()函数中使用complex类的对象

main(void)

{

complex A, B, C; //创建复数对象

//设置复数变量的值

A.set\_complex(1.2, 0.3);

B.set\_complex(-0.5, -0.8);

//显示复数数据

A.put\_complex("A");

B.put\_complex("B");

//赋值运算，显示结果

C = A;

C.put\_complex("C=A");

//加法及赋值运算，显示结果

C = A + B;

C.put\_complex("C=A+B");

return 0;

}

// Example of the friend class

#include <iostream.h>

//定义YourClass类，

class YourClass

{

//指定YourOtherClass是它的友元类

friend class YourOtherClass;

private:

int num;

public:

YourClass(int n){num=n;}

display(char \*YCname){

cout<<YCname<<".num :";

cout<<num<<endl;

}

};

//定义YourOtherClass，它是YourClass类的友元类

class YourOtherClass

{

public:

//使用YourClass类的私有成员

void disp1(YourClass yc,char \*YCname){

cout<<YCname<<".num :";

cout<<yc.num<<endl;

}

//使用YourClass类的公共成员

void disp2(YourClass yc,char\* YCname){

yc.display(YCname);

}

};

//在main()函数中创建和使用YourClass和YourOtherClass类对象

main(void)

{

//声明YourClass类对象

YourClass a(10),b(100);

//显示a和b对象的值

cout<<"YourClass:"<<endl;

a.display("a");

b.display("b");

//声明YourOtherClass类对象

YourOtherClass temp;

//通过temp显示a和b对象的值

cout<<"YourOtherClass:"<<endl;

temp.disp1(a,"a");

temp.disp2(b,"b");

}

#include<iostream.h>

//Y类的不完全定义

class Y;

//X类的定义

class X {

public:

void disp(Y py,char \*name); //成员函数原型

};

//定义Y类

class Y {

//声明本类的友元函数

//X类的disp()为本例的友元函数

friend void X::disp(Y py,char \*name);

//普通函数putY() 为本例的友元函数

friend void putY(Y& yc,char \*name);

private: //私有成员

int num;

dispY(char \*name){

cout<<name<<".num="<<num<<endl;

}

public: //公共成员函数

Y(int n){

num=n;

}

};

//X类成员函数的实现部分

void X::disp(Y py,char \*name){

cout<<"In X::disp():"<<endl;

py.dispY(name); //访问Y类的私有函数

}

//普通函数putY()的定义

void putY(Y& yc,char \*name){

cout<<"In getY:"<<endl;

yc.dispY(name);

cout<<name<<".num=";

cout<<yc.num<<endl;

}

//在main()函数测试X和Y类的功能

main()

{

//创建Y和X类的对象

Y y1(100),y2(200);

X x;

//不可用Y类对象的私有成员函数显示

//y1.dispY("y1");

//y2.dispY("y2");

//调用X类对象的友元函数显示

x.disp(y1,"y1");

x.disp(y2,"y2");

//用getY函数显示Y类的对象显示

putY(y1,"y1");

putY(y2,"y2");

}

#include <iostream.h>

//定义日期类

class Date

{

//定义友元重载输入运算符函数

friend istream& operator >> (istream& input,Date& dt );

//定义友元重载输出运算符函数

friend ostream& operator<< (ostream& output,Date& dt );

int mo, da, yr;

public:

Date(void){ //无参数构造函数

yr = 0;

mo = 0;

da = 0;

}

Date( int y, int m, int d ) //带参数构造函数

{

yr = y;

mo = m;

da = d;

}

};

//定义">>"运算符重载函数

istream& operator >> ( istream& input, Date& dt )

{

cout<<"Year:";

input>>dt.yr;

cout<<"Month:";

input>>dt.mo;

cout<<"Day:";

input>>dt.da;

return input;

}

//定义"<<"运算符重载函数

ostream& operator<< ( ostream& output, Date& dt )

{

output<< dt.yr << '/' << dt.mo << '/' << dt.da<<endl;

return output;

}

//在main()函数中测试Date类的插入（<<）和提取（>>）运算符

void main()

{

//声明对象

Date dt1(2002,5,1),dt2;

//显示dt1对象

cout<<dt1;

//对dt2对象进行输入和输出

cin>>dt2;

cout<<dt2;

}

#include<iostream.h>

//定义ex类

class ex\_class

{

int a;

double b;

public:

ex\_class(int n=1,double x=1.0):a(n),b(x) {}

void show\_value(char \*name) {

cout<<name<<" :"<<endl;

cout<<"a="<<a<<endl;

cout<<"b="<<b<<endl;

}

};

//定义main()函数

main()

{

//创建ex\_class的对象并显示

ex\_class obj1,obj2(100,3.5);

obj1.show\_value("obj1");

obj2.show\_value("obj2");

//创建ex\_class的指针变量

ex\_class \*p;

//p指向obj1并显示

p=&obj1;

p->show\_value("p->obj1");

//p指向obj2并显示

p=&obj2;

(\*p).show\_value("(\*p)obj2");

//p指向动态创建的对象并显示

p=new ex\_class;

p->show\_value("p->new");

delete p; //删除对象

}

#include<iostream.h>

//基类Box

class Box {

int width,height;

public:

void SetWidth(int w) {

width=w;

}

void SetHeight(int h) {

height=h;

}

int GetWidth() {return width;}

int GetHeight() {return height;}

};

//派生类ColoredBox

class ColoredBox:public Box

{

int color;

public:

void SetColor(int c){

color=c;

}

int GetColor() {return color;}

};

// 在main()中测试基类和派生类

main(void)

{

//声明并使用ColoredBox类的对象

ColoredBox cbox;

cbox.SetColor(3); //使用自己的成员函数

cbox.SetWidth(150); //使用基类的成员函数

cbox.SetHeight(100); //使用基类的成员函数

cout<<"cbox:"<<endl;

cout<<"Color:"<<cbox.GetColor()<<endl; //使用自己的成员函数

cout<<"Width:"<<cbox.GetWidth()<<endl; //使用基类的成员函数

cout<<"Height:"<<cbox.GetHeight()<<endl; //使用基类的成员函数

//cout<<cbox.width; Error!

}

#include<iostream.h>

//基类First

class First {

int val1;

public:

SetVal1(int v) {

val1=v;

}

void show\_First(void) {

cout<<"val1="<<val1<<endl;

}

};

//派生类Second

class Second:private First { //默认为private模式

int val2;

public:

void SetVal2(int v1,int v2) {

SetVal1(v1); //可见，合法

val2=v2;

}

void show\_Second(void) {

// cout<<"val1="<<val1<<endl; 不能访问First私有成员

show\_First();

cout<<"val2="<<val2<<endl;

}

};

main() {

Second s1;

//s1.SetVal1(1); //不可见，非法

s1.SetVal2(2,3); //合法

//s1.show\_First(); //不可见，非法

s1.show\_Second();

}

#include<iostream.h>

//基类First

class First {

int val1;

public:

SetVal1(int v) {

val1=v;

}

void show\_First(void) {

cout<<"val1="<<val1<<endl;

}

};

//派生类Second

class Second:public First { //默认为private模式

int val2;

public:

void SetVal2(int v1,int v2) {

SetVal1(v1); //可见，合法

val2=v2;

}

void show\_Second(void) {

// cout<<"val1="<<val1<<endl; 不能访问First私有成员

show\_First();

cout<<"val2="<<val2<<endl;

}

};

main() {

Second s1;

//调用Second类定义的成员函数

s1.SetVal2(2,3);

cout<<"s1.show\_Second():"<<endl;

s1.show\_Second();

//调用First类定义的成员函数

s1.SetVal1(10);

cout<<"s1.show\_First():"<<endl;

s1.show\_First();

}

#include<iostream.h>

//定义最低层基类，它作为其他类的基类

class First {

int val1;

public:

First(void) {

cout<<"The First initialized"<<endl;

}

};

//定义派生类，它作为其他类的基类

class Second :public First {

int val2;

public:

Second(void) {

cout<<"The Second initialized"<<endl;

}

};

//定义最上层派生类

class Three :public Second {

int val3;

public:

Three() {

cout<<"The Three initialized"<<endl;

}

};

//定义各基类的对象，测试构造函数的执行情况

//定义各基类的对象，测试构造函数的执行情况

main() {

cout<<"First f1;"<<endl;

First f1;

cout<<"Second s1;"<<endl;

Second s1;

cout<<"Three t1;"<<endl;

Three t1;

}

#include<iostream.h>

//定义基类First

class First {

int num;

float grade;

public:

//构造函数带参数

First(int n,float v ) : num(n),grade(v)

{

cout<<"The First initialized"<<endl;

}

DispFirst(void) {

cout<<"num="<<num<<endl;

cout<<"grade="<<grade<<endl;

}

};

//定义派生类Second

class Second :public First {

double val;

public:

//无参数构造函数，要为基类的构造函数设置参数

Second(void):First(10000,0) {

val=1.0;

cout<<"The Second initialized"<<endl;

}

//带参数构造函数，为基类的构造函数设置参数

Second(int n,float x,double dx):First(n,x) {

val=dx;

cout<<"The Second initialized"<<endl;

}

Disp(char \*name){

cout<<name<<".val="<<val<<endl;

DispFirst();

}

};

//main()函数中创建和使用派生类对象

main() {

//调用派生类的无参数构造函数

cout<<"Second s1;"<<endl;

Second s1;

cout<<"s1.Disp(\"s1\");"<<endl;

s1.Disp("s1");

//调用派生类的有参数构造函数

cout<<"Second s2(10002,95.7,3.1415926); "<<endl;

Second s2(10002,95.7,3.1415926);

cout<<"s2.Disp(\"s2\");"<<endl;

s2.Disp("s2");

}

#include<iostream.h>

//定义最低层基类First，它作为其他类的基类

class First {

int val1;

public:

First() {

cout<<"The First initialized"<<endl;

}

~First() {

cout<<"The First destroyed"<<endl;

}

};

//定义派生类Second，它作为其他类的基类

class Second :public First { //默认为private模式

int val2;

public:

Second() {

cout<<"The Second initialized"<<endl;

}

~Second() {

cout<<"The Second destroyed"<<endl;

}

};

//定义最上层派生类Three

class Three :public Second {

int val3;

public:

Three() {

cout<<"The Three initialized"<<endl;

}

~Three() {

cout<<"The Three destroyed"<<endl;

}

};

//main()函数中测试构造函数和析构函数的执行情况

main() {

Three t1;

cout<<"---- Use the t1----"<<endl;

}

#include<iostream.h>

//基类

class First {

int val1;

protected:

void SetVal1(int v) {

val1=v;

}

public:

show\_First(void) {

cout<<"val1="<<val1<<endl;

}

};

//派生类

class Second:public First {

int val2;

protected:

void SetVal2(int v) {

SetVal1(v); //使用First 基类的保护成员

val2=v;

}

public:

show\_Second(void) {

show\_First();

cout<<"val2="<<val2<<endl;

}

};

//派生类

class Third:public Second {

int val3;

public:

void SetVal3(int n) {

SetVal1(n); //使用First 基类的保护成员

SetVal2(n); //使用Second基类的保护成员

val3=n;

}

show\_Third(void) {

show\_Second();

cout<<"val3="<<val3<<endl;

}

};

//main()函数的定义

main(void)

{

First f1;

//f1.SetVal1(1); 不可访问

Second s1;

//s1.SetVal1(1); 不可访问

//s1.SetVal2(2); 不可访问

Third t1;

//t1.SetVal1(1); 不可访问

//t1.SetVal2(2); 不可访问

t1.SetVal3(10);

//显示t1对象的数据

cout<<"t1.show\_Third();"<<endl;

t1.show\_Third();

cout<<"t1.show\_Second();"<<endl;

t1.show\_Second();

cout<<"t1.show\_First();"<<endl;

t1.show\_First();

}

#include <iostream.h>

enum Color {Red,Yellow,Green,White};

//圆类Circle的定义

class Circle {

float radius;

public:

Circle(float r) {radius=r;}

float Area() {

return 3.1416\*radius\*radius;

}

};

//桌子类Table的定义

class Table {

float height;

public:

Table(float h) {height=h;}

float Height() {

return height;

}

};

//圆桌类RoundTable的定义

class RoundTable:public Table,public Circle {

Color color;

public:

RoundTable(float h,float r,Color c); //构造函数

int GetColor() {

return color;

}

};

//圆桌构造函数的定义

RoundTable::RoundTable(float h,float r,Color c):Table(h),Circle(r)

{

color=c;

}

//main()函数的定义

main() {

RoundTable cir\_table(15.0,2.0,Yellow);

cout<<"The table properties are:"<<endl;

//调用Height类的成员函数

cout<<"Height="<<cir\_table.Height()<<endl;

//调用circle类的成员函数

cout<<"Area="<<cir\_table.Area()<<endl;

//调用RoundTable类的成员函数

cout<<"Color="<<cir\_table.GetColor()<<endl;

}

#include <iostream.h>

//定义一个枚举类型

enum Color {Red,Yellow,Green,White};

//圆类Circle的定义

class Circle {

float radius;

public:

Circle(float r) {

radius=r;

cout<<"Circle initialized!"<<endl;

}

~Circle() { //析构函数

cout<<"Circle destroyed!"<<endl;

}

float Area() {

return 3.1416\*radius\*radius;

}

};

//桌子类Table的定义

class Table {

float height;

public:

Table(float h) {

height=h;

cout<<"Table initialized!"<<endl;

}

~Table() { //构造函数

cout<<"Table destroyed!"<<endl;

}

float Height() {

return height;

}

};

//圆桌类RoundTable的定义

class RoundTable:public Table,public Circle {

Color color;

public:

RoundTable(float h,float r,Color c); //构造函数

int GetColor() {

return color;

}

~RoundTable() { //构造函数

cout<<"RoundTable destroyed!"<<endl;

}

};

//圆桌构造函数的定义

RoundTable::RoundTable(float h,float r,Color c):Table(h),Circle(r)

{

color=c;

cout<<"RoundTable initialized!"<<endl;

}

//测试多继承中构造函数和析构函数的执行方式

main() {

RoundTable cir\_table(15.0,2.0,Yellow);

cout<<"The table properties are:"<<endl;

//调用Height类的成员函数

cout<<"Height="<<cir\_table.Height()<<endl;

//调用circle类的成员函数

cout<<"Area="<<cir\_table.Area()<<endl;

//调用RoundTable类的成员函数

cout<<"Color="<<cir\_table.GetColor()<<endl;

}

#include<iostream.h>

//定义有两个虚函数的基类

class Base {

public:

//定义两个虚函数

virtual void aFn1(void){

cout<<"aFnl is in Base class."<<endl;

}

virtual void aFn2(void) {

cout<<"aFn2 is in Base class."<<endl;

}

//定义非虚函数

void aFn3(void) {

cout<<"aFn3 is in Base class."<<endl;

}

};

//派生类Derived\_1中重新定义了基类中的虚函数aFn1

class Derived\_1:public Base

{

public:

void aFn1(void) { //覆盖aFn1()函数

cout<<"aFnl is in First derived class."<<endl;

}

// void aFn3(void) { 语法错误

// cout<<"aFn3 is in First derived class."<<endl;

//}

};

//派生类Derived\_2中重新定义了基类中的虚函数aFn2

class Derived\_2:public Base{

public:

void aFn2(void){ //覆盖aFn2()函数

cout<<"aFn2 is in Second derived class."<<endl;

}

// void aFn3(void) { 语法错误

// cout<<"aFn3 is in Second derived class."<<endl;

//}

};

//main()函数的定义

main(void)

{

//创建和使用基类Base的对象

Base b;

cout<<"Base:"<<endl;

b.aFn1();

b.aFn2();

b.aFn3();

cout<<"----------------------"<<endl;

//创建和使用派生类Derived\_1的对象

Derived\_1 d1;

cout<<"Derived\_1:"<<endl;

d1.aFn1();

d1.aFn2();

d1.aFn3();

cout<<"----------------------"<<endl;

//创建和使用派生类Derived\_2的对象

Derived\_2 d2;

cout<<"Derived\_2:"<<endl;

d2.aFn1();

d2.aFn2();

d2.aFn3();

}

#include<iostream.h>

//定义抽象类

class Base {

public:

//定义两个纯虚函数

virtual void aFn1(void)=0;

virtual void aFn2(void)=0;

};

//派生类Derived\_1中覆盖了基类中的纯虚函数

class Derived\_1:public Base

{

public:

void aFn1(void) {

cout<<"aFnl is in First derived class."<<endl;

}

void aFn2(void) {

cout<<"aFn2 is in First derived class."<<endl;

}

};

//派生类Derived\_2中覆盖了基类中的纯虚函数

class Derived\_2:public Base{

public:

virtual void aFn1(void){

cout<<"aFn1 is in Second derived class."<<endl;

}

void aFn2(void){

cout<<"aFn2 is in Second derived class."<<endl;

}

};

//main()函数中测试抽象类及其派生类的对象

main(void)

{

//用抽象类不能创建对象

// Base b; 语法错误

// b.aFn1();

// b.aFn2();

//创建和使用Derived\_1类的对象

Derived\_1 d1;

cout<<"Derived\_1 d1:"<<endl;

d1.aFn1();

d1.aFn2();

cout<<"------------------"<<endl;

//创建和使用Derived\_2类的对象

Derived\_2 d2;

cout<<"Derived\_2 d2:"<<endl;

d2.aFn1();

d2.aFn2();

}

#include<iostream.h>

int extract\_int()

{

char ch;

int n=0;

while(ch=cin.get())

if (ch>='0' && ch<='9')

{

cin.putback(ch);

cin>>n;

break;

}

return n;

}

//main()函数

main(void)

{

//提取字符串中的数字

int a=extract\_int();

int b=extract\_int();

int c=extract\_int();

//显示结果

cout<<a<<"+"<<b<<"="<<c<<endl;

}

#include<iostream.h>

//定义节点（数据对象）的接口

class Node

{

//声明list类为本类的友元类

friend class list;

//私有成员

private:

int Data; //节点数据

Node \*previous; //前趋指针

Node \*next; //后继指针

};

//定义双向链表list的接口声明

class list

{

//私有成员

private:

Node \*Head; //链表头指针

Node \*Tail; //链表尾指针

//定义接口函数

public:

//构造函数

list();

//析构函数

~list();

//从链表尾后添加数据

void Build\_HT(int Data);

//从链表前头添加数据

void Build\_TH(int Data);

//从头到尾显示数据

void list::Display\_HT();

//从尾到头显示数据

void list::Display\_TH();

//清除链表的全部数据

void Clear();

};

//main()函数测试双向链表

int main(void)

{

list list1;

int i;

//从尾添加数据

cout<<"Add to the back of the list1:"<<endl;

for (i=1;i<=20;i=i+2) {

list1.Build\_HT(i);

cout<<i<<" ";

}

cout<<endl;

//从头添加数据

cout<<"Add to the front of the list1:"<<endl;

for (i=0;i<=20;i=i+2) {

list1.Build\_TH(i);

cout<<i<<" ";

}

cout<<endl;

//显示链表

list1.Display\_HT();

list1.Display\_TH();

return 0;

}

//list类函数的定义

//构造函数的定义

list::list()

{

//初值

Head=0;

Tail=0;

}

//析构函数的定义

list::~list()

{

Clear();

}

//从链表尾后添加数据

void list::Build\_HT(int Data)

{

Node \*Buffer;

Buffer=new Node;

Buffer->Data=Data;

if(Head==0)

{

Head=Buffer;

Head->next=0;

Head->previous=0;

Tail=Head;

}

else

{

Tail->next=Buffer;

Buffer->previous=Tail;

Buffer->next=0;

Tail=Buffer;

}

}

//从链表前头添加数据

void list::Build\_TH(int Data)

{

Node \*NewNode;

NewNode=new Node;

NewNode->Data=Data;

if(Tail==0)

{

Tail=NewNode;

Tail->next=0;

Tail->previous=0;

Head=Tail;

}

else

{

NewNode->previous=0;

NewNode->next=Head;

Head->previous=NewNode;

Head=NewNode;

}

}

//从头到尾显示数据

void list::Display\_HT()

{

Node \*TEMP;

TEMP=Head;

cout<<"Display the list from Head to Tail:"<<endl;

while(TEMP!=0)

{

cout<<TEMP->Data<<" ";

TEMP=TEMP->next;

}

cout<<endl;

}

//从尾到头显示数据

void list::Display\_TH()

{

Node \*TEMP;

TEMP=Tail;

cout<<"Display the list from Tail to Head:"<<endl;

while(TEMP!=0)

{

cout<<TEMP->Data<<" ";

TEMP=TEMP->previous;

}

cout<<endl;

}

//清除链表的全部数据

void list::Clear()

{

Node \*Temp\_head=Head;

if (Temp\_head==0) return;

do

{

Node \*TEMP\_NODE=Temp\_head;

Temp\_head=Temp\_head->next;

delete TEMP\_NODE;

}

while (Temp\_head!=0);

}

#include <iostream>

#include <string>

using namespace std;

//测试字符串(string)对象

void main()

{

//创建string对象,并显示

string s1;

string s2="ABCDEFGHIJK";

string s3=s2;

string s4(20,'A');

string s5(s2,3,3);

cout<<"s1="<<s1<<endl;

cout<<"s2="<<s2<<endl;

cout<<"s3="<<s3<<endl;

cout<<"s4="<<s4<<endl;

cout<<"s5="<<s5<<endl;

//为string对象输入数据,并显示

cout<<"s1=";

cin>>s1;

cout<<"s2=";

cin>>s2;

cout<<"s3=";

cin>>s3;

cout<<"s4=";

cin>>s4;

cout<<"s5=";

cin>>s5;

cout<<"s1="<<s1<<endl;

cout<<"s2="<<s2<<endl;

cout<<"s3="<<s3<<endl;

cout<<"s4="<<s4<<endl;

cout<<"s5="<<s5<<endl;

}

#include <iostream>

#include <string>

using namespace std;

//测试字符串(string)对象

void main()

{

//创建string对象

string s1,s2;

//string对象的赋值运算

s1="One";

s2="Two";

cout<<"s1="<<s1<<endl;

cout<<"s2="<<s2<<endl;

//string对象的连接运算

string s3;

s3=s1+" and "+s2;

cout<<"s3="<<s3<<endl;

//组合赋值连接运算

s3+=" and Three";

cout<<"s3="<<s3<<endl;

//比较运算及其结果显示

for (int i=1;i<=3;i++) {

cout<<"---------------------"<<endl;

cout<<"s1=";

cin>>s1;

cout<<"s2=";

cin>>s2;

if (s1<s2) //小于

cout<<s1<<" < "<<s2<<endl;

if (s1<=s2) //小于等于

cout<<s1<<" <= "<<s2<<endl;

if (s1==s2) //等于

cout<<s1<<" == "<<s2<<endl;

if (s1>s2) //大于

cout<<s1<<" > "<<s2<<endl;

if (s1>=s2) //大于等于

cout<<s1<<" >= "<<s2<<endl;

if (s1!=s2) //不等

cout<<s1<<" != "<<s2<<endl;

}

}

#include <iostream>

#include <string>

using namespace std;

//测试字符串(string)对象

void main()

{

//创建string对象,并显示

string s1="This";

string s2="book.";

cout<<"s1: "<<s1<<endl;

cout<<"s2: "<<s2<<endl;

//使用length成员函数

cout<<"s1.length()="<<s1.length()<<endl;

cout<<"s2.length()="<<s2.length()<<endl;

//使用append成员函数

s1.append(s2);

cout<<"s1: "<<s1<<endl;

//使用find成员函数和下标运算

int pos=s1.find('b');

cout<<"s1["<<pos<<"]="<<s1[pos]<<endl;

//使用insert成员函数

s1.insert(pos," is a ");

cout<<s1<<endl;

//使用assign成员函数

s1.assign("Good");

cout<<s1<<endl;

}

//根据半径计算圆的周长和面积

#include <iostream.h>

const float PI=3.1416; //声明常量(只读变量)PI为3.1416

float fCir\_L(float); //声明自定义函数fCir\_L()的原型

float fCir\_S(float); //声明自定义函数fCir\_S()的原型

//以下是main()函数

main()

{

float r,l,s; //声明3个变量

cout<<"R="; //显示字符串

cin>>r; //键盘输入

l=fCir\_L(r); //计算圆的周长，赋值给变量l

s=fCir\_S(r); //计算圆的面积，赋值给变量s

cout<<"l="<<l; //显示计算结果

cout<<"\ns="<<s;

}

//定义计算圆的周长的函数fCir\_L()

float fCir\_L(float x)

{

float z=-1.0; //声明局部变量

if (x>=0.0) //如果参数大于0，则计算圆的周长

z=2\*PI\*x;

return(z); //返回函数值

}

//定义计算圆的面积的函数fCir\_S()

float fCir\_S(float x)

{

float z=-1.0; //声明局部变量

if (x>=0.0) //如果参数大于0，则计算圆的面积

z=PI\*x\*x;

return(z); //返回函数值

}

#include<iostream.h>

#include<stdlib.h>

#define MAX 30

//main()的定义

int main(void)

{

char str[MAX],\*p;

//从键盘上输入int数

cout<<"Please input a int:"<<endl;

int n;

cin>>n;

//将整型数n按十进制转换为字符串并输出

p=itoa(n,str,10);

cout<<"str="<<str<<endl;

cout<<"p="<<p<<endl;

//将整型数n按十六进制转换为字符串并输出

p=itoa(n,str,16);

cout<<"str="<<str<<endl;

cout<<"p="<<p<<endl;

//从键盘上输入double类型的数据

cout<<"Please input a double:"<<endl;

double x;

cout<<"x=";

cin>>x;

//将浮点数x转换为字符串后输出

p=gcvt(x,10,str);

cout<<"str="<<str<<endl;

cout<<"p="<<p<<endl;

return 0;

}

#include<iostream.h>

#include<stdlib.h>

#define MAX 30

//main()的定义

int main(void)

{

char str[MAX];

//字符串转换为int和long类型数据

cout<<"Please input a string:"<<endl;

cin>>str;

int n=atoi(str);

cout<<"n="<<n<<endl;

long l=atol(str);

cout<<"l="<<l<<endl;

//字符串转换为double类型

cout<<"Please input a string:"<<endl;

cin>>str;

double x=atof(str);

cout<<"x="<<x<<endl;

return 0;

}

#include<iostream.h>

#include <stdlib.h>

#include <time.h>

//定义产生[n1,n2]范围int随机数的函数

int rand(int n1,int n2) {

if (n1>n2) return -1;

if (n1==n2) return 0;

int temp=n1+int((n2-n1)\*double(rand())/RAND\_MAX);

return temp;

}

//main()函数的定义，加法练习程序

void main( void )

{

int i;

//使用当前的系统时间初始化随机数种子

srand( (unsigned)time( NULL ) );

//加法练习

int a,b,c;

do {

a=rand(0,20);

b=rand(0,20);

L1: cout<<a<<"+"<<b<<"=";

cin>>c;

if (c==0) break;

if (c!=a+b) {

cout<<"Error! Try again!"<<endl;

goto L1;

}

cout<<"OK!"<<endl;

} while (1);

}

#include<iostream.h>

#include <stdlib.h>

#include <math.h>

#define PI 3.1415926535

//main()函数的定义

void main( void )

{

int i;

double x=PI/180;

cout<<"X\tSIN(X)\t\tCOS(X)"<<endl;

cout<<"---------------------------------------"<<endl;

for (i=0;i<=360;i=i+30) {

cout<<i<<"\t";

cout.precision(2);

cout<<sin(i\*x)<<"\t\t";

cout<<cos(i\*x)<<endl;

}

}

#include<iostream.h>

#include <stdlib.h>

#include <math.h>

#define PI 3.1415926535

//main()函数的定义

void main( void )

{

int i;

double d=180/PI;

cout<<"X\tASIN(X)\t\tACOS(X)"<<endl;

cout<<"---------------------------------------"<<endl;

for (double x=0;x<=1.0+0.05;x=x+0.1) {

cout<<x<<"\t";

cout<<int(asin(x)\*d)<<"\t\t";

cout<<int(acos(x)\*d)<<endl;

}

}

#include<iostream.h>

#include <stdlib.h>

#include <math.h>

//main()函数的定义

void main( void )

{

\_complex a={3,4},b={3,-4};

double d=cabs(a);

cout<<"cabs("<<a.x<<","<<a.y<<")="<<d<<endl;

cout<<"cabs("<<b.x<<","<<b.y<<")="<<cabs(b)<<endl;

}

##include<iostream.h>

#include <stdlib.h>

#include <math.h>

//main()函数的定义

void main( void )

{

double x;

//循环输入数据计算对数

do {

cout<<"x=";

cin>>x;

if (x<=0) break;

cout<<"log("<<x<<")="<<log(x)<<endl;

cout<<"log10("<<x<<")="<<log10(x)<<endl;

} while(1);

}

#include<iostream.h>

#include <stdlib.h>

#include <math.h>

//main()函数的定义

void main( void )

{

double y;

for(double x=-5;x<=5;x++){

y=exp(x);

cout<<"exp("<<x<<")="<<y<<endl;

}

}

#include<iostream.h>

#include <stdlib.h>

#include <math.h>

//main()函数的定义

void main( void )

{

double y;

int N;

//输入一个大于等于0的数

do {

cout<<"N=";

cin>>N;

if (N>=0) break;

} while (1);

//计算并显示

for(int i=0;i<=N;i++){

y=pow(2,i);

cout<<"pow("<<2<<","<<i<<")="<<y<<endl;

}

}

#include<iostream.h>

#include <stdlib.h>

#include <math.h>

//main()函数的定义

void main( void )

{

double y;

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

y=sqrt(i);

cout<<"sqrt("<<i<<")="<<y<<endl;

}

}

#include<iostream.h>

#include <time.h>

//时间延迟函数

void Dtime(int dt) {

time\_t current\_time;

time\_t start\_time;

// 得到开始时间

time(&start\_time);

do

{

time(&current\_time);

}

while ((current\_time - start\_time) < dt);

}

//main()函数的定义

void main(void)

{

cout<<"The First information!"<<endl;

cout<<"About to delay 5 seconds"<<endl;

Dtime(5);

cout<<"The Second information!"<<endl;

}

#include<iostream.h>

#include <time.h>

//main()函数的定义

void main(void)

{

//声明time\_t类型的变量，其以秒为单位存放系统时间

time\_t current\_time;

//得到当前的系统时间（秒）

time(&current\_time);

//转换系统时间为tm结构的时间信息

tm \*ptime=gmtime(&current\_time);

//显示time\_t结构的时间

cout<<"current\_time:"<<current\_time<<endl;

//显示tm结构的时间信息

cout<<"seconds after the minute:"<<(ptime->tm\_sec)<<endl;

cout<<"minutes after the hour:"<<(ptime->tm\_min)<<endl;

cout<<"hours since midnight:"<<(ptime->tm\_hour)<<endl;

cout<<"day of the month:"<<(ptime->tm\_mday)<<endl;

cout<<"months since January:"<<(ptime->tm\_mon)<<endl;

cout<<"years since 1900:"<<(ptime->tm\_year)<<endl;

cout<<"days since Sunday:"<<(ptime->tm\_wday)<<endl;

cout<<"days since January 1:"<<(ptime->tm\_yday)<<endl;

cout<<"daylight savings time flag:"<<(ptime->tm\_isdst)<<endl;

}

#include<iostream.h>

#include <time.h>

//main()函数的定义

void main(void)

{

//声明变量

time\_t current\_time;

//得到当前系统时间

time(&current\_time);

//转换系统时间为tm结构

tm \*ptime=gmtime(&current\_time);

//转换time\_t类型的时间字符串并显示

char \*timep=ctime(&current\_time);

cout<<"ctime(&current\_time):"<<endl;

cout<<timep;

//转换tm类型的数据转换为时间字符串并显示

char \*tmp=asctime(ptime);

cout<<"asctime(ptime):"<<endl;

cout<<timep;

}

#include<iostream.h>

#include<conio.h>

#include <time.h>

//定义时间延迟函数

void Dtime(double dt) {

time\_t current\_time;

time\_t start\_time;

//得到开始时间

time(&start\_time);

//延迟处理

do

{

time(&current\_time);

}

while (difftime(current\_time,start\_time)<dt);

}

//main()函数的定义

void main(void)

{

//声明变量

int i;

time\_t current\_time;

char \*timep;

//循环10次，每隔2秒显示一次时间

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

time(&current\_time);

timep=ctime(&current\_time);

cputs(timep);

Dtime(2);

}

}

#include<iostream.h>

#include<stdlib.h>

#include<malloc.h>

int main(void)

{

//定义结构类型

struct student {

int num;

char name[20];

float grade;

};

//声明结构指针变量

struct student \*sp;

//计算申请的内存量

int size=sizeof(struct student);

//申请需要的存储空间并强制类型转换

sp=(struct student\*)malloc(size);

//为结构对象输入数据

cout<<"nmu:";

cin>>(sp->num);

cout<<"name:";

cin>>(sp->name);

cout<<"grade:";

cin>>(sp->grade);

//输出结构对象的数据

cout<<"num:"<<(sp->num)<<endl;

cout<<"name:"<<(sp->name)<<endl;

cout<<"grade:"<<(sp->grade);

//释放内存

free(sp);

}

#include<iostream.h>

#include<conio.h>

#include <time.h>

//定义时间延迟函数

void Dtime(double dt) {

time\_t current\_time;

time\_t start\_time;

// 得到开始时间

time(&start\_time);

//延迟处理

do

{

time(&current\_time);

}

while (difftime(current\_time,start\_time)<dt);

}

//控制台函数显示

void cputs\_show(int n) {

time\_t current\_time;

char \*timep;

cputs("Show time with cputs\n");

for(int i=0;i<5;i++) {

time(&current\_time);

timep=ctime(&current\_time);

cputs(timep);

Dtime(n);

}

}

//cout对象显示

void cout\_show(int n) {

time\_t current\_time;

char \*timep;

cout<<"Show time with cout"<<endl;

for(int i=0;i<5;i++) {

time(&current\_time);

timep=ctime(&current\_time);

cout<<timep;

Dtime(n);

}

}

//main()函数的定义

void main(void)

{

cputs\_show(1);

cout\_show(1);

}

#include<stdio.h>

main()

{

//输出字符串

printf("He said \"Hello!\"");

//输出各进制整数

int i=64;

printf("\ni=%d",i); //以十进制格式输出

printf("\ni=%o",i); //以八进制格式输出

printf("\ni=%x",i); //以十六进制格式输出

printf("\ni=%d,%o,%x",i,i,i); //各种格式混合输出

//输出浮点数

float x=3141.5926;

printf("\nx=%f",x); //指定输出浮点数的格式为十进制形式

printf("\nx=%e",x); //指定输出浮点数的格式为指数形式

//控制输出项宽度

int j=123;

printf("\nj=%-10d",j); //任选项"-"指定左对齐，W 指定宽度为10

printf("\nj=%10d\n",j); //W 指定宽度为10

//控制输出精度

float y=3.1415926;

printf("y=%10.2f\n",y); //W 指定宽度为10，P指定小数点后保留2位

printf("y=%10.5f\n",y); //W 指定宽度为10，P指定小数点后保留5位

}

#include<stdio.h>

main()

{

//输入字符串

char str[80];

printf("str:"); //显示提示

scanf("%s",str);

printf("The string:%s",str);

//输入各进制整数

int a,b,c,sum;

printf("\na\tb\tc\n"); //显示提示

scanf("%d %o %x",&a,&b,&c); //以十进制、八进制、十六进制形式输入数据

sum=a+b+c;

printf("a=%d b=%d c=%d sum=%d",a,b,c,sum);

//输入浮点数并计算显示

float x,y; //声明变量

printf("\nx\ty\n"); //显示提示

scanf("%f %f",&x,&y); //对非空白字符"x= y="读入，不保存

printf("sum=%f product=%f\n",x+y, x\*y); //显示表达式的值

}

#include<iostream.h>

#include<direct.h>

#include<errno.h>

#define MAX\_PATH 250

main()

{

//声明变量

char \*p,str[MAX\_PATH];

//设置新目录

if (mkdir("d:\\ABC")){

cout<<"mkdir Error!"<<endl;

}

//更该工作目录

if (chdir("d:\\ABC")){

cout<<"chdir Error!"<<endl;

}

//读取当前目录

if ((p=getcwd(str,MAX\_PATH))==NULL) {

cout<<"getcwd Error!"<<endl;

}

else

{

cout<<"p:"<<p<<endl;

cout<<"str:"<<str<<endl;

}

//更该工作目录

if (chdir("d:\\")){

cout<<"chdir Error!"<<endl;

}

//删除指定目录

if (rmdir("d:\\ABC")==-1)

cout<<"rmdir Error!"<<endl;

}

#include<iostream.h>

#include <time.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <stdio.h>

void main( void )

{

struct stat buf;

int result;

//获得c:\Windows\Calc.exe文件的状态信息

result =stat( "c:\\windows\\Calc.exe", &buf );

//显示Calc.exe文件的状态信息

if( result != 0 )

perror( "Problem getting information" );

else

{

cout<<"Size of the file in bytes:"<<buf.st\_size<<endl;

cout<<"Drive number of the disk containing the file :";

cout<<char(buf.st\_dev + 'A')<<endl;

cout<<"Time of creation of the file:"<<ctime(&buf.st\_ctime);

cout<<"Time of last access of the file:"<<ctime(&buf.st\_atime);

cout<<"Time of last modification of the file:"<<ctime(&buf.st\_mtime);

}

}

#include<iostream.h>

#include <string.h>

void main( void )

{

//设置字符串

char string[] = "Fill the string with something";

cout<<"string:"<<string<<endl;

char \*p=strset(string,'\*');

cout<<"p :"<<p<<endl;

cout<<"string:"<<string<<endl;

//按指定字符和指定数目设置字符数组

char string1[] = "Fill the string with something";

cout<<"string1:"<<string1<<endl;

p=strnset(string1,'\*',5);

cout<<"p :"<<p<<endl;

cout<<"string1:"<<string1<<endl;

}

#include<iostream.h>

#include <string.h>

void main( void )

{

//拷贝字符串常量到字符数组

char string[80] = "Fill the string with something";

cout<<"string:"<<string<<endl;

cout<<"strcpy:"<<endl;

char \*p=strcpy(string,"abc");

cout<<"p :"<<p<<endl;

cout<<"string:"<<string<<endl;

char str[80];

cout<<"str:";

cin>>str;

p=strcpy(string,str);

cout<<"p :"<<p<<endl;

cout<<"string:"<<string<<endl;

//拷贝前5个字符到string中

cout<<"str:";

cin>>str;

cout<<"strncpy:"<<endl;

p=strncpy(string,str,strlen(str));

cout<<"p :"<<p<<endl;

cout<<"string:"<<string<<endl;

}

#include<iostream.h>

#include <string.h>

void main( void )

{

//声明字符数组和字符型指针变量

char string[80],\*p;

//拷贝字符串

strcpy( string, "I'll see you");

cout<<"string:"<<string<<endl;

//追加字符串

p=strcat( string, " in the morning.");

cout<<"String: "<<string<<endl;

cout<<"p : "<<p<<endl;

}

#include<iostream.h>

#include <string.h>

//字符串输入函数

void str\_input(char \*p1,char \*p2)

{

cout<<"string1:";

cin>>p1;

cout<<"string2:";

cin>>p2;

}

//显示strcmp()函数的比较结果

void strcmp\_put(char \*p1,char \*p2)

{

cout<<"strcmp():"<<endl;

int result=strcmp(p1,p2);

if (result>0)

cout<<p1<<" greater than "<<p2<<endl;

if (result<0)

cout<<p1<<" less than "<<p2<<endl;

if (result==0)

cout<<p1<<" identical to "<<p2<<endl;

}

//显示stricmp()函数的比较结果

void stricmp\_put(char \*p1,char \*p2)

{

cout<<"stricmp():"<<endl;

int result=stricmp(p1,p2);

if (result>0)

cout<<p1<<" greater than "<<p2<<endl;

if (result<0)

cout<<p1<<" less than "<<p2<<endl;

if (result==0)

cout<<p1<<" identical to "<<p2<<endl;

}

//显示strncmp()函数的比较结果

void strncmp\_put(char \*p1,char \*p2,size\_t count )

{

cout<<"strncmp():"<<endl;

int result=strncmp(p1,p2,count);

if (result>0)

cout<<p1<<" greater than "<<p2<<endl;

if (result<0)

cout<<p1<<" less than "<<p2<<endl;

if (result==0)

cout<<p1<<" identical to "<<p2<<endl;

}

//main()函数

void main( void )

{

//声明字符数组

char str1[80],str2[80],p;

int i;

//测试测试各字符串比较函数

for(i=1;i<=3;i++) {

str\_input(str1,str2);

strcmp\_put(str1,str2);

stricmp\_put(str1,str2);

strncmp\_put(str1,str2,3);

cout<<"----------------------"<<endl;

}

}

#include<iostream.h>

#include <string.h>

//main()函数

void main( void )

{

//声明字符数组

char string[80],\*p;

int i;

//转换字符串中的小写字母为大写

cout<<"Convert a string to uppercase:"<<endl;

cout<<"string:";

cin>>string;

p=strupr(string);

cout<<"p:"<<p<<endl;

cout<<"string:"<<string<<endl;

cout<<"----------------------"<<endl;

//转换字符串中的大写字母为小写

cout<<"Convert a string to lowercase:"<<endl;

cout<<"string:";

cin>>string;

p=strlwr(string);

cout<<"p:"<<p<<endl;

cout<<"string:"<<string<<endl;

}

#include<iostream.h>

#include <string.h>

//main()函数

void main( void )

{

//声明字符数组

char string[]="This is a test.";

int n;

//获得字符串的长度

cout<<"string:"<<string<<endl;

n=strlen(string);

cout<<"The length of "<<"\""<<string<<"\": "<<n<<endl;

//输入字符并计算其长度

cout<<"string:";

cin>>string;

n=strlen(string);

cout<<"The length of "<<"\""<<string<<"\": "<<n<<endl;

}

#include<iostream.h>

#include <string.h>

//main()函数

void main( void )

{

//声明字符数组

char ch,string[80],\*p;

int n;

//输入字符串和要查找的字符

cout<<"Test strchr():"<<endl;

cout<<"string:";

cin>>string;

cout<<"ch :";

cin>>ch;

//在string中查找ch中的字符并显示

p=strchr(string,ch);

cout<<"p :"<<p<<endl;

//输入字符串和要查找的字符串并查找

char substr[80];

cout<<"Test strstr():"<<endl;

cout<<"substr:";

cin>>substr;

//在string中查找substr中的字符串并显示

p=strstr(string,substr);

cout<<"p :"<<p<<endl;

}

#include<iostream.h>

#include <string.h>

//main()函数

void main( void )

{

//声明字符数组

char string[80],\*p;

//输入字符串并将其反转

cout<<"string:";

cin>>string;

p=strrev(string );

cout<<"p :"<<p<<endl;

cout<<"string:"<<string<<endl;

}

#include<iostream.h>

#include <string.h>

char string[80];

char seps[] = " ,\t\n";

char \*token;

void main( void )

{

//从键盘上输入两个语句

for (int i=1;i<3;i++) {

cout<<"Please input a sentence:"<<endl;

//整行输入

cin.getline(string,80);

cout<<"Tokens:"<<endl;

//首次分离字符串

token = strtok( string, seps );

while( token != NULL ) //结束分离判断

{

cout<<token<<endl;

//下次分离字符串

token = strtok( NULL, seps );

}

}

}

#include<iostream.h>

#include<stdio.h>

#include <string.h>

//main()函数

void main( void )

{

//声明变量和数组

char buffer[200], s[] = "computer", c = 'l';

int i = 35, j;

float fp = 1.7320534f;

//格式化输出到buffer

j = sprintf( buffer, "\tString: %s\n", s );

j += sprintf( buffer + j, "\tCharacter: %c\n", c );

j += sprintf( buffer + j, "\tInteger: %d\n", i );

j += sprintf( buffer + j, "\tReal: %f\n", fp );

cout<<"Output:"<<endl;

cout<<buffer;

cout<<"character count ="<<j<<endl;

}

//根据半径计算圆的周长和面积

#include <iostream.h>

const float PI=3.1416; //声明常量(只读变量)PI为3.1416

float fCir\_L(float); //声明自定义函数fCir\_L()的原型

float fCir\_S(float); //声明自定义函数fCir\_S()的原型

//以下是main()函数

main()

{

float r,l,s; //声明3个变量

cout<<"R="; //显示字符串

cin>>r; //键盘输入

l=fCir\_L(r); //计算圆的周长，赋值给变量l

s=fCir\_S(r); //计算圆的面积，赋值给变量s

cout<<"l="<<l; //显示计算结果

cout<<"\ns="<<s;

}

//定义计算圆的周长的函数fCir\_L()

float fCir\_L(float x)

{

float z=-1.0; //声明局部变量

if (x>=0.0) //如果参数大于0，则计算圆的周长

z=2\*PI\*x;

return(z); //返回函数值

}

//定义计算圆的面积的函数fCir\_S()

float fCir\_S(float x)

{

float z=-1.0; //声明局部变量

if (x>=0.0) //如果参数大于0，则计算圆的面积

z=PI\*x\*x;

return(z); //返回函数值

}

#include<iostream.h>

//定义名为max\_value的函数模板

template <class T> T max\_value (T a,T b)

{

return ((a> b)? a: b);

}

//在main()函数中测试max\_value函数模板

void main(void)

{

//double类型数据使用max\_value模板函数

double x = 1.2, y = 2.1;

cout<<"x="<<x<<"\t";

cout<<"y="<<y<<endl;

double result=max\_value(x,y);

cout<<"max\_value(x,y)="<<result<<endl;

cout<<"max\_value(2\*3.0,2+3.0)="<<max\_value(2\*3.0,2+3.0)<<endl;

cout<<"------------------"<<endl;

//int类型数据使用max\_value模板函数

int n= 1, m= 6;

cout<<"n="<<n<<"\t";

cout<<"m="<<m<<endl;

cout<<"max\_value(n,m)="<<max\_value(n,m)<<endl;

cout<<"------------------"<<endl;

//char类型数据使用max\_value模板函数

char ch1='A',ch2='a';

cout<<"ch1="<<ch1<<"\t";

cout<<"ch2="<<ch2<<endl;

cout<<"max\_value(ch1,ch2)="<<max\_value(ch1,ch2)<<endl;

cout<<"------------------"<<endl;

//字符串数据使用max\_value模板函数

char str1[]="abc",str2[]="ABC",\*p;

p=max\_value(str1,str2);

cout<<"max\_value("<<str1<<","<<str2<<")="<<p<<endl;

}

#include<iostream.h>

//函数模板的原型

template <class T1, class T2> void display(T1 x, T2 y);

//在main()函数中测试display函数模板

void main(void)

{

//声明变量

char c='A';

char str[]="This is a test";

int n=10;

float x=1.5;

double z=3.1415926;

//两个参数类型相同

display(c, char(c+2));

display(str, str);

display(n, 2\*n);

display(x,2\*x);

display(z, 2\*z);

cout<<"------------------"<<endl;

//两个参数类型不同

display(c, str);

display(str, c);

display(n, str);

display(str,2\*x);

display(z, n);

}

//定义名为display的函数模板

template <class T1, class T2> void display(T1 x, T2 y)

{

cout << x << " " << y << endl;

}

#include<iostream.h>

//声明引用参数的函数模板原型

template <class T> void swap(T &x, T &y);

//定义一个结构类型

struct student {

int n;

char name[20];

float grade;

};

//在main()函数中测试swap()函数模板

void main(void)

{

//交换两个int型变量中的数据

int m=3,n=5;

cout<<"m="<<m<<" n="<<n<<endl;

swap(m,n);

cout<<"m="<<m<<" n="<<n<<endl;

cout<<"-------------------"<<endl;

//交换两个double型变量中的数据

double x=3.5,y=5.7;

cout<<"x="<<x<<" y="<<y<<endl;

swap(x,y);

cout<<"x="<<x<<" y="<<y<<endl;

cout<<"-------------------"<<endl;

//交换两个char型变量中的数据

char c1='A',c2='a';

cout<<"c1="<<c1<<" c2="<<c2<<endl;

swap(c1,c2);

cout<<"c1="<<c1<<" c2="<<c2<<endl;

cout<<"-------------------"<<endl;

//交换两个结构变量中的数据

student s1={1001,"ZhangHua",90};

student s2={1011,"LiWei",95.5};

cout<<"s1: ";

cout<<s1.n<<" "<<s1.name<<" "<<s1.grade<<endl;

cout<<"s2: ";

cout<<s2.n<<" "<<s2.name<<" "<<s2.grade<<endl;

swap(s1,s2);

cout<<"swap(s1,s2):"<<endl;

cout<<"s1: ";

cout<<s1.n<<" "<<s1.name<<" "<<s1.grade<<endl;

cout<<"s2: ";

cout<<s2.n<<" "<<s2.name<<" "<<s2.grade<<endl;

}

//定义名为swap的函数模板用于交换两个变量中的数据

template <class T> void swap(T &x, T &y)

{

T temp;

temp=x;

x=y;

y=temp;

}

#include<iostream.h>

//声明函数模板的原型语句

template <class T> void swap(T \*x, T \*y);

//定义一个结构类型

struct student {

int n;

char name[20];

float grade;

};

//在main()函数中测试swap()函数模板

void main(void)

{

//交换两个int型变量中的数据

int m=3,n=5;

cout<<"m="<<m<<" n="<<n<<endl;

swap(&m,&n);

cout<<"m="<<m<<" n="<<n<<endl;

cout<<"-------------------"<<endl;

//交换两个double型变量中的数据

double x=3.5,y=5.7;

cout<<"x="<<x<<" y="<<y<<endl;

swap(&x,&y);

cout<<"x="<<x<<" y="<<y<<endl;

cout<<"-------------------"<<endl;

//交换两个char型变量中的数据

char c1='A',c2='a';

cout<<"c1="<<c1<<" c2="<<c2<<endl;

swap(&c1,&c2);

cout<<"c1="<<c1<<" c2="<<c2<<endl;

cout<<"-------------------"<<endl;

//交换两个结构变量中的数据

student s1={1001,"ZhangHua",90};

student s2={1011,"LiWei",95.5};

cout<<"s1: ";

cout<<s1.n<<" "<<s1.name<<" "<<s1.grade<<endl;

cout<<"s2: ";

cout<<s2.n<<" "<<s2.name<<" "<<s2.grade<<endl;

swap(&s1,&s2);

cout<<"swap(s1,s2):"<<endl;

cout<<"s1: ";

cout<<s1.n<<" "<<s1.name<<" "<<s1.grade<<endl;

cout<<"s2: ";

cout<<s2.n<<" "<<s2.name<<" "<<s2.grade<<endl;

}

//定义名为swap的函数模板用于交换两个变量中的数据

template <class T> void swap(T \*x, T \*y)

{

T temp;

temp=\*x;

\*x=\*y;

\*y=temp;

}

#include<iostream.h>

//定义输入函数模板

template <class T> void input(char \*str,T &x) {

cout<<str<<"=";

cin>>x;

}

//定义输出函数模板

template <class T> void output(char \*str,T x) {

cout<<str<<"="<<x<<endl;

}

//在main()函数中测试输入输出函数模板

void main(void)

{

//输入输出int型数据

int a,b;

input("a",a);

output("a",a);

b=3\*a;

output("3\*a",b);

output("a+b",a+b);

cout<<"-------------------"<<endl;

//输入输出double型数据

double x,y;

input("x",x);

output("x",x);

y=2\*x;

output("y",y);

cout<<"-------------------"<<endl;

//输入输出char型数据

char c1;

input("c1",c1);

output("c1+2",char(c1+2));

cout<<"-------------------"<<endl;

//输入输出字符串数据

char string[80];

input("string",string);

output("string",string);

}

#include<iostream.h>

#include<string.h>

//显示数组的函数模板

template <class T> void arr\_put(T arr[],int size) {

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

cout<<arr[i]<<" ";

cout<<endl;

}

//选择排序数组的函数模板

template <class T> void sort(T arr[],int size) {

T temp;

int i,j;

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

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

if (arr[i]<=arr[j])

{

temp=arr[i];

arr[i]=arr[j];

arr[j]=temp;

}

}

//在main()函数中测试数组排序的函数模板

void main(void)

{

//用排序函数模板处理int型数组

cout<<"int:"<<endl;

int a[]={1,5,2,7,9,0,10,-1};

arr\_put(a,7);

sort(a,7);

arr\_put(a,7);

//用排序函数模板处理double型数组

cout<<"double:"<<endl;

double x[]={1.2,2.1,1.414,1.732};

arr\_put(x,3);

sort(x,3);

arr\_put(x,3);

//用排序函数模板处理char类型数组

cout<<"char:"<<endl;

char str[80];

cout<<"str:";

cin>>str;

int size=strlen(str);

arr\_put(str,size);

sort(str,size);

arr\_put(str,size);

}

#include<iostream.h>

#include<string.h>

//显示数组的函数模板

template <class T> void arr\_put(T arr[],int size) {

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

cout<<arr[i]<<" ";

cout<<endl;

}

//选择法对数组排序的函数模板

template <class T> void sort(T arr[],int size) {

T temp;

int i,j;

for (i=0;i<size-1;i++)

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

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

{

temp=arr[i];

arr[i]=arr[j];

arr[j]=temp;

}

}

//二分查找法的函数模板

template <class T> int binary\_search(T array[], T value, int size)

{

int found = 0;

int high = size, low = 0, mid;

mid = (high + low) / 2;

cout<<"Looking for "<<value<<endl;

while ((! found) && (high >= low))

{

if (value == array[mid])

found = 1;

else if (value < array[mid])

high = mid - 1;

else

low = mid + 1;

mid = (high + low) / 2;

}

return((found) ? mid: -1);

}

//main()函数中使用处理数组的函数模板

void main(void)

{

//处理int型数组

int array[10]={1,3,5,7,9,2,4,6,8,10};

//显示数组初值

arr\_put(array,10);

//对数组排序并显示

sort(array,10);

arr\_put(array,10);

//查找数组

cout<<"Result of search: "<<binary\_search(array, 3, 10)<<endl;

cout<<"Result of search: "<<binary\_search(array, 2, 10)<<endl;

cout<<"Result of search: "<<binary\_search(array, 9, 10)<<endl;

cout<<"Result of search: "<<binary\_search(array, 5, 10)<<endl;

cout<<"------------------------------"<<endl;

//处理字符串型数组

char ch1,str[]="happy";

int size=strlen(str);

//显示数组初值

arr\_put(str,size);

//对数组排序并显示

sort(str,size);

arr\_put(str,size);

//查找数组

cout<<"Input a char:";

cin>>ch1;

cout<<"Result of search: "<<binary\_search(str, ch1, size)<<endl;

}

#include<iostream.h>

//定义名为ex\_class的类模板

template <class T> class ex\_class

{

T value;

public:

ex\_class(T v) { value=v; }

void set\_value(T v) { value=v; }

T get\_value(void) {return value;}

};

//main()函数中测试ex\_class类模板

void main(void)

{

//测试int类型数据

ex\_class <int> a(5),b(10);

cout<<"a.value:"<<a.get\_value()<<endl;

cout<<"b.value:"<<b.get\_value()<<endl;

//测试char类型数据

ex\_class <char> ch('A');

cout<<"ch.value:"<<ch.get\_value()<<endl;

ch.set\_value('a');

cout<<"ch.value:"<<ch.get\_value()<<endl;

//测试double类型数据

ex\_class <double> x(5.5);

cout<<"x.value:"<<x.get\_value()<<endl;

x.set\_value(7.5);

cout<<"x.value:"<<x.get\_value()<<endl;

}

#include <iostream.h>

//定义栈的尺寸

const int SIZE = 100;

//定义处理栈的类模板接口

template <class T> class stack {

T stck[SIZE];

int tos;

public:

stack(void) {

tos = 0;

cout << "Stack Initialized." << endl;

}

~stack(void) {

cout << "Stack Destroyed." << endl;

}

void push(T);

T pop(void);

};

//定义栈的成员函数

template <class T> void stack<T>::push(T i)

{

if(tos==SIZE)

{

cout << "Stack is full." << endl;

return;

}

stck[tos++] = i;

}

template <class T> T stack<T>::pop(void)

{

if(tos==0)

{

cout << "Stack underflow." << endl;

return 0;

}

return stck[--tos];

}

//main()函数中测试stack类模板

void main(void)

{

//处理int类型数据的栈

cout<<"stack<int> a :"<<endl;

stack<int> a;

a.push(1);

a.push(2);

cout << a.pop() << " ";

cout << a.pop() << endl;

//处理double类型数据的栈

cout<<"stack<double> b :"<<endl;

stack<double> b;

b.push(99.3);

b.push(-12.23);

cout << b.pop() << " ";

cout << b.pop() <<endl;

//处理char类型数据的栈

cout<<"stack<char> c :"<<endl;

stack<char> c;

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

c.push((char) 'A' + i);

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

cout <<c.pop();

cout << endl;

}

#include<iostream.h>

//定义名为ex\_class的类模板

template <class T1,class T2> class ex\_class

{

T1 value1;

T2 value2;

public:

ex\_class(T1 v1,T2 v2) {

value1=v1;

value2=v2;

}

void set\_value(T1 v1,T2 v2) {

value1=v1;

value2=v2;

}

void put\_value(void) {

cout<<"valu1="<<value1<<endl;

cout<<"valu2="<<value2<<endl;

}

};

//main()函数中测试ex\_class类模板

void main(void)

{

//测试int和double类型数据

ex\_class <int,double> a(5,1.5);

cout<<"ex\_class <int,double> a:"<<endl;

a.put\_value();

a.set\_value(100,3.14);

a.put\_value();

//测试double和int类型数据

ex\_class <double,int> b(0.5,5);

cout<<"ex\_class <double,int> b:"<<endl;

b.put\_value();

b.set\_value(1.732,100);

b.put\_value();

//测试char和int类型数据

ex\_class <char,int> c('a',5);

cout<<"ex\_class <char,int> c:"<<endl;

c.put\_value();

c.set\_value('B',100);

c.put\_value();

//测试int和int类型数据

ex\_class <int,int> d(5,10);

cout<<"ex\_class <int,int> d:"<<endl;

d.put\_value();

d.set\_value(100,200);

d.put\_value();

}

#include <iostream>

#include <list>

#include <numeric>

#include <algorithm>

using namespace std;

//创建一个list容器的实例LISTINT

typedef list<int> LISTINT;

//创建一个list容器的实例LISTCHAR

typedef list<int> LISTCHAR;

void main(void)

{

//--------------------------

//用list容器处理整型数据

//--------------------------

//用LISTINT创建一个名为listOne的list对象

LISTINT listOne;

//声明i为迭代器

LISTINT::iterator i;

//从前面向listOne容器中添加数据

listOne.push\_front (2);

listOne.push\_front (1);

//从后面向listOne容器中添加数据

listOne.push\_back (3);

listOne.push\_back (4);

//从前向后显示listOne中的数据

cout<<"listOne.begin()--- listOne.end():"<<endl;

for (i = listOne.begin(); i != listOne.end(); ++i)

cout << \*i << " ";

cout << endl;

//从后向后显示listOne中的数据

LISTINT::reverse\_iterator ir;

cout<<"listOne.rbegin()---listOne.rend():"<<endl;

for (ir =listOne.rbegin(); ir!=listOne.rend();ir++) {

cout << \*ir << " ";

}

cout << endl;

//使用STL的accumulate(累加)算法

int result = accumulate(listOne.begin(), listOne.end(),0);

cout<<"Sum="<<result<<endl;

cout<<"------------------"<<endl;

//--------------------------

//用list容器处理字符型数据

//--------------------------

//用LISTCHAR创建一个名为listOne的list对象

LISTCHAR listTwo;

//声明i为迭代器

LISTCHAR::iterator j;

//从前面向listTwo容器中添加数据

listTwo.push\_front ('A');

listTwo.push\_front ('B');

//从后面向listTwo容器中添加数据

listTwo.push\_back ('x');

listTwo.push\_back ('y');

//从前向后显示listTwo中的数据

cout<<"listTwo.begin()---listTwo.end():"<<endl;

for (j = listTwo.begin(); j != listTwo.end(); ++j)

cout << char(\*j) << " ";

cout << endl;

//使用STL的max\_element算法求listTwo中的最大元素并显示

j=max\_element(listTwo.begin(),listTwo.end());

cout << "The maximum element in listTwo is: "<<char(\*j)<<endl;

}

#include <iostream>

#include <vector>

using namespace std;

typedef vector<int> INTVECTOR;

//测试vector容器的功能

void main(void)

{

//vec1对象初始为空

INTVECTOR vec1;

//vec2对象最初有10个值为6的元素

INTVECTOR vec2(10,6);

//vec3对象最初有3个值为6的元素

INTVECTOR vec3(vec2.begin(),vec2.begin()+3);

//声明一个名为i的双向迭代器

INTVECTOR::iterator i;

//从前向后显示vec1中的数据

cout<<"vec1.begin()--vec1.end():"<<endl;

for (i =vec1.begin(); i !=vec1.end(); ++i)

cout << \*i << " ";

cout << endl;

//从前向后显示vec2中的数据

cout<<"vec2.begin()--vec2.end():"<<endl;

for (i =vec2.begin(); i !=vec2.end(); ++i)

cout << \*i << " ";

cout << endl;

//从前向后显示vec3中的数据

cout<<"vec3.begin()--vec3.end():"<<endl;

for (i =vec3.begin(); i !=vec3.end(); ++i)

cout << \*i << " ";

cout << endl;

//测试添加和插入成员函数

vec1.push\_back(2);

vec1.push\_back(4);

vec1.insert(vec1.begin()+1,5);

vec1.insert(vec1.begin()+1,vec3.begin(),vec3.end());

cout<<"push() and insert():" <<endl;

for (i =vec1.begin(); i !=vec1.end(); ++i)

cout << \*i << " ";

cout << endl;

//测试赋值成员函数

vec2.assign(8,1);

cout<<"vec2.assign(8,1):" <<endl;

for (i =vec2.begin(); i !=vec2.end(); ++i)

cout << \*i << " ";

cout << endl;

//测试引用类函数

cout<<"vec1.front()="<<vec1.front()<<endl;

cout<<"vec1.back()="<<vec1.back()<<endl;

cout<<"vec1.at(4)="<<vec1.at(4)<<endl;

cout<<"vec1[4]="<<vec1[4]<<endl;

//测试移出和删除

vec1.pop\_back();

vec1.erase(vec1.begin()+1,vec1.end()-2);

cout<<"vec1.pop\_back() and vec1.erase():" <<endl;

for (i =vec1.begin(); i !=vec1.end(); ++i)

cout << \*i << " ";

cout << endl;

//显示序列的状态信息

cout<<"vec1.capacity(): "<<vec1.capacity()<<endl;

cout<<"vec1.max\_size(): "<<vec1.max\_size()<<endl;

cout<<"vec1.size(): "<<vec1.size()<<endl;

cout<<"vec1.empty(): "<<vec1.empty()<<endl;

//vector序列容器的运算

cout<<"vec1==vec3: "<<(vec1==vec3)<<endl;

cout<<"vec1<=vec3: "<<(vec1<=vec3)<<endl;

}

#include <iostream>

#include <deque>

using namespace std;

typedef deque<int> INTDEQUE;

//从前向后显示deque队列的全部元素

void put\_deque(INTDEQUE deque, char \*name)

{

INTDEQUE::iterator pdeque;

cout << "The contents of " << name << " : ";

for(pdeque = deque.begin(); pdeque != deque.end(); pdeque++)

cout << \*pdeque << " ";

cout<<endl;

}

//测试deqtor容器的功能

void main(void)

{

//deq1对象初始为空

INTDEQUE deq1;

//deq2对象最初有10个值为6的元素

INTDEQUE deq2(10,6);

//deq3对象最初有3个值为6的元素

INTDEQUE deq3(deq2.begin(),deq2.begin()+3);

//声明一个名为i的双向迭代器变量

INTDEQUE::iterator i;

//从前向后显示deq1中的数据

put\_deque(deq1,"deq1");

//从前向后显示deq2中的数据

put\_deque(deq2,"deq2");

//从前向后显示deq3中的数据

put\_deque(deq3,"deq3");

//从deq1序列后面添加两个元素

deq1.push\_back(2);

deq1.push\_back(4);

cout<<"deq1.push\_back(2) and deq1.push\_back(4):"<<endl;

put\_deque(deq1,"deq1");

//从deq1序列前面添加两个元素

deq1.push\_front(5);

deq1.push\_front(7);

cout<<"deq1.push\_front(5) and deq1.push\_front(7):"<<endl;

put\_deque(deq1,"deq1");

//在deq1序列中间插入数据

deq1.insert(deq1.begin()+1,3,9);

cout<<"deq1.insert(deq1.begin()+1,3,9):"<<endl;

put\_deque(deq1,"deq1");

//测试引用类函数

cout<<"deq1.front()="<<deq1.front()<<endl;

cout<<"deq1.back()="<<deq1.back()<<endl;

cout<<"deq1.at(4)="<<deq1.at(4)<<endl;

cout<<"deq1[4]="<<deq1[4]<<endl;

deq1.at(1)=10;

deq1[2]=12;

cout<<"deq1.at(1)=10 and deq1[2]=12 :"<<endl;

put\_deque(deq1,"deq1");

//从deq1序列的前后各移去一个元素

deq1.pop\_front();

deq1.pop\_back();

cout<<"deq1.pop\_front() and deq1.pop\_back():"<<endl;

put\_deque(deq1,"deq1");

//清除deq1中的第2个元素

deq1.erase(deq1.begin()+1);

cout<<"deq1.erase(deq1.begin()+1):"<<endl;

put\_deque(deq1,"deq1");

//对deq2赋值并显示

deq2.assign(8,1);

cout<<"deq2.assign(8,1):"<<endl;

put\_deque(deq2,"deq2");

//显示序列的状态信息

cout<<"deq1.max\_size(): "<<deq1.max\_size()<<endl;

cout<<"deq1.size(): "<<deq1.size()<<endl;

cout<<"deq1.empty(): "<<deq1.empty()<<endl;

//deqtor序列容器的运算

cout<<"deq1==deq3: "<<(deq1==deq3)<<endl;

cout<<"deq1<=deq3: "<<(deq1<=deq3)<<endl;

}

#include <iostream>

#include <list>

using namespace std;

typedef list<int> INTLIST;

//从前向后显示list队列的全部元素

void put\_list(INTLIST list, char \*name)

{

INTLIST::iterator plist;

cout << "The contents of " << name << " : ";

for(plist = list.begin(); plist != list.end(); plist++)

cout << \*plist << " ";

cout<<endl;

}

//测试list容器的功能

void main(void)

{

//list1对象初始为空

INTLIST list1;

//list2对象最初有10个值为6的元素

INTLIST list2(10,6);

//list3对象最初有3个值为6的元素

INTLIST list3(list2.begin(),--list2.end());

//声明一个名为i的双向迭代器

INTLIST::iterator i;

//从前向后显示各list对象的元素

put\_list(list1,"list1");

put\_list(list2,"list2");

put\_list(list3,"list3");

//从list1序列后面添加两个元素

list1.push\_back(2);

list1.push\_back(4);

cout<<"list1.push\_back(2) and list1.push\_back(4):"<<endl;

put\_list(list1,"list1");

//从list1序列前面添加两个元素

list1.push\_front(5);

list1.push\_front(7);

cout<<"list1.push\_front(5) and list1.push\_front(7):"<<endl;

put\_list(list1,"list1");

//在list1序列中间插入数据

list1.insert(++list1.begin(),3,9);

cout<<"list1.insert(list1.begin()+1,3,9):"<<endl;

put\_list(list1,"list1");

//测试引用类函数

cout<<"list1.front()="<<list1.front()<<endl;

cout<<"list1.back()="<<list1.back()<<endl;

//从list1序列的前后各移去一个元素

list1.pop\_front();

list1.pop\_back();

cout<<"list1.pop\_front() and list1.pop\_back():"<<endl;

put\_list(list1,"list1");

//清除list1中的第2个元素

list1.erase(++list1.begin());

cout<<"list1.erase(++list1.begin()):"<<endl;

put\_list(list1,"list1");

//对list2赋值并显示

list2.assign(8,1);

cout<<"list2.assign(8,1):"<<endl;

put\_list(list2,"list2");

//显示序列的状态信息

cout<<"list1.max\_size(): "<<list1.max\_size()<<endl;

cout<<"list1.size(): "<<list1.size()<<endl;

cout<<"list1.empty(): "<<list1.empty()<<endl;

//list序列容器的运算

put\_list(list1,"list1");

put\_list(list3,"list3");

cout<<"list1>list3: "<<(list1>list3)<<endl;

cout<<"list1<list3: "<<(list1<list3)<<endl;

//对list1容器排序

list1.sort();

put\_list(list1,"list1");

//结合处理

list1.splice(++list1.begin(), list3);

put\_list(list1,"list1");

put\_list(list3,"list3");

}

#include <iostream.h>

#include <set>

using namespace std;

//创建set模板的实例

typedef set<int> SET\_INT;

//put\_HTset函数，从头向尾显示set容器的所有元素

void put\_HTset(SET\_INT set1,char \*name)

{

SET\_INT::iterator it;

cout<<name<<": ";

cout<<"Head to Tail=";

for (it=set1.begin();it!=set1.end();++it)

cout<<(\*it)<<" ";

cout<<endl;

}

//put\_THset函数，从尾向头显示set容器的所有元素

void put\_THset(SET\_INT s1,char \*name)

{

SET\_INT::reverse\_iterator i;

cout<<name<<": ";

cout<<"Tail to Head=";

for (i=s1.rbegin(); i!=s1.rend();i++)

cout <<(\*i) <<" ";

cout<<endl;

}

//测试set模板

void main(void)

{

int i;

//声明set的对象和迭代器

SET\_INT s1; //容器初始尾空

SET\_INT::iterator it;

//向s1对象中插入值

for (i=1;i<20;i=i+2) {

s1.insert(i);

}

//正向显示s1中的数据

put\_HTset(s1,"s1");

//反向显示s1中的数据

put\_THset(s1,"s1");

//构造含有元素的序列并显示

SET\_INT s2(s1);

put\_HTset(s2,"s2");

//删除s2的第2个元素并显示

s2.erase(++s2.begin());

put\_HTset(s2,"s2");

//向s2插入8和9并显示

s2.insert(8);

s2.insert(9);

put\_HTset(s2,"s2");

//清空s2的序列

s2.clear();

put\_HTset(s2,"s2");

//按关键给定的区间显示序列中的元素

cout<<"[s1.lower\_bound(5),s1.upper\_bound(15)] :";

for (it=s1.lower\_bound(4);it!=s1.upper\_bound(16);it++)

cout<<(\*it)<<" ";

cout<<endl;

//显示s1的状态信息

cout<<"s1.size():"<<s1.size()<<endl;

cout<<"s1.max\_size():"<<s1.max\_size()<<endl;

cout<<"s1.count(15):"<<s1.count(15)<<endl;

//交换两个set容器的元素并显示

s1.swap(s2);

put\_HTset(s1,"s1");

put\_HTset(s2,"s2");

//关系运算

s1.insert(5);

cout<<"s1>s2 = "<<(s1>s2)<<endl;

}

#include <iostream.h>

#include <set>

using namespace std;

//创建multiset模板的实例

typedef multiset<int> MULTISET\_INT;

//put\_HTset函数，从头向尾显示multiset容器的所有元素

void put\_HTset(MULTISET\_INT set1,char \*name)

{

MULTISET\_INT::iterator it;

cout<<name<<": ";

cout<<"Head to Tail=";

for (it=set1.begin();it!=set1.end();++it)

cout<<(\*it)<<" ";

cout<<endl;

}

//put\_THset函数，从尾向头显示multiset容器的所有元素

void put\_THset(MULTISET\_INT s1,char \*name)

{

MULTISET\_INT::reverse\_iterator i;

cout<<name<<": ";

cout<<"Tail to Head=";

for (i=s1.rbegin(); i!=s1.rend();i++)

cout <<(\*i) <<" ";

cout<<endl;

}

//测试multiset模板

void main(void)

{

int i;

//声明multiset的对象和迭代器

MULTISET\_INT s1; //容器初始尾空

MULTISET\_INT::iterator it;

//向s1对象中插入值

for (i=1;i<20;i=i+2) {

s1.insert(i);

}

//正向显示s1中的数据

put\_HTset(s1,"s1");

//反向显示s1中的数据

put\_THset(s1,"s1");

//构造含有元素的序列并显示

MULTISET\_INT s2(s1);

put\_HTset(s2,"s2");

//删除s2的第2个元素并显示

s2.erase(++s2.begin());

put\_HTset(s2,"s2");

//向s2插入8和9并显示

s2.insert(8);

s2.insert(9);

put\_HTset(s2,"s2");

//清空s2的序列

s2.clear();

put\_HTset(s2,"s2");

//按键给定的区间显示序列中的元素

cout<<"[s1.lower\_bound(5),s1.upper\_bound(15)] :";

for (it=s1.lower\_bound(4);it!=s1.upper\_bound(16);it++)

cout<<(\*it)<<" ";

cout<<endl;

//显示s1的状态信息

cout<<"s1.size():"<<s1.size()<<endl;

cout<<"s1.max\_size():"<<s1.max\_size()<<endl;

cout<<"s1.count(15):"<<s1.count(15)<<endl;

//交换两个multiset容器的元素并显示

s1.swap(s2);

put\_HTset(s1,"s1");

put\_HTset(s2,"s2");

//关系运算

s1.insert(2);

put\_HTset(s1,"s1");

put\_HTset(s2,"s2");

cout<<"s1>s2 = "<<(s1>s2)<<endl;

}

#include <iostream>

#include <string>

#include <map>

using namespace std;

//创建map的实例，整数(int)映射字符串(string)

typedef map<int, string> INT2STRING;

//测试map容器

void main()

{

//创建map对象theMap

INT2STRING theMap;

INT2STRING::iterator theIterator,it;

//向theMap容器中添入数据，数字和字符串配对

//每个元素是一个映射对

theMap.insert(INT2STRING::value\_type(0,"Zero"));

theMap.insert(INT2STRING::value\_type(2,"Two"));

theMap.insert(INT2STRING::value\_type(4,"Four"));

theMap.insert(INT2STRING::value\_type(6,"Six"));

theMap.insert(INT2STRING::value\_type(8,"Eight"));

//显示map容器的所有对象

cout<<"theMap.begin()--theMap.end():"<<endl;

for (theIterator=theMap.begin();theIterator!=theMap.end();++theIterator){

cout<<(\*theIterator).first;

cout<<","<<(\*theIterator).second<<" ";

}

cout<<endl;

//测试map容器key的惟一性

theMap.insert(INT2STRING::value\_type(0,"Zero"));

theMap.insert(INT2STRING::value\_type(1,"One"));

theMap.insert(INT2STRING::value\_type(2,"Two"));

theMap.insert(INT2STRING::value\_type(3,"Three"));

theMap.insert(INT2STRING::value\_type(4,"Four"));

theMap.insert(INT2STRING::value\_type(5,"Five"));

theMap.insert(INT2STRING::value\_type(6,"Six"));

theMap.insert(INT2STRING::value\_type(7,"Seven"));

theMap.insert(INT2STRING::value\_type(8,"Eight"));

theMap.insert(INT2STRING::value\_type(9,"Nine"));

//下列语句将不能插入到map容器中

theMap.insert(INT2STRING::value\_type(5,"AAA"));

//显示map容器的所有对象

cout<<"theMap.begin()--theMap.end():"<<endl;

for (theIterator=theMap.begin();theIterator!=theMap.end();++theIterator){

cout<<(\*theIterator).first;

cout<<","<<(\*theIterator).second<<" ";

}

cout<<endl;

//按键给定的区间显示序列中的元素

cout<<"[theMap.lower\_bound(3),theMap.upper\_bound(8)] :"<<endl;

for (it=theMap.lower\_bound(3);it!=theMap.upper\_bound(8);it++) {

cout<<(\*it).first;

cout<<","<<(\*it).second<<" ";

}

cout<<endl;

//显示theMap的状态信息

cout<<"theMap.size():"<<theMap.size()<<endl;

cout<<"theMap.max\_size():"<<theMap.max\_size()<<endl;

cout<<"theMap.count(15):"<<theMap.count(15)<<endl;

// 从键盘上输入数字，显示对应的字符串

string theString = "";

int index;

for( ; ; )

{

cout << "Enter \"q\" to quit, or enter a Number: ";

cin >> theString;

if(theString == "q")

break;

for(index = 0; index < theString.length(); index++){

theIterator = theMap.find(theString[index] - '0');

if(theIterator != theMap.end() )

cout << (\*theIterator).second << " ";

else

cout << "[err] ";

}

cout << endl;

}

}

#include <iostream>

#include <string>

#include <map>

using namespace std;

//创建multimap的实例，整数(int)映射字符串(string)

typedef multimap<int, string> INT2STRING;

//测试multimap容器

void main()

{

//创建multimap对象theMap

INT2STRING theMap;

INT2STRING::iterator theIterator,it;

//向theMap容器中添入数据，数字和字符串配对

//每个元素是一个映射对

theMap.insert(INT2STRING::value\_type(90,"张卫"));

theMap.insert(INT2STRING::value\_type(85,"李华"));

theMap.insert(INT2STRING::value\_type(73,"赵明"));

theMap.insert(INT2STRING::value\_type(96,"郝名"));

//显示multimap容器的所有对象

cout<<"theMap.begin()--theMap.end():"<<endl;

for (theIterator=theMap.begin();theIterator!=theMap.end();++theIterator){

cout<<(\*theIterator).second;

cout<<"\t"<<(\*theIterator).first<<endl;

}

//测试multimap容器key的非惟一性

theMap.insert(INT2STRING::value\_type(90,"李朋"));

theMap.insert(INT2STRING::value\_type(85,"钱德"));

theMap.insert(INT2STRING::value\_type(93,"赵刚"));

//按成绩高低输出multimap容器的所有对象

INT2STRING::reverse\_iterator i;

cout<<"theMap.rbegin()--theMap.rend():"<<endl;

for (i=theMap.rbegin();i!=theMap.rend();++i){

cout<<(\*i).second;

cout<<"\t"<<(\*i).first<<endl;

}

//按关键给定的区间显示序列中的元素

cout<<"[theMap.lower\_bound(80),theMap.upper\_bound(90)] :"<<endl;

for (it=theMap.lower\_bound(80);it!=theMap.upper\_bound(90);it++) {

cout<<(\*it).second;

cout<<"\t"<<(\*it).first<<endl;

}

//显示theMap的状态信息

cout<<"theMap.size():"<<theMap.size()<<endl;

cout<<"theMap.max\_size():"<<theMap.max\_size()<<endl;

cout<<"theMap.count(90):"<<theMap.count(90)<<endl;

//清除90分以下的数据,并显示结果

theMap.erase(theMap.lower\_bound(60),theMap.upper\_bound(89));

cout<<"theMap.rbegin()--theMap.rend():"<<endl;

for (i=theMap.rbegin();i!=theMap.rend();++i){

cout<<(\*i).second;

cout<<"\t"<<(\*i).first<<endl;

}

}

#include <iostream>

#include <valarray>

#include <math.h>

using namespace std;

#define ARRAY\_SIZE 3 //array size

//测试valarray容器

void main()

{

//创建具有3个元素的数组val\_array

valarray<double> val\_array(ARRAY\_SIZE);

//设置数组的值为1, 4, 9

for (int i = 0; i < ARRAY\_SIZE; i++)

val\_array[i] = (i+1) \* (i+1);

//显示val\_array数组的大小

cout << "Size of val\_array = " << val\_array.size() << endl;

// 显示val\_array数组的值

cout << "The values in val\_array before calling sqrt() and pow():" << endl;

for (i = 0; i < ARRAY\_SIZE; i++)

cout << val\_array[i] << " ";

cout << endl;

//声明一个rev\_valarray数组，其保存对数组val\_array的取反

valarray<double> rev\_valarray(ARRAY\_SIZE);

for (i = 0; i < ARRAY\_SIZE; i++)

rev\_valarray[i] = val\_array[ARRAY\_SIZE - i - 1];

//显示rev\_valarray数组的大小和元素

cout << "Size of rev\_valarray = " << rev\_valarray.size() << endl;

cout << "The values in rev\_valarray:" << endl;

for (i = 0; i < ARRAY\_SIZE; i++)

cout << rev\_valarray[i] << " ";

cout <<endl;

// 声明rvalue\_array数组，其存放调用sqrt()和pow()函数的返回值

valarray<double> rvalue\_array;

//调用sqrt()函数并显示结果

rvalue\_array = sqrt(val\_array);

cout << "The result of rvalue\_array after calling sqrt():" << endl;

for (i = 0; i < ARRAY\_SIZE; i++)

cout << rvalue\_array[i] << " ";

cout <<endl;

//对val\_array数组元素计算幂函数并显示

rvalue\_array = pow(val\_array, rev\_valarray);

cout << "The result after calling pow(val\_array, rev\_valarray):"

<< endl;

for (i = 0; i < ARRAY\_SIZE; i++)

cout << rvalue\_array[i] << " ";

cout <<endl;

//对val\_array数组元素计算幂函数，指数均为2.0，并显示

rvalue\_array = pow(val\_array, 2.0);

cout << "The result after calling pow(val\_array, 2.0):" << endl;

for (i = 0; i < ARRAY\_SIZE; i++)

cout << rvalue\_array[i] << " ";

cout <<endl;

//对2.0进行幂函数运算，指数均为数组val\_array的各元素值

rvalue\_array = pow(2.0, val\_array);

cout << "The result after calling pow(2.0, val\_array):" << endl;

for (i = 0; i < ARRAY\_SIZE; i++)

cout << rvalue\_array[i] << " ";

cout <<endl;

//对val\_array和rvalue\_array求和

cout<<"val\_array.sum()="<<val\_array.sum()<<endl;

cout<<"rvalue\_array.sum()="<<rvalue\_array.sum()<<endl;

//求最大值并显示

cout<<"val\_array.max()="<<val\_array.max()<<endl;

cout<<"rvalue\_array.max()="<<rvalue\_array.max()<<endl;

}

#include <stack>

#include <iostream>

using namespace std ;

typedef stack<int> STACK\_INT;

void main()

{

STACK\_INT stack1;

int i;

//判断栈是否空

cout << "stack1.empty() returned " <<

(stack1.empty()? "true": "false") << endl;

//0,2,4,6...入栈

for (i=0;i<10;i=i+2)

stack1.push(i);

//top()函数

if (!stack1.empty())

cout << "stack1.top() returned " <<stack1.top() << endl;

//计算栈的长度

cout<<"stack1.size(): "<<stack1.size()<<endl;

//改变栈顶的值 20.

if (!stack1.empty()) {

cout << "stack1.top()=20;" << endl;

stack1.top()=20;

}

//弹出栈中所有的数据并显示

cout<<"stack1: ";

while (!stack1.empty()) {

cout<<stack1.top()<<" ";

stack1.pop();

}

cout<<endl;

}

#include <iostream>

#include <list>

#include <numeric>

using namespace std;

//创建一个list容器的实例LISTINT，其存放int型数据

typedef list<int> LISTINT;

void main(void)

{

//用LISTINT创建一个名为listOne的list对象

LISTINT listOne;

//指定i为迭代器变量

LISTINT::iterator i;

LISTINT::reverse\_iterator ir;

//从前面向listOne容器中添加数据

listOne.push\_front (2);

listOne.push\_front (1);

//从后面向listOne容器中添加数据

listOne.push\_back (3);

listOne.push\_back (4);

//从前向后显示listOne中的数据

for (i = listOne.begin(); i != listOne.end(); ++i)

cout << \*i << " ";

cout << endl;

//从后向后显示listOne中的数据

for (ir =listOne.rbegin();ir!=listOne.rend(); ++ir)

cout << \*ir << " ";

cout << endl;

//从键盘上输入数据

for (i = listOne.begin(); i != listOne.end(); ++i) {

cout<<"listOne :";

cin>>(\*i);

}

//从前向后显示listOne中的数据

for (i = listOne.begin(); i != listOne.end(); ++i)

cout << \*i << " ";

cout << endl;

//bidirectional迭代器不允许加减运算

// i=listOne.begin()+1;

}

#include <iostream>

#include <iostream>

#include <numeric>

#include <vector>

#include <list>

#include <set>

using namespace std;

//利用类模板生成类实例

typedef vector < int > IntArray;

typedef list <int> LISTINT;

typedef set<int> SET\_INT;

int add(int a, int b) {

return a+b;

}

//在main()函数中测试accumulate算法

void main ()

{

//--------------------------------------------

// accumulate算法对于普通数组的计算

//---------------------------------------------

int x[]={1,3,5,7,9};

cout<<"x[]:";

for (int i=0;i<5;i++)

cout<<x[i]<<" ";

cout<<endl;

cout<<"accumulate(x,x+5,0)=";

cout<<accumulate(x,x+5,0)<<endl;

int val=100;

cout<<"val="<<val<<endl;

cout<<"accumulate(x,x+5,val)=";

cout<<accumulate(x,x+5,val)<<endl;

//--------------------------------------------

// accumulate算法对于vector容器的计算

//---------------------------------------------

//声明intvector容器和迭代器ii

IntArray intvector;

IntArray::iterator ii;

//向intvector容器中插入元素

for (i=1; i<=5; i++) {

intvector.push\_back(i);

};

//显示intvector容器中的元素值和累加结果

cout << "intvector: "<<endl;

for (ii=intvector.begin();ii !=intvector.end();++ii)

cout<<(\*ii)<<" ";

cout<<endl;

cout<<"accumulate(intvector.begin(),intvector.end(),0)=";

cout<<accumulate(intvector.begin(),intvector.end(),0)<<endl;

//--------------------------------------------

// accumulate算法对于list容器的计算

//---------------------------------------------

//声明list容器对象和迭代器

LISTINT::iterator iL;

LISTINT list1;

//向list1容器对象中插入元素并显示

list1.push\_front(1);

list1.push\_front(3);

list1.push\_front(5);

list1.push\_back(2);

list1.push\_back(6);

//显示list1容器的元素值和累加结果

cout << "list1: "<<endl;

for (iL=list1.begin();iL !=list1.end();++iL)

cout<<(\*iL)<<" ";

cout<<endl;

cout<<"accumulate(list1.begin(),list1.end(),0)=";

cout<<accumulate(list1.begin(),list1.end(),0)<<endl;

//--------------------------------------------

// accumulate算法对于set容器的计算

//---------------------------------------------

//声明set容器对象和迭代器

SET\_INT set1;

SET\_INT::iterator si;

//向set1容器中插入元素

set1.insert(5);

set1.insert(20);

set1.insert(10);

set1.insert(15);

set1.insert(25);

//显示set1容器的元素值和累加结果

cout <<"set1: "<<endl;

for (si=set1.begin();si !=set1.end();++si)

cout<<(\*si)<<" ";

cout<<endl;

cout<<"accumulate(set1.begin(),set1.end(),0)=";

cout<<accumulate(set1.begin(),set1.end(),0)<<endl;

cout<<"accumulate(set1.begin(),set1.end(),100)=";

cout<<accumulate(set1.begin(),set1.end(),100)<<endl;

}

#include <iostream>

#include <algorithm>

#include <vector>

#include <list>

#include <set>

#define size 10

using namespace std;

//产生指定范围的整数随机数

int getrand(int min,int max) {

int m;

m=(max-min);

m=min+double(rand())/RAND\_MAX\*m ;

return m;

}

//利用类模板生成实例

typedef vector < int > IntArray;

typedef list <int> LISTINT;

typedef set<int> SET\_INT;

//在main()函数中测试accumulate算法

void main ()

{

//--------------------------------------------

// count算法对于普通数组的计算

//---------------------------------------------

int x[size];

cout<<"x[]:";

for (int i=0;i<size;i++) {

x[i]=getrand(1,3);

cout<<x[i]<<" ";

}

cout<<endl;

cout<<"count(x,x+size,2)=";

cout<<count(x,x+size,2)<<endl;

cout<<"count(x+2,x+8,2)=";

cout<<count(x+2,x+8,2)<<endl;

//--------------------------------------------

// count算法对于vector容器的计算

//---------------------------------------------

//声明intvector容器和迭代器ii

IntArray intvector;

IntArray::iterator ii;

//向intvector容器中插入元素

for (i=1; i<size; i++) {

intvector.push\_back(getrand(2,6));

};

//显示intvector容器中的元素值和统计结果

cout << "intvector: ";

for (ii=intvector.begin();ii !=intvector.end();++ii)

cout<<(\*ii)<<" ";

cout<<endl;

cout<<"count(intvector.begin(),intvector.end(),4)=";

cout<<count(intvector.begin(),intvector.end(),4)<<endl;

//--------------------------------------------

// count算法对于list容器的计算

//---------------------------------------------

//声明list容器对象和迭代器

LISTINT::iterator iL;

LISTINT list1;

//向list1容器对象中插入元素并显示

for (i=1; i<size; i++) {

list1.push\_front(getrand(3,5));

};

//显示list1容器的元素值和统计结果

cout << "list1: ";

for (iL=list1.begin();iL !=list1.end();++iL)

cout<<(\*iL)<<" ";

cout<<endl;

cout<<"count(list1.begin(),list1.end(),3)=";

cout<<count(list1.begin(),list1.end(),3)<<endl;

//--------------------------------------------

// count算法对于set容器的计算

//---------------------------------------------

//声明set容器对象和迭代器

SET\_INT set1;

SET\_INT::iterator si;

//向set1容器中插入元素

for (i=1; i<size; i++) {

set1.insert(getrand(1,10));

};

//显示set1容器的元素值和统计结果

cout <<"set1: ";

for (si=set1.begin();si !=set1.end();++si)

cout<<(\*si)<<" ";

cout<<endl;

cout<<"count(set1.begin(),set1.end(),5)=";

cout<<count(set1.begin(),set1.end(),5)<<endl;

}

#include <iostream>

#include <algorithm>

#include <string>

#include <vector>

using namespace std;

//如果字符串以'S'开头，则返回true

int MatchFirstChar( const string& str)

{

string s("S") ;

return s == str.substr(0,1) ;

}

//测试count\_if算法

void main()

{

const int VECTOR\_SIZE = 8 ;

//生成成员类型为strings的vector容器类

typedef vector<string > StringVector ;

//定义迭代器类型

typedef StringVector::iterator StringVectorIt ;

//声明vector容器的对象

StringVector NamesVect(VECTOR\_SIZE) ;

//声明迭代器

StringVectorIt start, end, it ;

int result = 0 ; // 存放统计数据

//初始化vector容器NamesVect

NamesVect[0] = "She" ;

NamesVect[1] = "Sells" ;

NamesVect[2] = "Sea" ;

NamesVect[3] = "Shells" ;

NamesVect[4] = "by" ;

NamesVect[5] = "the" ;

NamesVect[6] = "Sea" ;

NamesVect[7] = "Shore" ;

//设置容器的起始位置和终止位置

start = NamesVect.begin() ;

end = NamesVect.end() ;

//显示NamesVect容器的元素

cout << "NamesVect: " ;

for(it = start; it != end; it++)

cout << \*it << " " ;

cout <<endl ;

//统计并显示NamesVect容器的所有元素中以'S'字符开头的字符串

result = count\_if(start, end, MatchFirstChar) ;

cout << "Number of elements that start with letter \"S\" = "

<< result << endl ;

//显示NamesVect容器[1,6]之间的元素

cout <<"NamesVect[1]--NamesVect[6]: " ;

for(it =&NamesVect[1]; it != &NamesVect[7]; it++)

cout << \*it << " " ;

cout <<endl ;

//统计并显示NamesVect容器的所有元素中以'S'字符开头的字符串

result = count\_if(&NamesVect[1], &NamesVect[7], MatchFirstChar) ;

cout << "Number of elements that start with letter \"S\" = "

<< result << endl ;

}

#include <iostream>

#include <algorithm>

#include <vector>

using namespace std;

//利用类模板生成实例

typedef vector < int > IntArray;

//显示数组

void put\_array(int x[],int size) {

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

cout<<x[i]<<" ";

cout<<endl;

}

//显示vector容器中的元素

void put\_vector(IntArray v)

{

IntArray::iterator theIterator;

for (theIterator=v.begin();theIterator!=v.end();++theIterator){

cout<<(\*theIterator)<<" ";

}

cout<<endl;

}

//在main()函数中测试fill和fill\_n算法

void main ()

{

//--------------------------------------------

// fill和fill\_n算法对普通数组的计算

//---------------------------------------------

int x[]={1,3,5,7,9};

cout << "x[]: ";

put\_array(x,5);

//填数处理

fill(x+1,x+3,2);

cout << "fill(x+1,x+3,2): "<<endl;

put\_array(x,5);

fill\_n(x,3,8);

cout << "fill\_n(x,3,8): "<<endl;

put\_array(x,5);

//--------------------------------------------

// fill和fill\_n算法对于vector容器的计算

//---------------------------------------------

//声明intvector容器和迭代器ii

IntArray intvector;

//向intvector容器中插入元素

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

intvector.push\_back(i);

};

//显示intvector容器中的元素值和统计结果

cout << "intvector: "<<endl;

put\_vector(intvector);

//填数处理

fill(intvector.begin(),intvector.begin()+3,2);

cout << "fill(intvector.begin(),intvector.begin()+3,2): "<<endl;

put\_vector(intvector);

fill\_n(&intvector[5],3,8);

cout << "fill\_n(&intvector[5],3,8): "<<endl;

put\_vector(intvector);

}

#include <iostream>

#include <algorithm>

#include <vector>

#define ARRAY\_SIZE 10

using namespace std;

//利用类模板生成实例

typedef vector < int > IntArray;

//显示数组

void put\_array(int x[],int size) {

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

cout<<x[i]<<" ";

cout<<endl;

}

//显示vector容器中的元素

void put\_vector(IntArray v)

{

IntArray::iterator theIterator;

for (theIterator=v.begin();theIterator!=v.end();++theIterator){

cout<<(\*theIterator)<<" ";

}

cout<<endl;

}

//在main()函数中测试find()算法

void main ()

{

int i,value,\*p;

//--------------------------------------------

// find()算法对于普通数组的处理

//---------------------------------------------

int x[ARRAY\_SIZE]={1,3,5,7,9,2,4,6,8,10};

cout << "x[]: ";

put\_array(x,ARRAY\_SIZE);

//find()算法查找,并显示查找结果

for(i=0;i<=2;i++) {

cout<<"value=";

cin>>value;

p=find(x,x+ARRAY\_SIZE,value);

if (p != x + ARRAY\_SIZE) { //查到

cout << "First element that matches " << value;

cout<< " is at location " << p - x<< endl;

}

else { //未查到

cout << "The sequence does not contain any elements";

cout<< " with value " << value << endl ;

}

}

//--------------------------------------------

// find()算法对于vector容器的处理

//---------------------------------------------

//声明intvector容器对象

IntArray intvector;

//向intvector容器中插入元素

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

intvector.push\_back(i);

};

//显示intvector容器中的元素值

cout << "intvector: ";

put\_vector(intvector);

//find()算法查找,并显示查找结果

IntArray::iterator pos;

for (i=0;i<=2;i++) {

cout<<"value=";

cin>>value;

pos=find(intvector.begin(),intvector.end(),value);

if (pos != intvector.end()) { //查到

cout << "First element that matches " << value;

cout<< " is at location " <<pos - intvector.begin()<< endl;

}

else { //未查到

cout << "The sequence does not contain any elements";

cout<< " with value " << value << endl ;

}

}

}

#include <iostream>

#include <algorithm>

#include <vector>

#define ARRAY\_SIZE 10

using namespace std;

//利用类模板生成实例

typedef vector < int > IntArray;

//显示数组

void put\_array(int x[],int size) {

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

cout<<x[i]<<" ";

}

//显示vector容器中的元素

void put\_vector(IntArray v)

{

IntArray::iterator theIterator;

for (theIterator=v.begin();theIterator!=v.end();++theIterator){

cout<<(\*theIterator)<<" ";

}

}

//在main()函数中测试find()\_end()算法

void main ()

{

//--------------------------------------------

// find\_end()算法对普通数组的处理

//---------------------------------------------

int x[ARRAY\_SIZE]={1,3,5,7,9,2,4,6,8,10};

cout << "x[]: ";

put\_array(x,ARRAY\_SIZE);

cout<<endl;

int y[]={5,7,9};

cout << "y[]: ";

put\_array(y,3);

cout<<endl;

// find\_end()算法查找,并显示查找结果

int \*p=find\_end(x,x+ARRAY\_SIZE,&y[0],&y[2]);

if (p != x + ARRAY\_SIZE) { //查到

cout << "The first element that matches :" ;

put\_array(y,3);

cout<< " is at location in x" << p - x<< endl;

}

else { //未查到

cout << "The sequence does not contain any elements";

cout<< " with value " ;

put\_array(&x[3],3);

}

//--------------------------------------------

// find\_end()算法对vector容器的处理

//---------------------------------------------

//声明intvector容器对象

IntArray intvector;

//向intvector容器中插入元素

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

intvector.push\_back(i);

};

//显示intvector容器中的元素值

cout << "intvector: ";

put\_vector(intvector);

cout<<endl;

IntArray temp;

temp.push\_back(5);

temp.push\_back(6);

temp.push\_back(7);

cout << "temp: ";

put\_vector(temp);

cout<<endl;

// find\_end()算法查找,并显示查找结果

IntArray::iterator pos;

pos=find\_end(intvector.begin(),intvector.end(),temp.begin(),temp.end());

if (pos != intvector.end()) { //查到

cout << "The first element that matches ";

put\_vector(temp);

cout<< " is at location in intvector " <<pos - intvector.begin()<< endl;

}

else { //未查到

cout << "The sequence does not contain any elements";

cout<< " with value ";

put\_vector(temp);

cout<< endl ;

}

}

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

//返回一个Fibonacci数，其由generate\_n()算法调用

int Fibonacci1(void)

{

static int r;

static int f1 = 0;

static int f2 = 1;

r = f1 + f2 ;

f1 = f2 ;

f2 = r ;

return f1 ;

}

//返回一个Fibonacci数，其由generate()算法调用

int Fibonacci2(void)

{

static int r;

static int f1 = 0;

static int f2 = 1;

r = f1 + f2 ;

f1 = f2 ;

f2 = r ;

return f1 ;

}

//定义整型数的vector容器类

typedef vector<int > IntVector ;

//显示vector容器中的元素

void put\_vector(IntVector v,char \*name)

{

IntVector::iterator theIterator;

cout<<name<<":"<<endl;

for (theIterator=v.begin();theIterator!=v.end();++theIterator){

cout<<(\*theIterator)<<" ";

}

cout<<endl;

}

//测试generate()和generate\_n()算法

void main()

{

const int VECTOR\_SIZE = 15 ;

//定义迭代器类

typedef IntVector::iterator IntVectorIt ;

//声明vector容器对象

IntVector Numbers1(VECTOR\_SIZE),Numbers2(VECTOR\_SIZE);

int i ;

//初始化vector容器对象

for(i = 0; i < VECTOR\_SIZE; i++)

Numbers1[i] = i ;

//显示vector容器对象的元素

cout << "Before calling generate\_n:" << endl ;

put\_vector(Numbers1,"Numbers1");

//利用generate\_n算法用Fibonacci 数填充vector容器

generate\_n(Numbers1.begin(), VECTOR\_SIZE, Fibonacci1) ;

//显示vector容器对象的元素

cout << "After calling generate\_n:" << endl ;

put\_vector(Numbers1,"Numbers1");

//利用generate算法用Fibonacci 数填充vector容器

generate(Numbers2.begin(),Numbers2.end(), Fibonacci2) ;

//显示vector容器对象的元素

cout << "After calling generate:" << endl ;

put\_vector(Numbers2,"Numbers2");

}

#include <iostream>

#include <algorithm>

#include <vector>

using namespace std;

//利用类模板生成实例

typedef vector < int > IntArray;

//显示数组

void put\_array(int x[],int size) {

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

cout<<x[i]<<" ";

cout<<endl;

}

//显示vector容器中的元素

void put\_vector(IntArray v)

{

IntArray::iterator theIterator;

for (theIterator=v.begin();theIterator!=v.end();++theIterator){

cout<<(\*theIterator)<<" ";

}

cout<<endl;

}

//在main()函数中测试reverse()和reverse\_copy()算法

void main ()

{

//--------------------------------------------

// reverse()和reverse\_copy()算法对普通数组处理

//---------------------------------------------

int x[]={1,3,5,7,9};

cout<<"x[]:";

put\_array(x,5);

//reverse()反转x数组并显示

reverse(x,x+5);

cout<<"x[]:";

put\_array(x,5);

int y[]={2,4,6,8,10};

cout<<"y[]:";

put\_array(y,5);

//reverse\_copy()反转y数组的部分元素并拷贝到x数组第2个元素位置

reverse\_copy(y+1,y+3,x+1);

cout<<"x[]:";

put\_array(x,5);

cout<<"y[]:";

put\_array(y,5);

//--------------------------------------------

// reverse()和reverse\_copy()算法对vector容器的处理

//---------------------------------------------

//声明intvector容器和迭代器ii

IntArray intvector;

//向intvector容器中插入元素

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

intvector.push\_back(i);

};

//显示intvector容器中的元素值

cout << "intvector: "<<endl;

put\_vector(intvector);

//reverse()对于vector容器的处理

reverse(intvector.begin(),intvector.end());

cout << "intvector: "<<endl;

put\_vector(intvector);

// reverse\_copy对于vector容器的处理

IntArray temp(5);

reverse\_copy(intvector.begin()+2,intvector.begin()+7,temp.begin());

cout << "temp: "<<endl;

put\_vector(temp);

}

#include <iostream>

#include <algorithm>

#include <vector>

#include <stdlib.h>

#define ARRAY\_SIZE 15

using namespace std;

//定义整型数的vector容器类

typedef vector<int > IntVector ;

//显示数组

void put\_array(int x[],int size) {

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

cout<<x[i]<<" ";

cout<<endl;

}

//显示vector容器中的元素

void put\_vector(IntVector v,char \*name)

{

IntVector::iterator theIterator;

cout<<name<<": ";

for (theIterator=v.begin();theIterator!=v.end();++theIterator){

cout<<(\*theIterator)<<" ";

}

cout<<endl;

}

//产生指定范围的整数随机数

int getrand(int min,int max) {

int m;

m=(max-min);

m=min+double(rand())/RAND\_MAX\*m ;

return m;

}

//在main()函数中测试sort()和partial\_sort()算法

void main ()

{

int i;

//--------------------------------------------

// sort()和partial\_sort()算法对普通数组处理

//---------------------------------------------

//sort()算法处理数组，并显示

int x[ARRAY\_SIZE];

for (i=0;i<ARRAY\_SIZE;i++) {

x[i]=getrand(1,20);

}

cout<<"x[]:";

put\_array(x,ARRAY\_SIZE);

sort(x,x+ARRAY\_SIZE);

cout<<"sort(x,x+ARRAY\_SIZE):"<<endl;

put\_array(x,ARRAY\_SIZE);

//partial\_sort()算法对于数组进行处理

int y[ARRAY\_SIZE];

for (i=0;i<ARRAY\_SIZE;i++) {

y[i]=getrand(1,30) ;

}

cout<<"y[]:";

put\_array(y,ARRAY\_SIZE);

partial\_sort(y+2,y+7,y+ARRAY\_SIZE);

cout<<"partial\_sort(y+2,y+7,y+ARRAY\_SIZE):"<<endl;

put\_array(y,ARRAY\_SIZE);

//--------------------------------------------

// sort()和partial\_sort()算法对vector容器的处理

//---------------------------------------------

IntVector Numbers1,Numbers2;

for(i=0;i<15;i++) {

Numbers1.push\_back(getrand(1,30));

Numbers2.push\_back(getrand(1,30));

}

put\_vector(Numbers1,"Numbers1");

put\_vector(Numbers2,"Numbers2");

//sort()算法处理并显示

sort(Numbers1.begin(),Numbers1.end());

cout<<"After call sort():"<<endl;

put\_vector(Numbers1,"Numbers1");

//partial\_sort()算法处理并显示

partial\_sort(Numbers2.begin()+2,Numbers2.begin()+7,Numbers2.end());

cout<<"After call partial\_sort():"<<endl;

put\_vector(Numbers2,"Numbers2");

}

#include <iostream>

#include <algorithm>

#include <stdlib.h>

#include <time.h>

#define ARRAY\_SIZE 15

using namespace std;

//显示数组

void put\_array(int x[],int size) {

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

cout<<x[i]<<" ";

cout<<endl;

}

//产生指定范围的整数随机数

int getrand(int min,int max) {

int m;

m=(max-min);

m=min+double(rand())/RAND\_MAX\*m ;

return m;

}

//在main()函数中测试max\_element()和 min\_element()算法

void main ()

{

//声明变量和数组

int i;

int x[ARRAY\_SIZE];

//用1到100的随机数初始化数组，并显示

srand( (unsigned)time( NULL ) );

for (i=0;i<ARRAY\_SIZE;i++) {

x[i]=getrand(1,100);

}

cout<<"x[]:";

put\_array(x,ARRAY\_SIZE);

//对数组x使用max\_element()算法，并显示

int \*pMax=max\_element(x,x+ARRAY\_SIZE);

cout<<"pMax ="<<pMax<<endl;

cout<<"Location="<<(pMax-x)<<endl;

cout<<"\*pMax ="<<(\*pMax)<<endl;

//对数组x使用min\_element()算法，并显示

int \*pMin=min\_element(x,x+ARRAY\_SIZE);

cout<<"pMin ="<<pMin<<endl;

cout<<"Location="<<(pMin-x)<<endl;

cout<<"\*pMin ="<<(\*pMin)<<endl;

}