# 《C++程序设计基础》（第4版）（上）

# 习题与解答

## 第1章 基本数据与表达式

#### 1.1 选择题

1．一个最简单的C++程序，可以只有一个（ C ）。

（A）库函数 （B）自定义函数 （C）main函数 （D）空函数

2．用C++语言编写的源程序要成为目标程序必须要经过（ D ）。

（A）解释 （B）汇编 （C）编辑 （D）编译

3．C++程序中的简单语句必须以（ B ）结束。

（A）冒号 （B）分号 （C）空格 （D）花括号

4．假设有说明 int a=0; double x=5.16;，则在以下语句中，（ C ）属于编译错误。

（A）x=a/x; （B）x=x/a; （C）a=a%x; （D）x=x\*a;

5．执行C++程序时出现的“溢出”错误属于（ C ）错误。

（A）编译 （B）连接 （C）运行 （D）逻辑

6．在下列选项中，全部都是C++关键字的选项为（ C ）。

（A）while IF Static （B）break char go

（C）sizeof case extern （D）switch float integer

7．按C++标识符的语法规定，合法的标识符是（ A ）。

（A）\_abc （B）new （C）π （D）"age"

8．在C++语句中，两个标识符之间（ A ）不能作为C++的分隔符。

（A）数字 （B）; （C）: （D）+

9．下列正确的八进制整型常量表示是（ B ）。

（A）0a0 （B）015 （C）080 （D）0x10

10．下列错误的十六进制整型常量表示是（ C ）。

（A）0x11 （B）0xaf （C）0xg （D）0x1f

11．在下列选项中，全部都合法的浮点型数据的选项为（ B ）。

（A）−1e3.5 15. 2e−4 （B）12.34 −1e+5 0.1E−12

（C）0.2e−2 −12345. e−5 （D）5.0e（1+4） 0.1 8e+2

12．在下列选项中，正确的字符常量为（ D ）。

（A）"a" （B）'name' （C）a （D）'\101'

13．在下列选项中，（ D ）不能交换变量a和b的值。

（A）t=b; b=a; a=t; （B）a=a+b; b=a-b; a=a–b;

（C）t=a; a=b; b=t; （D）a=b; b=a;

14．关于下列语句，叙述错误的是（ A ）。

int i=10， \*p=&i;

（A）p的值为10 （B）p指向整型变量i

（C）\*p表示变量i的值 （D）p的值是变量i的地址

15．有以下变量说明，下面不正确的赋值语句是（ B ）。

int a=5, b=10, c; int \*p1 = &a, \*p2 = &b;

（A）\*p2 = b; （B）p1 = a;

（C）p2 = p1; （D）c = \*p1 \*(\*p2);

16．有以下变量说明，下面正确的语句是（ B ）。

int a=10, b; int &pa=a, &pb=b;

（A）&pb = a; （B）pb = pa; （C）pb = &pa; （D）\*pb = \*pa;

17．执行下面语句序列后，a和b的值分别为（ B ）。

int a=5, b=3, t;

int &ra=a;

int &rb=b;

t=ra; ra=rb; rb=t;

（A）3和3 （B）3和5 （C）5和3 （D）5和5

18．在下列运算符中，（ D ）优先级最高。

（A）<= （B）\*= （C）+ （D）\*

19．在下列运算符中，（ D ）优先级最低。

（A）! （B）&& （C）!= （D）? :

20．已知int i=1, j=2;，则表达式 i+++j 的值为（ C ）。

（A）1 （B）2 （C）3 （D）4

21．已知int i=1, j=2;，则表达式 ++i+j 的值为（ D ）。

（A）1 （B）2 （C）3 （D）4

22．在下列表达式选项中，（ C ）是正确。

（A）++(a++) （B）a++b （C）a+++b （D）a++++b

23．已知 int i=0, j=1, k=2;，则逻辑表达式 ++i||− −j&&++k 的值为（ B ）。

（A）0 （B）1 （C）2 （D）3

24．执行下列语句后，x的值是（ D ），y的值是（ ）。

int x, y;

x=y=1; ++x || ++y;

（A）不确定 （B）0 （C）1 （D）2

25．设x为整型变量，不能正确表达数学关系 1＜x＜5 的C++逻辑表达式是（ A ）。

（A）1< x <5 （B）x==2||x==3||x==4

（C）1<x && x<5 （D）! （x<=1）&&! （x>=5）

26．已知 int x=5;，执行下列语句后，x的值为（ C ）。

x+=x-=x\*x;

（A）25 （B）40 （C）–40 （D）20

27．设 int a=1, b=2, c=3, d=4;，则以下条件表达式的值为（ A ）。

a<b?a:c<d?c:d

（A）1 （B）2 （C）3 （D）4

28．以下逗号表达式的值为（ D ）。

(x=4\*5, x\*5), x+25

（A）25 （B）20 （C）100 （D）45

#### 1.2 根据下列数学表达式写出C++算术表达式

1． 2．*x*{*x*[*x*(*ax*+*b*)+*c*]+*d*}+*e*

3．ln 4．

5．cot 6．lg(*a*2+*ab*+*b*2)

【解答】

1. 1/(1 + 1/(1 + 1/(x + y)))

2. x \* ( x \* ( x \* ( a \* x + b ) + c ) + d ) + e

3. log( 1 + pow( fabs( ( a + b )/( a – b ) )，10)

4. sqrt( 1 + 3.14159/2 \* cos( 48 \* 3.14159/180 ) )

5. 1/tan( ( 1 - x\*x )/( 1 + x\*x))

或者 cos( ( 1 - x\*x )/( 1 + x\*x ) )/sin( ( 1 - x\*x )/( 1 + x\*x ) )

6. log10( a \* a + a \* b + b \* b )

#### 1.3 用逻辑表达式表示下列条件

1．*i* 被*j*整除 2．*n*是小于正整数*k*的偶数

3．1≤*x*＜10 4．*x*、*y*其中有一个小于*z*

5．y[–100,–10]，并且 *y*[10,100]

6．坐标点（*x*, *y*）落在以（10, 20）为圆心，以35为半径的圆内

7．三条边*a*，*b*和*c*构成三角形

8．年份Year能被4整除，但不能被100整除，或者能被400整除

【解答】

1. i%j == 0 2.（n<k）&&（n%2 == 0）

3. 1<=x && x<10 4. x<z||y<z

5. !( y>=-100 && y<=-10 ) && !( y>=10 && y<=100 )

6. sqrt(pow((x-10),2) + pow((y-20),2))< 35

7. a+b>c && b+c>a && c+a>b

8. (year%4 == 0) && (year%100!=0)||(year%400==0)

#### 1.4 阅读下列程序，写出运行结果

1．

#include <iostream>

using namespace std;

int main()

{

int a = 1, b = 2;

bool x, y;

cout << (a++)+(++b) << endl;

cout << a % b << endl;

x = !a>b;

y = a-- && b;

cout << x << endl;

cout << y << endl;

}

【解答】

4

2

0

1

2．

#include <iostream>

using namespace std;

int main()

{

int x,y,z,f;

x = y = z = 1;

f = --x || y-- && z++;

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

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

cout << "z = " << z << endl;

cout << "f = " << f << endl;

}

【解答】

x=0

y=0

z=2

f=1

3．

#include <iostream>

#include<iomanip>

using namespace std;

int main()

{

int a=123;

int &ra=a;

int \*pa=&a;

cout<<setw(5)<<dec<<a<<setw(5)<<oct<<ra<<setw(5)<<hex<<\*pa<<endl;

}

【解答】

123 173 7b

#### 1.5 思考题

1．什么是数据类型？变量的类型定义有什么作用？

【解答】

数据“类型”是对数据的抽象。类型相同的数据有相同的表示形式、存储格式以及相关的操作。定义一个变量时，计算机根据变量的类型分配存储空间，并以该类型解释存放的数据。

2．普通数据类型变量和指针类型变量的定义、存储和使用方式有何区别？请编写一个程序验证之。

【解答】

|  |  |  |  |
| --- | --- | --- | --- |
| **变量类型** | **定义** | **存储** | **使用方式** |
| 数据 | 类型 标识符 | 数据值 | 通过名访问即直接访问对变量内容操作 |
| 指针 | 类型 \* 标识符 | 地址值 | 通过指针变量的地址值间址访问对象 |

验证程序：

#include<iostream>

using namespace std;

int main()

{ int a,b,c;

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

cin>>a>>b>>c; //对普通数据类型变量赋值

int \*pa=&a,\*pb=&b,\*pc=&c; //用变量地址值初始化指针变量

cout<<"a,b,c= "<<a<<", "<<b<<", "<<c<<endl; /名访问，输出a,b,c的值

cout<<"pa,pb,pc= "<<pa<<", "<<pb<<", "<<pc<<endl; //输出指针变量的地址值

//间址访问，输出pa,pb,pc指向的变量的赋值

cout<<"\*pa,\*pb,\*pc= "<<\*pa<<", "<<\*pb<<", "<<\*pc<<endl;

}

3．什么是数据对象的引用？对象的引用和对象的指针有什么区别？请用一个验证程序说明之。

【解答】

引用是为数据对象定义别名。引用与指针有以下几点区别：

（1）引用名不是内存变量，而指针变量要开辟内存空间。

（2）引用名需要在变量定义与变量名绑定，并且不能重定义；指针变量可以在程序中赋给不同的地址值，改变指向。

（3）程序中用变量名和引用名访问对象的形式和效果一样；指针变量通过间址访问对象。

验证程序：

#include<iostream>

using namespace std;

int main ()

{ int a;

cout<<"a=";

cin>>a;

int ra=a;

int \*pa=&a;

cout<<"a的值："<<a<<endl;

cout<<"a的地址："<<&a<<endl;

cout<<"ra的值："<<ra<<endl;

cout<<"ra的地址："<<&ra<<endl;

cout<<"pa所指向的变量的值："<<\*pa<<endl;

cout<<"pa的地址："<<pa<<endl;

}

4．数据对象在C++中有几种不同的访问方式？请编写一个程序验证之。

【解答】

数据对象在C++中的访问方式有：名访问，引用（别名）访问，间址访问。

验证程序：

#include<iostream>

using namespace std;

int main()

{ int a;

cout<<"a=";

cin>>a;

a=a+5; //名访问

cout<<&a<<endl; //输出变量地址

cout<<\*(&a)<<endl; //地址访问，输出变量值

int \*pa=&a; //说明指针变量，指向变量a

cout<<\*pa<<endl; //间址访问，输出变量值

int &ra=a; //ra是a的引用

cout<<ra<<endl; //引用访问，输出变量a的值

}

5．为了约束对数据对象的值进行只读操作，C++采用什么方式？请进行简要归纳。

【解答】

约束数据对象只读形式如下：

|  |  |
| --- | --- |
| **约束对象** | **说明形式** |
| 标识常量 | const 类型 常量标识符=常量表达式; |
| 指针常量 | 类型 \* const 指针; |
| 指向常量的指针 | const 类型 \* 指针; 或者 类型 const \* 指针; |
| 指向常量的指针常量 | const 类型 \* const 指针; 或者 类型 const \* const 指针; |
| 常引用 | const 类型 & 引用名 = 对象名; |

6．什么是表达式？表达式值的类型由什么因素决定？使用不同运算符连接以下3个变量，请写出5个以上获得值为true的表达式。

int a=1, b=2; double x=0.5;

【解答】

表达式是由数据和运算符，按求值规则，表达一个值的式子。

表达式值的类型的决定因素为操作数的类型。

（1）如果运算符左右操作数类型相同，运算结果也是相同类型。

（2）如果运算符左右操作数类型不同，首先把类型较低（存储要求，示数能力较低）的数据转换成类型较高的数据，然后运算。

（3）赋值表达式的类型由被赋值变量的类型决定。当把一个表达式的值赋给一个变量时，系统首先强制把运算值转换成变量的类型，然后执行写操作。

6个值等于true的表达式：

（1）b>a && a>x （2）(a+b)!=x （3）a||(b+x)

（4）a==(b\*x) （5）a-b<x （6）(a/x==b)

7．阅读以下程序，分析下面语句序列中每一个字符“\*”和“&”的含义，并写出输出结果。

#include <iostream>

using namespace std;

int main()

{

int a=10, b=20;

int \*p = &a, \*q = &b;

\*p = \*p \* \*q;

int & ra = a;

ra=a;

int \* & rt = q;

\*rt = 30;

cout<<"a="<<a<<"\nb="<<b<<"\n\*p="<<\*p<<"\n\*q="<<\*q

<<"\nra="<<ra<<"\n\*rt="<<\*rt<<endl;

}

【解答】

字符“\*”和“&”的意义见程序中添加的注释。

#include <iostream>

using namespace std;

int main()

{

int a=10, b=20;

int \*p = &a, \*q = &b; //“\*”是指针类型说明符，“&”是取址运算符

\*p = \*p \* \*q; //第1、2、4个“\*”是间址访问符，第3个“\*”算术乘运算符

int & ra = a; //“&”是引用说明符

ra=a;

int \* & rt = q; //“\*”是指针类型说明符，“&”是引用说明符

\*rt = 30; //“\*”是间址访问符

//输出语句中的“\*”是间址访问符

cout<<"a="<<a<<"\nb="<<b<<"\n\*p="<<\*p<<"\n\*q="<<\*q

<<"\nra="<<ra<<"\n\*rt="<<\*rt<<endl;

}

程序输出结果为：

a=200

b=30

\*p=200

\*q=30

ra=200

\*rt=30

#### 1.6 编程题

1．输入一个3位整数，将它反向输出。

【解答】

#include <iostream>

using namespace std;

int main()

{

int x,i,j,k;

cout << "please input x:";

cin >> x;

i = x/100;

j = x/10 %10;

k = x%10;

cout << k << j << i << endl;

}

2．输入平面上某点横坐标*x*和纵坐标*y*，若该点位于图1.11所示的方块区域内，则输出1；否则，输出0。

【解答】

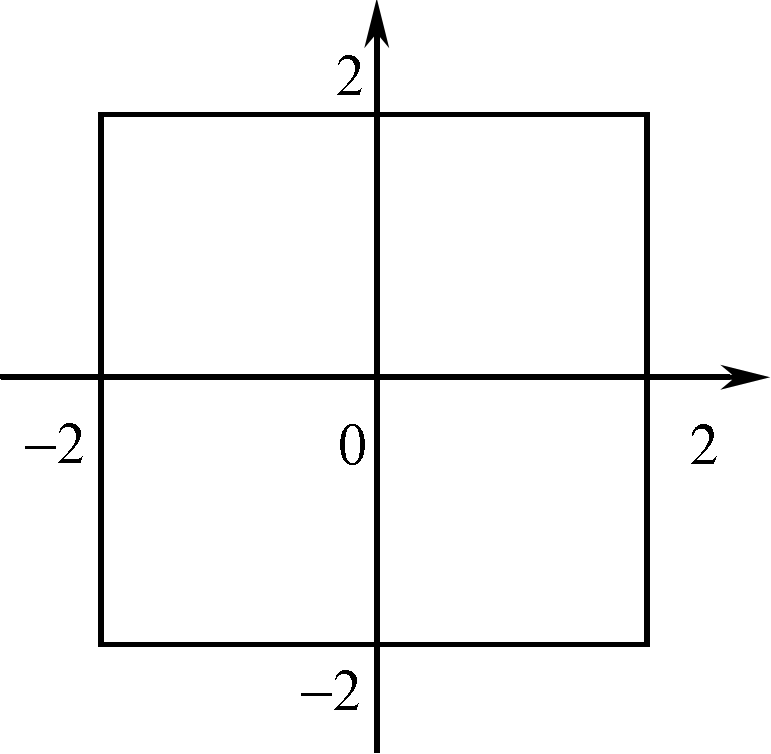


图1.11 方形区域

#include <iostream>

using namespace std;

int main()

{

double x,y,b;

cout << "please input x,y:";

cin >> x >> y;

b = ( -2<=x ) && ( x<=2 ) && ( -2<=y ) && ( y<=2 );

cout << b << endl;

}

3．输入3个整数，求出其中最小数（要求使用条件表达式）。

【解答】

#include <iostream>

using namespace std;

int main()

{

int a,b,c,temp,min;

cout << "please input a,b,c:";

cin >> a >> b >> c;

temp = ( a<b ) ? a:b;

min = ( temp<c ) ? temp:c;

cout << "min=" << min << endl;

}

## 第2章 程序控制结构

#### 2.1 选择题

1．已知 int i=0, x=1, y=0;，在下列选项中，使i的值变成1的语句是( C )。

（A）if( x&&y ) i++; （B）if( x==y ) i++;

（C）if( x||y ) i++; （D）if( !x ) i++;

2．设有函数关系为*y*=，下列选项中，能正确表示上述关系的是（ C ）。

（A） y = 1; （B） y = -1;

if( x >= 0 ) if( x != 0 )

if( x == 0 ) y = 0; if( x > 0 ) y = 1;

else y = -1; else y = 0

（C） if( x <= 0 ) （D） y = -1;

if( x < 0 ) y = -1; if( x <= 0 )

else y = 0; if( x < 0 ) y = -1;

else y = 1; else y = 0;

3．假设i=2，执行下列语句后i的值为（ B ）。

switch( i )

{

case 1 : i ++;

case 2 : i --;

case 3 : ++ i; break;

case 4 : -- i;

default : i ++;

}

（A）1 （B）2 （C）3 （D）4

4．已知int i=0，x=0;，在下面while语句执行时循环次数为（ D ）。

while( !x && i< 3 ) { x++; i++; }

（A）4 （B）3 （C）2 （D）1

5．已知int i=3;，在下面do\_while 语句执行时的循环次数为（ B ）。

do{ i--; cout<<i<<endl;}while（ i!= 1 ）;

（A）1 （B）2 （C）3 （D）无限

6．下面for语句执行时的循环次数为（ B ）。

int i, j;

for ( i=0, j=5;i=j; )

{ cout<<i<<j<< ndl; i++; j--; }

（A）0 （B）5 （C）10 （D）无限

7．以下程序段形成死循环的是（ B ）。

（A）int x; for( x=0; x<3; ) { x++; };

（B）int k = 0; do { ++k; } while( k>=0 );

（C）int a=5; while( a ) { a--; };

（D）int i=3; for(; i; i -- );

#### 2.2 阅读下列程序，写出运行结果

1．

#include<iostream>

using namespace std;

int main()

{

int a,b,c,d,x;

a = c = 0; b = 1; d = 20;

if( a )

d = d-10;

else

if( !b )

if( !c )

x = 15;

else x = 25;

cout << d << endl;

}

【解答】

20

2．

#include<iostream>

using namespace std;

int main()

{

int a = 0, b = 1;

switch( a )

{

case 0:

switch( b )

{

case 0 : cout<<"a="<<a<<" b="<<b<<endl; break;

case 1 : cout<<"a="<<a<<" b="<<b<<endl; break;

}

case 1:

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

}

}

【解答】

a= 0 b= 1

a= 1 b= 2

3．

#include<iostream>

using namespace std;

int main()

{

int i = 1;

while( i<=10 )

{

if( ++i % 3 != 1 )

continue;

else

cout << i << endl;

}

}

【解答】

4

7

10

4．

#include<iostream>

using namespace std;

int main()

{

int i = 0, j = 5;

do

{

i++; j--;

if ( i>3 ) break;

} while ( j>0 );

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

}

【解答】

i= 4 j= 1

5．

#include<iostream>

using namespace std;

int main()

{

int i,j;

for( i=1, j=5; i<j; i++ )

{ j--; }

cout<<i<<'\t'<<j<<endl;

}

【解答】

3 3

6．

#include<iostream>

using namespace std;

int main()

{

int i, s = 0;

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

switch( i )

{

case 0: s += i; break;

case 1: s += i; break;

case 2: s += i; break;

default: s += 2;

}

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

}

【解答】

s= 7

7．

#include<iostream>

using namespace std;

int main()

{

int i, j, x = 0;

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

{

x++;

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

{

if( j % 2 )

continue;

x++;

}

x++;

}

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

}

【解答】

x= 16

#### 2.3 思考题

1．C++语言中有什么形式的选择控制语句？归纳它们的语法形式、应用场合。根据一个实际问题使用不同的条件语句编程。

【解答】

|  |  |  |
| --- | --- | --- |
| **语句** | **使用方式** | **使用场合** |
| if语句 | if(表达式)语句1;  else 语句2; | 需要对给定的条件进行判断，并根据判断的结果选择不同的操作。  适用于复杂的条件表达式判断。 |
| switch 语句 | switch(表达式)  { case 常量表达式1: 语句1;  case 常量表达式2: 语句2;  ……  case 常量表达式n; 语句n;  [default : 语句n+1;]  } | 根据整型表达式的不同值决定程序分支的情况。  适用于判断表达式简单，需要多个分支处理的情况。 |

演示程序：

**程序（1）**

//此程序用if输出等级对应的分数段

//A->=90,B-(90,80],C-(80,70] ,D-(70,60],,E-<60

#include<iostream>

using namespace std;

int main()

{ char gd;

cout<<"Enter the grade:";

cin>>gd;

//直到输入有效等级，否则程序不继续运行

while(!((gd>='A' && gd<='E')||(gd>='a' && gd<='e')))

{ cout<<"Invalid grade! Please retry:";

cin>>gd;

}

if(gd=='A'||gd=='a') cout<<"\nScored 90-100!\n";

else if(gd=='B'||gd=='b') cout<<"\nScored 80-89!\n";

else if(gd=='C'||gd=='c') cout<<"\nScored 70-79!\n";

else if(gd=='D'||gd=='d') cout<<"\nScored 60-69!\n";

else if(gd=='E'||gd=='e') cout<<"\nScore under 60!\n";

else cout<<"Unexpect error!\n"; //防止意外错误

}

**程序（2）**

//此程序用switch输出等级对应的分数段

//A->=90,B-(90,80],C-(80,70] ,D-(70,60],,E-<60

#include<iostream>

using namespace std;

int main()

{ char gd;

cout<<"Enter the grade:";

cin>>gd;

//直到输入有效等级，否则程序不继续运行

while(!((gd>='A' && gd<='E')||(gd>='a' && gd<='e')))

{ cout<<"Invalid grade! Please retry:";

cin>>gd;

}

switch(gd)

{ case 'A':

case 'a': cout<<"\nScored 90-100!\n";break;

case 'B':

case 'b': cout<<"\nScored 80-89!\n";break;

case 'C':

case 'c':cout<<"\nScored 70-79!\n";break;

case 'D':

case 'd':cout<<"\nScored 60-69!\n";break;

case 'E':

case 'e':cout<<"\nScore under 60!\n";break;

default:cout<<"Unexpect error!\n";//防止意外错误

}

}

2．什么叫做循环控制？归纳比较C++语言中各种循环控制语句的语法、循环条件和循环结束条件的表示形式及执行流程。

【解答】

循环控制是在特定的条件下，程序重复执行一些特定动作。

|  |  |  |  |
| --- | --- | --- | --- |
| **语句** | **语法** | **执行流程** | **使用场合** |
| while语句 | while(表达式)  循环体;  循环条件：表达式值为非0(真)  循环结束条件：表达式值为0(假) |  | 程序中常用于根据条件执行操作而不需关心循环次数的情况。  先判断形式循环，条件不成立时不进入循环体。 |
| do-while语句 | do  循环体  while(表达式);  循环条件：表达式值为非0(真)  循环结束条件：表达式值为0(假) |  | 程序中常用于根据条件执行操作而不需关心循环次数。  后判断形式循环，至少执行1次循环体。  一般情况，while语句和do while语句可以互换使用。 |
| for 语句 | for([表达式1];[表达式2];[表达式3])  循环体;  （1）表达式1称为初始化表达式，不是循环体执行部分。  （2）表达式3称为后置表达式，作为循环体的最后一个执行表达式。  （3）循环条件：表达式2值为非0（真）  循环结束条件：表达式2值为0（假） |  | for语句称为步长循环语句，通常用于确定循环次数的情况。  由于语句的3个表达式均可以缺省，也可以用于条件循环，即循环次数不确定的情况。 |

3．根据一个实际问题，用不同的循环语句编程，分析其优缺点。

【解答】

略。

4．用if语句和goto语句组织循环，改写上面第3题编写的程序，并分析在什么情况下可以适当使用goto语句。

【解答】

在不破坏程序基本流程控制的情况下，可以适当使用goto语句实现从语句结构内部向外的必要跳转，即按特定条件结束结构语句块的执行。

程序略。

5. 有以下程序，希望判断两个输入的整数是否相等。程序通过编译，但不能达到预期结果。请分析程序能够通过C++编译而不能得到期望结果的原因。

#include<iostream>

using namespace std;

int main()

{

int a,b;

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

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

if( a=b )

cout<<a<<"等于"<<b<<endl;

else

cout<<a<<"不等于"<<b<<endl;

}

运行程序，输入a的值为4，b的值为9，显示结果如下：

a: 4

b: 9

9等于9

【解答】

在if语句的判断表达式(a=b)中，赋值号“=”应该是逻辑等“==”。从语法上，C++的if语句把a=b这个赋值表达式视为逻辑表达式，没有编译错误。a=b的值决定于b。若b的输入值不等于0，if语句的判断表达式作为逻辑真（true），否则作为逻辑假（false）。所以，题目中输入b的值虽然不等于a，但表达式a=b为逻辑true，执行了if语句的第1个分支。

#### 2.4 编程题

1．输入某学生成绩，若成绩在85分以上，则输出“very good”；若成绩在60～85分之间，则输出“good”；若成绩低于60分，则输出“no good”。

【解答】

#include<iostream>

using namespace std;

int main()

{

double score;

cout << "please input score:";

cin >> score;

if ( score>=85 ) cout << "Very good!" ;

else if ( score>=60 ) cout << "Good!";

else cout << "No good!";

}

2．输入3个整数，按从小到大的顺序输出它们的值。

【解答】

#include<iostream>

using namespace std;

int main()

{

int a, b, c, t;

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

cin >> a >> b >> c;

if(a>b) { t=a; a=b; b=t; }

if(a>c) { t=a; a=c; c=t; }

if(b>c) { t=b; b=c; c=t; }

cout<< a << '\t'<< b << '\t' << c << endl;

}

3．输入三角形的3条边，判别它们能否形成三角形，若能，则判断是等边、等腰三角形，还是一般三角形。

【解答】

#include<iostream>

using namespace std;

int main()

{

double a, b, c ;

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

cin >> a >> b >> c ;

if ( a+b > c && b+c > a && c+a > b )

{

if ( a == b && b == c )

cout << "等边三角形！" << endl;

else if ( a == b || a == c || b == c )

cout << "等腰三角形！" << endl;

else cout << "一般三角形！" << endl;

}

else

cout << "不能形成三角形！" << endl ;

}

4．输入百分制成绩，并把它转换成五级分制，转换公式为：

A（优秀） 90～100

B（良好） 80～89

C（中等） 70～79

D（合格） 60～69

E（不合格） 0～59



grade（级别）＝

【解答】

#include<iostream>

using namespace std;

int main()

{

double score; char grade;

cout << "score=";

cin >> score;

if ( score >= 0 && score <= 100 )

{

switch ( int( score ) /10 )

{

case 10:

case 9: grade = 'a'; break;

case 8: grade = 'b'; break;

case 7: grade = 'c'; break;

case 6: grade = 'd'; break;

case 5:

case 4:

case 3:

case 2:

case 1:

case 0: grade = 'e'; break;

}

}

else

{

cout <<"数据输入错误！"<< endl;

goto end;

}

cout << grade << endl;

end: ; //分号不能省

}

5．编程模拟剪刀、石头和布游戏。游戏规则为：剪刀剪纸，石头砸剪刀，布包石头。玩游戏者从键盘输入s（表示剪刀）或r（表示石头）或p（表示布），要求两个游戏者交替输入，计算机给出输赢的信息。

【解答】

#include<iostream>

using namespace std;

int main()

{

char first,second;

cout << "First input( s,r or p ):";

cin >> first;

cout << "Second input( s,r or p ):";

cin >> second;

switch ( first )

{

case 's':

switch ( second )

{

case 's': cout << "Scissor ties scissor." << endl; goto end;

case 'r': cout << "Scissor is crushed by rock." << endl; goto end;

case 'p': cout << "Scissor cuts paper." << endl; goto end;

default : cout << "second input error!" << endl ; goto end;

}

case 'r':

switch ( second )

{

case 's': cout << "Rock crushes scissor." << endl; goto end;

case 'r': cout << "Rock ties rock." << endl; goto end;

case 'p': cout << "Rock is wrapped by paper." << endl; goto end;

default : cout << "second input error!" << endl; goto end;

}

case 'p':

switch ( second )

{

case 's': cout << "Paper is cut by scissor." << endl; goto end;

case 'r': cout << "Paper wraps the rock." << endl; goto end;

case 'p': cout << "Paper ties paper." << endl; goto end;

default : cout << "second input error!" << endl; goto end;

}

default : cout << "First input error!" << endl; goto end;

}

end: ;

}

6．输入一个整数，输出该整数的所有素数因子。例如，输入120，输出为2、2、2、3和5。

【解答】

#include<iostream>

using namespace std;

int main()

{

int m,i = 2;

cout << "please input m:";

cin >> m;

while( i<=m )

if( m % i == 0 )

{

cout << i << ",";

m = m / i;

}

else i++;

}

7．使用迭代公式编程求某一正整数*a*的平方根。

【解答】

#include<iostream>

#include<cmath>

using namespace std;

int main()

{

const double eps = 1e-8;

double a,x0,x;

cout << "please input a:";

cin >> a;

x0 = a / 2;

x = ( x0 + a/x0 )/2;

while( fabs( x-x0 )>eps )

{

x0 = x; x =( x0 + a/x0 )/2;

}

cout << x << endl;

}

8．已知x=0, 10, 20, …, 180，求sin*x*,cos*x*和tan*x*的值。

【解答】

#include<iostream>

#include<cmath>

#include<iomanip>

using namespace std;

int main()

{

const double pi = 3.14159265;

int i;

double x,y1,y2,y3;

cout << setw(2) << "x" << setw(15) << "sin(x)" << setw(15)

<< "cos(x)" << setw(15) << "tg(x)" << endl;

for( i=0; i<=18; i++ )

{

x = i\*10\*pi/180;

y1 = sin( x );

y2 = cos(x);

y3 = y1/y2;

cout << setw(2) << i << setw(15) << y1 << setw(15)

<< y2 << setw(15) << y3 << endl;

}

}

9．在100～200之间找出满足用3除余2，用5除余3和用7除余2的所有整数。

【解答】

#include<iostream>

using namespace std;

int main()

{

int i;

for( i=100; i<=200; i++ )

{

if ( ( i % 3 == 2) && ( i % 5 == 3 ) && ( i % 7 == 2 ) )

cout << i << endl;

}

}

10．求100～999之间的水仙花数。所谓水仙花数，是指一个3位数，它的每位数字的立方之和等于该数。例如，因为153=1+5+3，所以153为水仙花数。

【解答】

#include<iostream>

using namespace std;

int main()

{

int i,a,b,c;

for( i=100; i<=999; i++ )

{

a = i/100;

b = ( i-a\*100 ) / 10;

c = i - a\*100 - b\*10;

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

cout << i <<endl;

}

}

11．求1000以内的所有完数。所谓完数，是指一个数恰好等于它的所有因子之和。例如，因为6=1+2+3，所以6为完数。

【解答】

#include<iostream>

using namespace std;

int main()

{

int i,j,s;

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

{

s = 0;

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

if ( i % j == 0 ) s = s + j;

if ( i == s ) cout << i << endl;

}

}

12．编写程序显示由符号组成的三角形图案。例如，程序运行后，

屏幕显示： How many lines ?

用户输入： 5

屏幕显示： What character ?

用户输入： \*

则输出如下图案：

\*

\* \* \*

\* \* \* \* \*

\* \* \* \* \* \* \*

\* \* \* \* \* \* \* \* \*

【解答】

#include<iostream>

using namespace std;

int main()

{

int i,j,k,n;

char ch;

cout<<"How many lines ?\n";

cin>>n;

cout<<"What character ?\n";

cin>>ch;

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

{

for( k=1; k<=n-i; k++ ) cout << " ";

for( j=1; j<=2\*i-1; j++ ) cout << ch ;

cout << endl;

}

}

13．已知*XYZ*+*YZZ*=532，其中*X*，*Y*和*Z*为数字，编写程序求出*X*，*Y*和*Z*的值。

【解答】

#include<iostream>

using namespace std;

int main()

{

int x,y,z,i;

for( x=1; x<=9; x++ )

for( y=1; y<=9; y++ )

for( z=0; z<=9; z++ )

{

i = 100\*x + 10\*y + z + 100\*y + 10\*z + z;

if ( i == 532 )

cout<<"x="<<x<<'\t'<<"y="<<y<<'\t'<<"z="<<z<<endl;

}

}

## 第3章 函数

#### 3.1 选择题

1．以下正确的函数原型为（ D ）。

（A）f1( int x; int y ); （B）void f1( x, y );

（C）void f1( int x, y ); （D）void f1( int, int );

2．有函数原型 void fun2( int );，在下列选项中，不正确的调用是（ C ）。

（A）int a = 21; fun2( a ); （B）int a = 15; fun2( a\*3 );

（C）int b = 100; fun2( &b ); （D）fun2( 256 );

3．有函数原型 void fun3( int \* );，在下列选项中，正确的调用是（ C ）。

（A）double x = 2.17; fun3( &x ); （B）int a = 15; fun3( a\*3.14 );

（C）int b = 100; fun3( &b ); （D）fun3( 256 );

4．有函数原型 void fun4( int & );，在下列选项中，正确的调用是（ C ）。

（A）int a = 2.17; fun4( &a ); （B）int a = 15; fun4( a\*3.14 );

（C）int b = 100; fun4( b ); （D）fun4( 256 );

5．有声明

void fun5( int \* & ); int a, int \*p = &a;

在下列选项中，正确的调用是（ B ）。

（A）fun5(&a); （B）fun5(p); （C）fun5(\*a); （D）fun5(\*p);

6．有声明

int fun6( int ); int (\*pf)(int) = fun6;

在下列选项中，正确的调用是（ C ）。

（A）int a=15; int n=fun6(&a); （B）int a = 15; cout<<(&pf)(a);

（C）cout<<(\*pf)( 256 ); （D）cout << \*pf( 256 );

7．在VC中，若定义一个函数的返回类型为void，则以下叙述正确的是（ C ）。

（A）函数返回值需要强类型转换 （B）函数不执行任何操作

（C）函数本身没有返回值 （D）函数不能修改实际参数的值

8．函数参数的默认值不允许为（ C ）。

（A）全局常量 （B）直接常量 （C）局部变量 （D）函数调用

9．使用重载函数编程序的目的是（ A ）。

（A）使用相同的函数名调用功能相似的函数 （B）共享程序代码

（C）提高程序的运行速度 （D）节省存储空间

10．在下列的描述中，（ B ）是错误的。

（A）使用全局变量可以从被调用函数中获取多个操作结果

（B）局部变量可以初始化，若不初始化，则系统默认它的值为0

（C）当函数调用完后，静态局部变量的值不会消失

（D）全局变量若不初始化，则系统默认它的值为0

11．在下列选项中，（ C ）具有文件作用域。

（A）语句标号 （B）局部变量 （C）全局变量 （D）静态变量

#### 3.2 阅读下列程序，写出运行结果

1．

#include<iostream>

using namespace std;

#include<cmath>

int f( int );

int main()

{

int i;

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

cout << f( i ) << endl;

}

int f( int a )

{

int b = 0, c = 1;

b ++; c++;

return int( a + pow( double(b), 2 ) + c );

}

【解答】

3

4

5

2．

#include<iostream>

using namespace std;

void func(int a, int b, int c = 3, int d = 4 );

int main()

{

func( 10, 15, 20, 30 );

func( 10, 11, 12 );

func( 12, 12 );

}

void func( int a, int b, int c, int d )

{

cout<<a<<'\t'<<b<<'\t'<<c<<'\t'<<d<< endl;

}

【解答】

10 15 20 30

10 11 12 4

12 12 3 4

3．

#include<iostream>

using namespace std;

void func( int, int, int \* );

int main()

{

int x, y, z;

func( 5, 6, &x );

func( 7, x, &y );

func( x, y, &z );

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

}

void func( int a, int b, int \*c )

{

b += a; \*c = b - a;

}

【解答】

6, 6, 6

4．

#include<iostream>

using namespace std;

void func( int, int, int & );

int main()

{

int x=0, y=1, z=2;

func( 1, 2, x );

func( x + y, y, y );

func( z, x + y, z );

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

}

void func( int a, int b, int &c )

{

b += a; c = b - a;

}

【解答】

2, 1, 3

5．

#include<iostream>

using namespace std;

void func( int \*, int \*, int \*& );

int main()

{

int a=10, b=20;

int \*p=&a, \*q=&b;

func( p, q, p );

cout << "\*p=" << \*p << ", \*q=" << \*q << endl;

}

void func( int \*t1, int \*t2, int \*& rt )

{

\*t1 += 5; \*t2 += 5;

rt = t1;

\*rt += 5;

cout << "\*t1=" << \*t1 << ", \*t2=" << \*t2<< ", \*rt=" << \*rt << endl;

}

【解答】

\*t1=20,\*t2=25,\*rt=20

\*p=20,\*q=25

6．

#include<iostream>

using namespace std;

int f2( int, int );

int f1( int a, int b )

{

int c;

a += a; b += b;

c = f2( a+b, b+1 );

return c;

}

int f2( int a, int b )

{

int c;

c = b % 2;

return a + c;

}

int main()

{

int a = 3, b = 4;

cout << f1( a, b ) << endl;

}

【解答】

15

7．

#include<iostream>

using namespace std;

int age( int n )

{

int f;

if( n == 1 )

f = 10;

else

f = age( n-1 ) + 2;

return f;

}

int main()

{

cout << "age : " << age( 5 ) << endl;

}

【解答】

age:18

8．

#include<iostream>

using namespace std;

int f1( int a, int b )

{

return a + b;

}

int f2( int a, int b )

{

return a - b;

}

int f3( int(\*t )( int, int ), int a, int b )

{

return (\*t )( a, b );

}

int main()

{

int (\*p )( int, int );

p = f1;

cout << f3( p, 4, 8 ) << endl;

p = f2;

cout << f3( p, 8, 4 ) << endl;

}

【解答】

12

4

9．

#include<iostream>

using namespace std;

int sub( int, int );

int a = 1;

int main()

{

int m = 1, n = 2, f;

f = sub( m, n );

cout << a << '\t' << f << endl;

f = sub( m, n );

cout << a << '\t' << f << endl;

}

int sub( int c, int d )

{

static int m = 2, n = 5;

cout << m << '\t' << n << '\t' << endl;

a = ++ a; c = m ++; d = n ++;

return c + d;

}

【解答】

2 5

2 7

3 6

3 9

#### 3.3 思考题

1．函数的作用是什么？如何定义函数？什么是函数原型？

【解答】

函数的两个重要作用：

（1）任务划分，把一个复杂任务划分为若干小任务，便于分工处理和验证程序正确性；（2）软件重用，把一些功能相同或相近的程序段，独立编写成函数，让应用程序随时调用，而不需要编写雷同的代码。

函数的定义形式：

类型 函数名 （ [ 形式参数表 ] ）

{

语句序列

}

函数原型是函数声明，告诉编译器函数的接口信息：函数名、返回数据类型、接收的参数个数、参数类型和参数顺序，编译器根据函数原型检查函数调用的正确性。

2．什么是函数值的返回类型？什么是函数的类型？如何通过指向函数的指针调用一个已经定义的函数？编写一个验证程序进行说明。

【解答】

（1）函数的返回类型是函数返回的表达式的值得类型；

（2）函数类型是指函数的接口，包括函数的参数定义和返回类型；

（3）若有

functionType functionName; //functionType是已经定义的函数类型

functionType \*functionPointer=functionName; //定义函数指针并获取函数地址

则可以通过函数指针调用函数：

(\*functionPointer)(argumentList);

或 functionPointer(argumentList)；

其中argumentList是实际参数表。

验证程序：

#include<iostream>

using namespace std;

int main()

{

typedef int myfunc(int,int);

myfunc f,\*fp;

int a=10,b=6;

fp=f;

cout<<"Using f(a):"<<f(a,b)<<endl; //函数名调用函数

cout<<"Using fp(a):"<<fp(a,b)<<endl; //函数指针调用函数

cout<<"Using (\*fp)(a):"<<(\*fp)(a,b)<<endl; //函数指针调用函数

return 0;

}

int f(int i,int j)

{

return i\*j;

}

3． 什么是形式参数？什么是实际参数？C++函数参数有什么不同的传递方式？编写一个验证程序进行说明。

【解答】

参数是调用函数与被调用函数之间交换数据的通道。函数定义首部的参数称为形式参数，调用函数时使用的参数称为实际参数。C++有三种参数传递机制：值传递（值调用）；指针传递（地址调用）；引用传递（引用调用）。

验证程序：

#include<iostream>

using namespace std;

void funcA(int i)

{ i=i+10; }

void funcB(int \*j)

{ \*j=\*j+20; }

void funcC(int &k)

{ k=k+30; }

int main()

{

int a=1;

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

funcB(&a);cout<<"a="<<a<<endl;

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

}

程序输出：

a=1 //传值参数，实际参数值不变

a=21 //指针参数，形式参数通过间址修改实际参数

a=51 //引用参数，形式参数通过别名方式修改实际参数

4．C++函数通过什么方式传递返回值？当一个函数返回指针类型时，对返回表达式有什么要求？当返回引用类型时，是否可以返回一个算术表达式？为什么？

【解答】

C++首先计算表达式的值，然后把该值赋给函数返回类型的匿名对象，通过这个对象，把数值带回调用点，继续执行后续代码。

当函数返回指针类型时，返回的地址值所指对象不能是局部变量。因为局部变量在函数运行结束后会被销毁，返回这个指针是毫无意义的。

返回引用的对象不能是局部变量，也不能返回表达式。算术表达式的值被储存在匿名空间中，函数运行结束后会被销毁，返回这个变量的引用也是无意义的。

5．变量的生存期和变量作用域有什么区别？请举例说明。

【解答】

变量的生存期是指程序运行后，变量占有内存的时间；变量作用域指的是指变量声明之后，在程序正文中有效的那部分区域。

例如，定义函数:

void count()

{

static int n=0;

//……

}

该函数中n被定义为static变量，生存期是整个程序运行时期；但作用域只在count函数中。

6．静态局部变量有什么特点？编写一个应用程序，说明静态局部变量的作用。

【解答】

静态局部变量的生存期是全程的，作用域是局部的。程序开始执行时就分配和初始化存储空间（默认初始化值为0）。定义静态局部变量的函数退出时，系统保持其存储空间和数值。下次调用这个函数时，static变量还是上次退出函数时的值。直至整个程序运行结束，系统才收回存储空间。

程序略。

7．在一个语句块中能否访问一个外层的同名局部变量？能否访问一个同名的全局变量？ 如果可以，应该如何访问？编写一个验证程序进行说明。

【解答】

一个语句块中不能访问外层的同名局部变量。可以访问一个同名的全局变量。

验证程序：

#include<iostream>

using namespace std;

int a=0; //全局变量a

int main()

{

int a=1; //外层局部变量a

{

int a=2; //内层局部变量a

cout<<"Local a is "<<a<<endl; //输出内层局部变量a

}

cout<<"Main a is "<<a<<endl; //输出外层局部变量a

cout<<"Global a is "<<::a<<endl; //输出全局部变量a

}

8．有函数原型

void f (int & n);

和函数调用

int a;

//…

f(a);

有人说，因为n是a的引用，则在函数f中访问n相当于访问a，所以，可以在f的函数体内直接使用变量名a。这种说法正确吗？为什么？编写一个验证程序。

【解答】

形式参数n的作用域是f函数，实际参数a的作用域是调用f函数的模块（例如main函数），所以在f函数中可见n而不可见a。因此，这种说法不正确。f函数内不能直接使用变量名a，只能通过别名n访问a。

验证程序：

#include<iostream>

using namespace std;

void f ( int&n );

int main()

{

int a = 1 ;

f( a );

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

}

void f ( int&n )

{

a++; //错误，直接使用a

n++; //正确

}

产生编译错误：error C2065: “a”: 未声明的标识符

9．有以下程序：

#include<iostream>

using namespace std;

int main()

{

char c;

cin>>c;

if(c=='y'||c=='Y')

int a=1;

else

int a=0;

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

}

编译错误为：error C2065: 'a' : undeclared identifier，指示语句行

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

变量a没有定义。请分析原因，并进行修改。

【解答】

if-else语句中分别出现变量a的说明语句，是作用域为if分支和else分支的两个变量。在if-else语句之后，这两个变量都是没有定义的，因此出现编译错误。正确的程序是：

#include<iostream>

using namespace std;

int main()

{

char c;

int a;

cin>>c;

if(c=='y'||c=='Y')

a=1;

else

a=0;

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

}

10．有函数原型

double function(int,double);

函数function的返回值类型是什么？函数的类型是什么？使用typedef定义函数的类型。

有函数调用语句

x=function(10,(2\* (0.314+5));

其中的括号“()”与函数原型中括号有什么语义区别？

【解答】

函数function的返回值类型是double

函数类型是： double (int,double)

可以定义为： typedef double funType (int,double);

函数调用function(10,(2\*(0.314+5)) 中，外层括号表示调用函数匹配的实际参数表，里面的两层括号是表达式运算。

11．请分析以下各语句的意义。

int \* fun();

int \* (\*pf)();

fun();

pf = fun;

pf();

【解答】

int \* fun() ; //函数原型声明。fun是返回int\*类型，没有参数的函数

int \* (\*pf)(); //声明指针变量。pf是指向函数的指针，函数类型为int\*()

fun() ; //调用函数语句

pf = fun ; //向指针变量赋值。函数指针pf指向函数fun

pf() ; //用指针变量间址调用函数

#### 3.4 编程题

1．已知 ，其中，sh为双曲正弦函数，即。编写一个程序，输入*x*的值，求*y*的值。

【解答】

#include<iostream>

#include<cmath>

using namespace std;

double sh( double t );

int main()

{

double x,y;

cout << "x=";

cin >> x;

y = sh( 1+sh(x) )/( sh( 2\*x )+sh( 3\*x ) );

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

}

double sh( double t )

{ return ( exp( t )-exp( -t ) )/2; }

2．输入*m*、*n*和*p*的值，求*s* = 的值。注意判断运算中的溢出。

【解答】

using namespace std;

double f( long k,long num );

int main()

{

long m,n,p; double s,f1,f2,f3;

cout << "m,n,p=";

cin>>m>>n>>p;

f1=f( 1,m );

f2=f( 3,n );

f3=f( 5,p );

if (f1 && f2 && f3 )

{

s = ( f1 + f2) /f3;

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

}

else

cout<<"溢出!\n";

}

double f( long k,long num )

{

long i;

double sum=0;

for( i=1; i<=num && sum<2147483647; i++ )

{

sum = sum + pow( double (i),double (k) );

}

if (i<=num)

return 0; //溢出时返回

return sum;

}

3．输入*a*、*b*和*c*的值，编写一个程序求这三个数的最大值和最小值。要求：把求最大值和最小值操作分别编写成一个函数，并使用指针或引用做为形式参数把结果返回main函数。

【解答】

（1）使用指针参数

#include<iostream>

using namespace std;

void fmaxmin( double,double ,double ,double \*,double \* );

int main()

{

double a,b,c,max,min;

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

cin >> a >> b >> c;

fmaxmin( a,b,c,&max,&min );

cout << "max=" << max << endl;

cout << "min=" <<min << endl;

}

void fmaxmin( double x,double y,double z,double \*p1,double \*p2 )

{

double u,v;

if ( x>y )

{ u = x; v = y; }

else

{ u = y; v = x; };

if ( z>u ) u = z;

if ( z<v ) v = z;

\*p1 = u;

\*p2 = v;

}

（2）使用引用参数

#include<iostream>

using namespace std;

void fmaxmin( double,double ,double ,double& ,double& );

int main()

{

double a,b,c,max,min;

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

cin >> a >> b >> c;

fmaxmin( a,b,c,max,min );

cout << "max=" << max << endl;

cout << "min=" << min << endl;

}

void fmaxmin( double x,double y,double z,double &p1,double &p2 )

{

double u,v;

if ( x>y )

{ u = x; v = y; }

else

{ u = y; v = x; };

if ( z>u ) u = z;

if ( z<v ) v = z;

p1 = u;

p2 = v;

}

4．用线性同余法生成随机数序列的公式为：

*rk* = ( multiplier × *rk*−1 + increment ) % modulus

序列中的每一个数*rk*都可以由它的前一个数*rk*−1计算出来。例如，如果有：

*rk* = ( 25 173 × *rk*−1 + 13 849 ) % 65 536

则可以产生65 536个各不相同的整型随机数。设计一个函数作随机数生成器，生成1位或2位数的随机数。

利用这个随机数生成器，编写一个小学生学习四则运算的练习程序，要求：

* 可以进行难度选择，一级难度只用1位数，二级难度用2位数；
* 可以选择运算类型，包括加、减、乘、除等；
* 可以给出错误提示；
* 可以统计成绩。

【解答】

#include<iostream>

using namespace std;

int Rand(int,int); //生成指定范围的随机数

int main()

{

int w,i,r,t = 0;

char op,answer;

int a,b,d;

while(1) //练习开始

{

cout<<"现在开始？( Y 或 N )\n" ;

cin>>answer;

if (answer=='N'||answer=='n') break;

while(1)

{

cout << "请输入难度( 1或2 ):";

cin >> w;

if ( w != 1 && w != 2 )

cout << "输入难度错误，重新输入！" << endl;

else

break ;

}

while(1)

{

cout << "请输入运算类型( +,-,\*,/ ):" ;

cin >> op;

if ( op != '+' && op != '-' && op != '\*' && op != '/' )

cout << "输入运算符错误，重新输入！" << endl;

else

break;

}

//出10道题，每题10分

t=0;

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

{

while(1)

{ if( w == 1 )

{

a = Rand(0,10); b = Rand(0,10);

}

else

if( w == 2 )

{

a = Rand(10,100); b = Rand(10,100);

}

if ( op == '-' )

if ( a<b ) continue ; //使被减数大于减数

if ( op == '/' )

if ( int( a/b ) != (a / b) ) continue; //只做结果为整数的除法

break;

}

cout << a << op << b << '=';

cin >> d;

switch ( op )

{

case '+': r = a + b; break;

case '-': r = a - b; break;

case '\*': r = a \* b; break;

case '/': r = a / b; break;

}

if ( r == d )

{

cout << "你算对了，加10分！" << endl;

t = t + 10;

}

else

cout << "你算错了！" << endl;

}

cout << "你的成绩是：" << t << "分" << endl;

}

}

int Rand(int m, int n)

{

static int r; //静态变量保留上一个随机数

do

{

r = ( 25173\*r + 13849 ) % 65536 ;

} while (r<m||r>=n);

return r;

}

5．已知勒让德多项式为：



编写程序，从键盘输入*x*和*n*的值，使用递归函数求*pn*(*x*)的值。

【解答】

#include<iostream>

using namespace std;

double p( double x,int n );

int main()

{

int n;

double x;

cout << "please input x and n:";

cin >> x >> n;

cout << "p(" << x << "," << n << ")=" << p( x,n ) <<endl;

}

double p( double x,int n )

{ double t1,t2;

if( n == 0 )

return 1;

else

if( n == 1 )

return x;

else

{

t1 = ( 2\*n-1 )\*p( x,n-1 );

t2 = ( n-1 )\*p( x,n-2 );

return ( t1-t2 )/n;

}

}

6．把以下程序中的print()函数改写为等价的递归函数。

#include <iostream>

using namespace std;

void print( int w )

{

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

{

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

cout << i << " ";

cout << endl;

}

}

int main()

{

print( 5 );

}

程序运行结果：

1

2 2

3 3 3

4 4 4 4

5 5 5 5 5

【解答】

#include<iostream>

using namespace std;

void print(int w)

{

int i;

if( w )

{ print( w-1 );

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

cout << w << " ";

cout << endl;

}

}

void main()

{

print( 5 );

}

7．已知用梯形法求积分的公式为：，其中，*h* = ( *b*−*a* ) / *n*，*n*为积分区间的等分数，编程求如下积分的值。要求：把求积分公式编写成一个函数，并使用函数指针作为形式参数。调用该函数时，给出不同的被积函数作为实际参数求不同的积分。

①  ②  ③ 

【解答】

#include<iostream>

#include<cmath>

using namespace std;

double f1( double x )

{

return 4 / ( 1 + x\*x );

}

double f2( double x )

{

return sqrt( 1 + x\*x );

}

double f3( double x )

{

return sin( x );

}

double trap( double( \*fun )( double x ), double a,double b,long n )

{

double t,h; int i;

t = ( ( \*fun )(a) + ( \*fun )( b ) ) / 2.0;

h = ( b - a ) / n;

for( i=1; i<=n-1; i++ ) t += ( \*fun )( a + i \* h );

t \*= h;

return t;

}

int main()

{

double t1,t2,t3;

t1 = trap( f1,0,1,10000 );

cout << "t1=" << t1 << endl;

t2 = trap( f2,1,2,10000 );

cout << "t2=" << t2 << endl;

t3 = trap( sin,0,3.14159265/2,10000 );

cout << "t3=" << t3 << endl;

}

8．编写一个程序，包含三个重载的display函数和一个主函数。要求：第一个函数输出double值，前面用字符串“a double:”引导；第二个函数输出一个int值，前面用字符串“a int:”引导；第三个函数输出一个char字符值，前面用字符串“a char:”引导；在主函数中，分别用double、int和char型变量作为实参调用display函数。

【解答】

#include<iostream>

using namespace std;

void display( double d )

{

cout << "a double:" << d << endl;

}

void display( int i )

{

cout << "a int:" << i << endl;

}

void display( char c )

{

cout << "a char:" << c << endl;

}

int main()

{

double d = 1.5; int i = 100; char c = 'a';

display( d );

display( i );

display( c );

}

9．使用重载函数编程序分别把两个数和三个数从大到小排列。

【解答】

#include<iostream>

using namespace std;

void sort( double x,double y );

void sort( double x,double y,double z );

int main()

{

sort( 5.6, 79 );

sort( 0.5, 30.8, 5.9 );

}

void sort(double x,double y)

{

if ( x>y )

cout << x << '\t' << y << endl;

else

cout << y << '\t' << x << endl;

}

void sort( double x,double y,double z )

{

double t;

if( y<z ) { t = y; y = z; z = t; }

if( x<z ) { t = x; x = z; z = t; }

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

cout << x << '\t' << y << '\t' << z << '\t' << endl;

}

10．给定求组合数公式为：，编一程序，输入*m*和*n*的值，求的值。注意优化算法，降低溢出可能。要求：主函数调用以下函数求组合数：

int Fabricate( int m, int n ); //返回的值

Fabricate函数内需调用Multi函数：

int Multi( int m, int n ); // 返回 

程序由4个文件组成。头文件用于存放函数原型，作为调用接口；其他3个cpp文件分别是main、Fabricate和Multi函数的定义。

【解答】

分析计算公式得到：

m!/(n!\*(m-n)!)=(m\*(m-1)\*…\*(m-n+1)\*(m-n)!)/ (n!\*(m-n)!)

=(m\*(m-1)\*…\*(m-n+1)/ (n!)

因此可以避免求较大数据m的阶乘，减少溢出的可能性。

//Fabricate.h

#ifndef FABRICATE\_H

#define FABRICATE\_H

int Fabricate( int m,int n );

int Multi( int m, int n );

#endif

//main.cpp

#include<iostream>

using namespace std;

#include "Fabricate.h"

int main()

{

int m ,n;

cout << "input m and n:";

cin >> m >> n;

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

}

//Fabricate.cpp

#include "Fabricate.h"

int Fabricate( int m, int n )

{

return Multi( m, m-n + 1 ) / Multi( n, 1 );

}

//Multi.cpp

int Multi( int m, int n )

{

int i, t = 1;

for( i=n; i<=m; i++ )

t = t \* i;

return t;

}

## 第4章 数组

#### 4.1 选择题

1．以下对一维数组 a 的定义正确的是（ C ）。

（A）int n = 5, a[n]; （B）int a(5);

（C）const int N = 5; int a[N]; （D）int n; cin>>n; int a[n];

2．下列数组定义语句中，不合法的是（ A ）。

（A）int a[3] = { 0, 1, 2, 3 }; （B）int a[] = { 0, 1, 2 };

（C）int a[3] = { 0, 1, 2 }; （D）int a[3] = { 0 };

3．已知 int a[10] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 }, \*p = a;，不能表示数组 a 中元素的式子是（ C ）。

（A）\*a （B）\*p （C）a （D）a[ p−a ]

4．已知 int a[] = { 0,2,4,6,8,10 }, \*p = a+1; 其值等于0的表达式是（ D ）。

（A）\* (p++) （B）\*(++p) （C）\*(p−−) （D）\*(−−p)

5．以下不能对二维数组a进行正确初始化的语句是（ C ）。

（A）int a[2][3] = { 0 };

（B）int a[][3] = { { 0,1 }, { 0 } };

（C）int a[2][3] = { { 0, 1 }, { 2, 3 }, { 4, 5 } };

（D）int a[][3] = { 0, 1, 2, 3, 4, 5 };

6．已知int a[][3] = { { 0, 1 }, { 2, 3, 4 }, { 5, 6 }, { 7 } }; 则 a[2][1]的值是（ C ）。

（A）0 （B）2 （C）6 （D）7

7．已知int a[3][3] = { 1, 2, 3, 4, 5, 6, 7, 8, 9 }; 不能表示数组元素a[2][1]的地址是（ B ）。

（A）&a[2][1] （B）\*(a[2]+1) （C）a[2]+1 （D）\*(a+2)+1

8．已知char \*a[]={ "fortran", " basic", "pascal", "java", "c++" };，则 cout<<a[3];的显示结果是（ C ）。

（A）t （B）一个地址值 （C）java （D）javac++

9．若用数组名作为调用函数的实参，则传递给形参的是（ A ）。

（A）数组存储首地址 （B）数组的第一个元素值

（C）数组中全部元素的值 （D）数组元素的个数

10．设有char \*s1="ABCDE", \*s2="ABCDE",\*s3=s1;，下列表达式中值等于true的是（ D ）。

（A）strcmp(s1,s2) （B）strcmp(s1,s3)==1 （C）strcmp(s2,s3)==-1（D）strcmp(s1,s3) ==0

11．设char \*s1, \*s2;分别指向两个字符串，可以判断字符串s1和s2是否相等的表达式为（ D ）。

（A）s1=s2 （B）strlen(s1,s2)

（C）strcpy(s1,s2)==0 （D）strcmp(s1,s2)==0

12．设char \*s1, \*s2;分别指向两个字符串，可以判断字符串s1是否大于字符串s2的表达式为（ C ）。

（A）strcmp(s1,s2)<0 （B）strcmp(s1,s2)==0

（C）strcmp(s1,s2)>0 （D）strcmp(s2,s1)>0

#### 4.2 阅读下列程序，写出运行结果

1．

#include <iostream>

using namespace std;

int main()

{

int i, count=0, sum=0;

double average;

int a[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

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

{

if( a[i] % 2 == 0 )

continue;

sum += a[ i ];

count ++;

}

average = sum/count;

cout << "count = " << count << '\t' << "average = " << average << endl;

}

【解答】

conut = 5 average = 5

2．

#include <iostream>

using namespace std;

int main()

{

int a[9] = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };

int \*p = a, sum = 0;

for(; p<a+9; p++ )

if(\*p % 2 == 0 )

sum += \*p;

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

}

【解答】

sum = 20

3．

#include <iostream>

#include <iomanip>

using namespace std;

const int N = 5;

int main()

{

int a[N][N]={ 0 }, i, j, k;

for( k=1, i=0; i<N; i++ )

for( j=i; j>= 0; j--, k++ )

a[j][i - j ] = k;

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

{

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

cout << setw( 3 ) << a[i][j];

cout << endl;

}

}

【解答】

1 3 6 10 15

2 5 9 14 0

4 8 13 0 0

7 12 0 0 0

11 0 0 0 0

4．

#include <iostream>

using namespace std;

int f(int [],int);

int main()

{

int a[] = { -1, 3, 5, -7, 9, -11 };

cout << f( a, 6 ) << endl;

}

int f( int a[], int size )

{

int i, t=1;

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

if( a[i]>0 ) \*= a[i];

return t;

}

【解答】

135

5．

#include <iostream>

using namespace std;

int f( int [][3], int, int );

int main()

{

int a[][3] = { 0, 1, 2, 3, 4, 5, 6, 7, 8 };

cout << f( a, 3, 3 ) << endl;

}

int f( int a[][3], int row, int col )

{

int i, j, t=1;

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

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

{

a[i][j]++;

if( i == j ) t \*= a[i][j];

}

return t;

}

【解答】

45

6．

#include <iostream>

using namespace std;

void test1( int \*a1 )

{

a1 = new int( 5 );

cout << "\*a1 = " << \*a1 << endl;

}

void test2(int \* & a2)

{

a2 = new int( 5 );

cout << "\*a2 = " << \*a2 << endl;

}

int main()

{

int \*p = new int( 1 );

test1( p );

cout << "test1: \*p1 = " << \*p << endl;

test2( p );

cout << "test2: \*p2 = " << \*p << endl;

}

【解答】

\*a1= 5

test1: \*p1= 1

\*a2= 5

test2: \*p2= 5

7．

#include <iostream>

using namespace std;

int main()

{

char s[] = "abccda";

int i;

char c;

for( i = 1; ( c=s[i] ) != '\0'; i ++ )

{

switch( c )

{

case 'a' : cout << '%'; continue;

case 'b' : cout << '$'; break;

case 'c' : cout << '\*'; break;

case 'd' : continue;

}

cout << '#' << endl;

}

}

【解答】

$#

\*#

\*#

%

8．

#include <iostream>

using namespace std;

int main()

{

char \*str[] = { "c++", "basic", "pascal" };

char \*\*p;

int i;

p = str;

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

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

}

【解答】

c++

basic

pascal

9．

#include <iostream>

using namespace std;

int main()

{

char s1[] = "Fortran", s2[] = "Foxpro";

char \*p, \*q;

p = s1; q = s2;

while(\*p && \*q )

{ if (\*p == \*q )

cout << \*p;

p ++;

q ++;

}

cout << endl;

}

【解答】

For

10．

#include <cstring>

#include <iostream>

using namespace std;

int main()

{

char str[][10] = { "vb", "pascal", "c++" }, s[10];

strcpy\_s( s, ( strcmp( str[0], str[1] ) < 0 ? str[0] : str[1] ) );

if( strcmp( str[2], s ) < 0 )

strcpy\_s( s, str[2] );

cout << s << endl;

}

【解答】

C++

#### 4.3 思考题

1．数组说明语句要向编译器提供什么信息？请写出一维数组、二维数组说明语句的形式。

【解答】

数组说明语句要向编译器提供数组名（标识符），数组元素的类型、数组维数、数组长度（元素的个数）等信息。

一维数组说明语句为： 类型 数组名[表达式]

二维数组说明语句为： 类型 数组名[表达式1] [表达式2]

2．数组名、数组元素的区别是什么？归纳一维数组元素地址、元素值不同的表示形式。若有说明

int aa [3], \*pa=aa;

请使用aa或pa，写出3个以上与aa[2] 等价的表达式。

【解答】

数组名是一个标识符，执行代码中代表数组的地址，即指向数组起始位置的指针；而数组元素是下标变量，性质相当于普通变量。

对一维数组aa第i个元素的地址可以表示为： &aa[i] aa+i;

对一维数组aa第i个元素的值可以表示为： a[i] \*(a+i);

与aa[2]等价的表达式：

\*(aa+2) \*(&a[2]) \*(pa+2) pa[2]

3．要把一维数组 int a[m\*n] 的元素传送到二维数组 int b[m][n]中，即在程序中要执行

b[i][j]=a[k];

请写出*k*→*i*,*j*的下标变换公式，并用程序进行验证。

【解答】

转换公式： i=k/n j=k%n

验证程序：

#include <iostream>

using namespace std;

int main()

{

const int M=3,N=4;

int k,a[M\*N]={1,2,3,4,5,6,7,8,9,10,11,12}, b[M][N];

int i,j;

cout<<"array a:"<<endl;

for( k=0; k<M\*N; k++ )

b[k/N][k%N] = a[k];

for( k=0; k<M\*N; k++ )

cout<<a[k]<<'\t';

cout<<endl;

cout<<"\*\*After convert\*\*"<<endl;

cout<<"array b:"<<endl;

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

{

for(j=0;j<N;j++) cout<<b[i][j]<<'\t';

cout<<endl;

}

}

4．有以下函数：

void query()

{

int \*p;

p=new int[3];

//…

delete []p;

p=new double[5];

//…

delete []p;

}

出现了编译错误。请分析错误的原因，并把上述程序补充完整，上机验证你的判断。

【解答】

在语句p=new double[5]; 中企图把动态浮点型数组的地址写入整型指针p，造成错误。错误为：

error C2440: “=”: 无法从“double \*”转换为“int \*”。

改正方法：增加一个double\*q指针。

void query()

{

int \*p;

p=new int[3];

delete [] p;

//……

double \*q;

q=new double[5];

//……

delete []q;

}

5．有以下程序，设计功能是调用函数create建立并初始化动态数组，令a[i]=i。但该程序运行后不能得到期望结果，请分析程序的错误原因并进行修改。

#include <iostream>

using namespace std;

void create(int \*, int);

int main()

{

int \*a = NULL, len;

cin>>len;

create(a,len);

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

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

cout << endl;

delete []a;

a = NULL;

}

void create(int \*ap, int n)

{

ap=new int[n];

for(int i=0; i<n; i++) ap[i]=i;

}

【解答】

函数create中，指针参数int\*ap是传地址值的参数。调用函数时接受实际参数a的值（地址值）作为初始值。ap仅是局部变量，ap=new int[n]获得新的地址值，函数执行完毕返回，ap被释放，完全与实际参数a无关。程序没有编译错误，但main不能获得动态数组。修改方法是把ap改为指针引用参数。

void create(int \*&,int); //函数原型声明,使用引用参数

void create(int \*&ap,int n) //函数定义

{

ap=new int[n];

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

ap[i]=i;

}

#### 4.4 编程题

1．已知求成绩的平均值和均方差公式：, ，其中，*n*为学生人数，*si*为第*i*个学生成绩。求某班学生的平均成绩和均方差。

【解答】

#include<iostream>

#include<cmath>

using namespace std;

void aveMsd( double [], int, double &, double & ); //求平均值和均方差值函数

int main()

{

double s[] = { 76, 85, 54, 77, 93, 83, 90, 67, 81, 65 };

double ave, msd;

int i,n;

n = sizeof( s )/sizeof( double ); //求数组元素的个数

cout<<"学生成绩：";

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

cout<<s[i]<<" ";

cout<<endl;

aveMsd( s, n, ave, msd );

cout << "平均值：" << ave << endl << "均方差值：" << msd << endl;

}

void aveMsd( double s[], int n, double &ave, double &msd )

{

int i;

double sum1=0, sum2=0;

for( i=0; i<n; i++ ) //求平均值

sum1 += s[i];

ave = sum1/n;

for( i=0; i<n; i++ ) //求均方差

sum2 += pow( s[i]-ave, 2 );

msd = sqrt( sum2/n );

}

2．用随机函数产生10个互不相同的两位整数存放到一维数组中，并输出其中的素数。

【解答】

#include<iostream>

#include<cmath>

#include <cstdlib>

#include<ctime>

using namespace std;

int main()

{

int a[10],i,j;

srand( int( time(0)) ); //为随机数生成器设置种子值

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

{

l: a[i] = rand(); //产生随机数存放到数组中

if ( a[i]<10 || a[i]>=100 ) //获取指定范围数据

goto l;

for( j=0; j<i; j++ ) //排除相同数据

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

goto l;

}

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

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

cout << endl;

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

{

double m=sqrt( double (a[i]) );

for( j=2; j<=m; j++)

if( a[i] % j == 0 )break;

if( j>m )

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

}

cout << "是素数！" << endl;

}

3．将一组数据从大到小排列后输出，要求显示每个元素及它们在原数组中的下标。

【解答】

#include<iostream>

using namespace std;

int main()

{

int a[] = { 38, 6, 29, 1, 25, 20, 6, 32, 78, 10 };

int index[10]; //记录下标的数组

int i,j,temp;

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

index[i] = i;

for( i=0; i<=8; i++ )

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

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

{

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

temp = index[i]; index[i] = index[j]; index[j] = temp;

}

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

cout << a[i] << '\t' << index[i] << endl;

}

4．从键盘输入一个正整数，判别它是否为回文数。所谓回文数，是指正读和反读都一样的数。例如，123321是回文数。

【解答】

#include<iostream>

using namespace std;

int main()

{

int b[10], i, j, k, flag ;

long num, n ;

cout << "num=" ; cin >> num;

k = 0;

n = num;

do //拆分整数，把各数字放入数组b

{

b[k++] = n % 10;

n = n/10;

} while( n != 0);

flag=1; //判断标志

i=0; j=k-1; //设置指示下标的指针

while(i<j)

if( b[i++] != b[j--] ) //对称位置元素不相等

{

flag = 0;

break ;

}

if( flag ) cout << num << "是回文数！" << endl;

else cout << num << "不是回文数！" << endl;

}

5．把两个升序排列的整型数组合并为一个升序数组。设计好算法，以得到较高的运行效率。

【解答】

#include<iostream>

using namespace std;

void merge(const int a[],int na, const int b[],int nb, int c[],int nc);

int main()

{

int a[4] = { 1, 2, 5, 7 };

int b[8] = { 3, 4, 8, 8, 9, 10, 11, 12 };

int c[12];

int i;

merge( a,4,b,8,c,12 );

for( i=0; i<12; i++ )

cout << c[i] << " ";

cout << endl;

}

void merge(const int a[],int na, const int b[],int nb, int c[],int nc)

{

int i,j,k;

i = j = k = 0;

while( i<na && j<nb )

{

if( a[i] > b[j] ) //当a[i]>b[j]，把b[i]写入数组c

{ c[k] = b[j]; k++; j++; }

else //当a[i]<=b[j]，把a[i]写入数组c

{ c[k] = a[i]; k++; i++; }

}

while( i<na )

{

c[k] = a[i]; i++; k++; //把数组a的剩余元素写入数组c

}

while( j<nb )

{

c[k] = b[j]; k++; j++; //把数组b的剩余元素写入数组c

}

}

6. 输入一个表示星期几的数，然后输出相应的英文单词。要求：使用指针数组实现。

【解答】

#include<iostream>

using namespace std;

int main()

{

char \*weekday[7]

= { "sunday", "monday", "tuesday","wednesday", "thursday", "friday", "saturday" };

int d;

cout << "please input week day: ";

cin >> d;

if( d>=0 && d<=6 )

cout << d << "---" << \*( weekday + d ) <<endl;

else

cout << "input error!" << endl;

}

7．编写以下函数：

（1）在一个二维数组中形成以下形式的*n*阶矩阵：



（2）去掉靠边的元素，生成新的*n*−2阶矩阵；

（3）求矩阵主对角线下元素之和；

（4）以方阵形式输出数组。

在main函数中调用以上函数进行测试。

【解答】

#include<iostream>

using namespace std;

void create( int \*&app, int n );

void del( int \*&app, int \*&bpp, int n );

int maindiagonal( int \*&app, int n );

void output( int \*&app, int );

int main()

{

int \*ap = NULL, \*bp = NULL, n;

cout << "输入矩阵的阶：";

cin >> n;

create( ap,n );

cout << "\n形成矩阵：\n";

output( ap, n );

cout << "去掉靠边元素生成的矩阵：\n";

del( ap,bp,n );

output( bp,n-2 );

cout << "主对角线元素之和：" << maindiagonal( ap, n ) <<endl;

}

//形成n阶矩阵函数

void create( int \* &app, int n )

{

app = new int[ n\*n ];

int i,j,k = 0;

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

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

{

if( i<=j )

app[k] = 1;

else

app[k] = i-j+1;

k++;

}

}

//去掉靠边元素生成n-2阶矩阵函数

void del( int \*&app, int \*&bpp, int n )

{

int i,j,k = 0;

bpp = new int[ ( n-2 )\*( n-2 ) ];

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

{

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

if ( i && j && i<n-1 && j <n-1 )

{

bpp[k]= \*( app + i\*n + j );

k++;

}

}

}

//求主对角线元素之和函数

int maindiagonal( int \*&app, int n )

{

int i,j,k = 0,sum = 0;

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

{

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

if( i==j )

sum += \*( app + i\*n + j);

}

return sum;

}

//以方阵的形式输出数组函数

void output( int \*&app, int n )

{

int i,j;

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

{

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

cout << \*( app + i\*n + j) << '\t';

cout<<endl;

}

cout<<endl;

}

8．设某城市3个百货公司某个季度销售电视机的情况和价格如下所示。编写程序，将表数据用数组存放，求各百货公司的电视机营业额。

|  |  |
| --- | --- |
| 牌号 | 价格 |
| 康佳 | 3500 |
| TCL | 3300 |
| 长虹 | 3800 |

|  |  |  |  |
| --- | --- | --- | --- |
| 牌号  公司 | 康佳 | TCL | 长虹 |
| 第一百货公司 | 300 | 250 | 150 |
| 第二百货公司 | 200 | 240 | 200 |
| 第三百货公司 | 280 | 210 | 180 |

【解答】

#include<iostream>

using namespace std;

int main()

{

long s[][3] = { { 300, 250, 150 }, { 200, 240, 200},{ 280, 210, 180 } };

long p[] = { 3500, 3300, 3800 };

int i,j;

double sum;

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

{

sum=0;

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

sum += s[i][j] \* p[j];

cout << "第" << i+1 << "百货公司的电视机营业额: " << sum << endl;

}

}

9．设计函数求一整型数组的最小元素及其下标。在主函数中定义和初始化该整型数组，调用该函数，并显示最小元素值和下标值。

【解答】

#include<iostream>

using namespace std;

int fmin(int [], int);

int main()

{

int a[ ] = { 73, 85, 62, 95, 77, 56, 81, 66, 90, 80 };

int index;

index = fmin( a, sizeof(a)/sizeof(int) );

cout << "The minnum number is : " << a[index] << endl;

cout << "The index is : " << index << endl;

}

int fmin( int a[], int size )

{

int i,min = a[0], index = 0;

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

if( a[i]<min )

{

min = a[i];

index = i;

};

return index;

}

10．假设有从小到大排列的一组数据存放在一个数组中，在主函数中从键盘输入一个在该数组的最小值和最大值之间的数，并调用一个函数把输入的数插入到原有的数组中，保持从小到大的顺序，并把最大数挤出。然后在主函数中输出改变后的数组。

【解答】

#include<iostream>

using namespace std;

void insert( int a[],int,int );

int main()

{

int a[] = { 10, 12, 23, 25, 48, 48, 53, 58, 60, 78 };

int x,n,i;

cout << "please input insert data: ";

cin >> x;

n = sizeof(a)/sizeof(int); //求数组长度

insert( a, n, x ); //插入元素

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

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

cout << endl;

}

void insert( int a[],int n,int x )

{

int i,p,j;

if ( x<a[n-1] )

{

for( i=1; i<n; i++ ) //查找插入位置

if( x<a[i] )

{

p=i; break;

}

for( j=n-1; j>=p; j-- ) //后移元素，挤出最大值

a[j] = a[j-1];

a[p] = x; //插入元素

}

}

11．一个整型数组的每个元素占4字节。编写一个压缩函数pack，把一个无符号小整数（0~255）数组进行压缩存储，只存放低8位；再编写一个解压函数unpack，把压缩数组舒展开来，以整数形式存放。主函数用随机函数生成数据初始化数组，测试pack和unpack函数。

【解答】

#include<iostream>

#include <cstdlib>

#include<ctime>

using namespace std;

void pack( int \*a, unsigned char \*p, int n );

void unpack( unsigned char \*p, int \*a, int n );

int main()

{

int \*ary, n, i;

unsigned char \*packary;

cout<<"请输入数组长度：";

cin>>n;

ary = new int [n]; //建立动态数组

packary = new unsigned char[n]; //压缩数组

srand(int(time(0)));

for(i=0;i<n;i++) //产生随机数并存放到动态数组中

ary[i]=rand()/255;

pack( ary, packary, n );

cout<<"\n输出压缩数组：";

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

{

if (i %10 == 0) cout<<endl; //控制一行输出10个数据

cout << int( packary[i] ) <<" ";

}

unpack( packary, ary, n);

cout<<"\n输出解压数组：";

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

{

if (i %10 == 0) cout<<endl;

cout<<ary[i]<<" ";

}

}

void pack( int \*a, unsigned char \*p, int n )

{

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

{

p[i] = unsigned char(a[i]);

}

}

void unpack( unsigned char \*p, int \*a, int n )

{

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

{

a[i] = int(p[i]);

}

}

12．编写程序，按照指定长度生成动态数组，用随机数对数组元素进行赋值，然后逆置该数组元素。例如，数组A[5]的初值为{6, 3, 7, 8, 2}，逆置后的值为{2, 8, 7, 3, 6}。要求：输出逆置前、后的数组元素序列。

【解答】

#include<iostream>

#include <cstdlib>

#include<ctime>

using namespace std;

void printarray(int \*p,int n);

void adverse(int \*p,int n);

int main()

{

int \*p,n,i;

cout<<"请输入数组长度:";

cin>>n;

p=new int [n]; //建立动态数组

srand(int(time(0)));

for(i=0;i<n;i++) //产生随机数并存放到动态数组中

\*(p+i)=rand()%1000;

cout<<"动态数组：";

printarray(p,n); // 输出动态数组

adverse(p,n); // 对数组逆置

cout<<"逆置数组：";

printarray(p,n); // 输出逆置数组

}

// 输出数组函数

void printarray(int \*p,int n)

{

int i;

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

{

if (i % 5==0) cout<<endl; // 控制一行输出5个数据

cout<<"ary["<<i<<"]="<<\*(p+i)<<"\t";

}

cout<<endl;

}

// 对数组逆置函数

void adverse(int \*p,int n)

{

int i,t;

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

{

t=\*(p+i);

\*(p+i)=\*(p+n-i-1);

\*(p+n-i-1)=t;

}

}

13．把一个字符串插到另一个字符串中指定的位置。

【解答】

#include<iostream>

using namespace std;

int main()

{

int p, i, j, k;

char s1[40], s2[40], s3[80];

cout << "s1=";

cin >> s1;

cout << "s2=";

cin >> s2;

cout << "input insert position:";

cin >> p; //输入插入位置

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

s3[i] = s1[i];

for( j=0; s2[j] != '\0'; j++ )

s3[i+j] = s2[j];

for( k=p; s1[k] != '\0'; k++ )

s3[j+k] = s1[k];

s3[j+k] = '\0';

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

}

14．把某班学生的姓名和学号分别存放到两个数组中，从键盘输入某位学生的学号，查找该学生是否在该班，若找到该学生，则显示出相应的姓名。

【解答】

#include<iostream>

using namespace std;

int main()

{

char name[5][20] = { "li ming", "zhang qing", "liu xiao ping", "wang ying", "lu pei" };

long num[5] = { 20030001, 20030002, 20030005, 20030007, 20030010 };

int i;

long snumber;

cout << "please input studentnumber:";

cin >> snumber;

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

{

if( num[i] == snumber )

{

cout << "Found! The name is :" << name[i] << endl;

break;

}

}

if( i==5 ) cout << "Can\'t found!" << endl;

}

15．将一组C++关键字存放到一个二维数组中，并找出这些关键字中的最小者。

【解答】

#include<iostream>

#include <cstring>

using namespace std;

int main()

{

char string[10];

char str[][10]={ "while", "break", "if", "extern", "void", "auto", "long", "static", "do", "const" };

int i;

strcpy( string, str[0] );

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

if( strcmp(str[i],string)<0 ) strcpy( string, str[i] );

cout << "The minimum string is:" << string << endl;

}

16．使用指针函数编写程序，把两个字符串连接起来。

【解答】

#include<iostream>

using namespace std;

char \*strcat( char \*str1,char \*str2 )

{

char \*p = str1;

while( \*p != '\0' ) p++;

\*p = \*str2;

do

{

p++;

str2++;

\*p = \*str2;

} while( \*str2 != '\0' );

return( str1 );

}

int main()

{

char str1[80], str2[80];

cout << "input str1:";

cin >> str1;

cout << "input str2:";

cin >> str2;

cout << "str1+str2=" << strcat( str1, str2 ) << endl;

}

17．使用string类，编写一个简单的文本编辑程序，能够实现基本的插入、删除、查找、替换等功能。

【解答】

略。

## 第5章 集合与结构

#### 5.1 选择题

1．语句cout<<(1&2)<<", "<<(1&&2)<<endl; 的输出结果是（ B ）。

（A）0, 0 （B）0, 1 （C）1, 0 （C）1, 1

2. 语句cout<<(1|2)<<", "<<(1||2)<<endl; 的输出结果是（ D ）。

（A）0, 0 （B）1, 1 （C）2, 0 （D）3, 1

3. 有以下说明语句：

struct point

{ int x; int y; }p;

则正确的赋值语句是（ C ）。

（A）point.x = 1; point.y = 2; （B）point={ 1, 2 };

（C）p.x = 1; p.y = 2; （D）p = { 1, 2 };

4．已知有职工情况结构变量emp定义为：

struct Date

{

int year;

int month;

int day;

};

strnct Employee

{

char name[20];

long code;

Date birth

};

Employee emp;

下列对emp的birth正确赋值方法是（ D ）。

（A）year=1980; month=5; day=1;

（B）birth.year=1980; birth.month=5; birth.day=1;

（C）emp.year=1980; emp.month=5; emp.day=1;

（D）emp.birth.year=1980; emp.birth.month=5; emp.birth.day=1;

5．有以下说明语句：

struct Student

{

int num;

double score;

};

Student stu[3]={{1001,80}, {1002,75}, {1003,91}}, \*p=stu;

则下面引用形式错误的是（ B ）。

（A）p−>num （B）(p++).num （C）(p++)−>num （D）(\*p).num

6．有以下说明语句：

struct Worker

{

int no;

char name[20];

};

Worker w, \*p ＝ &w;

则下列错误的引用是（ D ）。

（A）w.no （B）p->no （C）(\*p).no （D）\*p.no

7．s1和s2是两个结构类型变量，若要使赋值s1=s2合法，则它们的说明应该是（ C ）。

（A）s1只能接收相同类型的数据成员 （B）结构中的成员相同

（C）同一结构类型的变量 （D）存储字节长度一样的变量

#### 5.2 阅读下列程序，写出运行结果。

1．

#include <iostream>

using namespace std;

struct Data

{

int n;

double score;

};

int main()

{

Data a[3] = { 1001,87,1002,72,1003,90 }, \*p = a;

cout << (p++)->n << endl;

cout << (p++)->n << endl;

cout << p->n++ << endl;

cout << (\*p).n++ << endl;

}

【解答】

1001

1002

1003

1004

2．

#include <iostream>

using namespace std;

struct Employee

{

char name[ 20 ];

char sex;

};

void fun( Employee \*p )

{

if( (\*p).sex == 'm' )

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

}

int main()

{

Employee emp[5] = { "Liming", 'm', "Wangxiaoping", 'f', "Luwei", 'm' };

int i;

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

fun( emp+i );

}

【解答】

Liming

Luwei

3．

#include <iostream>

using namespace std;

struct Node

{

char \* s;

Node \* q;

};

int main()

{

Node a[ ] = { { "Mary", a+1 }, { "Jack", a+2 }, { "Jim", a } };

Node \*p = a;

cout << p->s << endl;

cout << p->q->s << endl;

cout << p->q->q->s << endl;

cout << p->q->q->q->s << endl;

}

【解答】

Mary

Jack

Jim

Mary

#### 5.3 思考题

1．判断一个整数n的奇偶性，可以利用位运算判断吗？请你试一试。

【解答】

可以。一个整数当最低位为1时，它是奇数，否则为偶数。以下函数返回对参数k的奇偶判断。

bool odd( int k )

{

return 1&k;

}

2．长度为N的数组可以表示N个元素的集合，若有：

S[i]==1，表示对应元素在集合中

如何实现集合的基本运算？请你试一试。并从内存和处理要求上与5.2.2节中集合的实现方法进行比较。

【解答】

长度为N的数组S可以表示有N个元素的集合。当S[i]==1，表示元素i+1在集合中；当S[i]==0，表示元素i+1不在集合中。集合运算通过对数组元素操作完成。

用数组实现集合运算的空间和时间消耗高于用无符号整数和位运算实现集合运算。

用数组实现集合运算程序如下。

#include<iostream>

using namespace std;

void setPut( unsigned \*S ); //输入集合S的元素

void setDisplay( const unsigned \*S ); //输出集合S中的全部元素

bool putX( unsigned \*S, unsigned x ); //元素x并入集合

void Com( unsigned \*C, const unsigned \*A, const unsigned \*B); //求并集C=A∪B

void setInt( unsigned \*C, const unsigned A, const unsigned B); //求交集C=A∩B

void setDif( const unsigned A, const unsigned B ); //求差集C=A-B

bool Inc( const unsigned \*A, const unsigned \*B ); //判蕴含

bool In( const unsigned \*S, const unsigned x ); //判属于x∈S

bool Null( const unsigned \*S ); //判空集

const int N=32;

//输入集合元素

void setPut(unsigned \*S)

{

unsigned x;

cin >> x;

while( x>0&&x<=N)

{

putX( S, x ); //把输入元素并入集合S

cin >> x;

}

}

//输出集合S中的全部元素

void setDisplay( const unsigned \*S )

{

cout << "{ ";

if (Null(S))

cout<<" }\n";

else

{

for( int i=0; i<N; i++ ) //输出元素

{

if( S[i] )

cout << i+1 << ", ";

}

cout << "\b\b }\n"; //擦除最后的逗号

}

return;

}

//元素x并入集合S

bool putX( unsigned \*S, unsigned x )

{

if(x>0&&x<=N)

{

S[x-1] = 1;

return true;

}

return false;

}

//求并集C=A∪B

void Com( unsigned \*C, const unsigned \*A, const unsigned \*B )

{

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

C[i]= int( A[i] || B[i] ) ;

}

//求交集C=A∩B

void setInt( unsigned \*C, const unsigned \*A, const unsigned \*B )

{

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

C[i]= int( A[i]&&B[i] ) ;

}

//求差集C=A-B

void setDif( unsigned \*C, const unsigned \*A, const unsigned \*B )

{

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

C[i]= int( A[i]&&!(A[1]&&B[i]) ) ;

}

//判蕴含，A蕴含于B时返回true

bool Inc( const unsigned \*A, const unsigned \*B )

{

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

{

if(A[i]&&!B[i])

return false;

}

return true;

}

//判属于，x∈S时返回true

bool In( const unsigned \*S, const unsigned x )

{

return S[x-1];

}

//判空集，S为空集时返回true

bool Null( const unsigned \*S )

{

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

{

if( S[i])

return false;

}

return true;

}

int main()

{

unsigned A[N]={0}, B[N]={0}, C[N]={0};

unsigned x;

cout << "Input the elements of set A, 1-"<<N<<", until input 0 :\n";

setPut( A );

cout << "Input the elements of set B, 1-"<<N<<", until input 0 :\n";

setPut( B );

cout<<"A = ";

setDisplay( A );

cout<<"B = ";

setDisplay( B );

cout << "Input x: ";

cin>>x;

cout << "Put " << x << " in A = ";

putX( A, x) ;

setDisplay(A);

cout << "C = A+B = ";

Com( C, A, B );

setDisplay( C );

cout << "C = A\*B = ";

setInt( C, A, B );

setDisplay( C );

cout << "C = A-B = ";

setDif( C, A, B );

setDisplay( C );

if( Inc( A, B ) )

cout << "A <= B is true\n";

else

cout << "not A <= B\n";

cout << "Input x: ";

cin >> x;

if( In( A, x ) )

cout << x << " in A\n";

else

cout << x << " not in A\n";

}

3．分析以下说明结构的语句：

struct Node

{

int data;

Node error; //错误

Node \* ok; //正确

};

error和ok分别属于什么数据类型？有什么存储要求？error出错的原因是什么？

【解答】

error是Node结构类型数据成员，错误。原因是结构定义的数据成员若为本身的结构类型，是一种无穷递归。ok是指向Node类型的指针，定义正确，占4字节。

4．本章例5-8中用辅助数组对结构数组进行关键字排序，有定义：

person \*index[100];

index数组存放结构数组元素的地址。如果把index定义改为：

int index[100];

用于存放结构数组元素的下标，可以实现对结构数组的索引排序吗？如何修改程序？请你试一试。

【解答】

可以。关键是通过整型索引数组元素作为下标访问结构数组。表示为：

all[pi[i]].name all[pi[i]].id all[pi[i]].salary

有关程序如下：

#include<iostream>

using namespace std;

struct person //说明结构类型

{

char name[10];

unsigned int id;

double salary;

} ;

void Input( person[], const int );

void Sort( person[], int[],const int );

void Output( const person[], int[],const int );

int main()

{

person allone[100] ; //说明结构数组

int index[100]; //说明索引数组

int total ;

for(int i=0; i<100; i++) //索引数组元素值初始化为结构数组元素下标

index[i]=i ;

cout<<"输入职工人数：";

cin>>total;

cout<<"输入职工信息：\n";

Input(allone,total);

cout<<"以工资做关键字排序\n";

Sort(allone,index, total);

cout<<"输出排序后信息：\n";

Output(allone,index,total);

}

void Input( person all[], const int n )

{

int i ;

for( i=0; i<n; i++ ) // 输入数据

{

cout<<i<<": 姓名: ";

cin>>all[i].name;

cout<<"编号: ";

cin >> all[i].id;

cout<<"工资: ";

cin >> all[i].salary ;

}

}

void Sort(person all[], int pi[], const int n)

{

int i,j;

int t; //交换用中间变量

for(i=1; i<n; i++) //以成员salary做关键字排序

{

for(j=0; j<=n-1-i; j++)

if(all[pi[j]].salary>all[pi[j+1]].salary) //通过索引数组访问结构数组元素

{

t=pi[j]; //交换索引数组元素值

pi[j]=pi[j+1];

pi[j+1]= t;

}

}

}

void Output(const person all[], int pi[], const int n)

{

for( int i=0; i<n; i++ ) // 输出排序后数据

cout<<all[pi[i]].name<<'\t'<<all[pi[i]].id<<'\t'<<all[pi[i]].salary<<endl;

}

5．有以下结构说明和遍历单向链表的函数。函数内有错误吗？是什么性质的错误？请上机验证你的分析。

struct Node

{

int data;

Node \* next;

};

void ShowList( Node \*head )

{

while( head )

{

cout << head->date << '\n';

head ++;

}

}

【解答】

head++错误，原因是动态链表的结点存放不是连续顺序的内存空间，它们是逐个结点通过new建立的，所以不能用++做地址偏移运算。应该用：

head=head->next;

#### 5.4 编程题

1．编写程序，将一个整型变量右移4位，并以二进制形式输出该整数在移位前和移位后的数值。观察系统填补空缺的数位情况。

#include<iostream>

using namespace std;

void bitDisplay(unsigned value);

int main()

{

unsigned x;

cout << "Enter an unsigned integer: ";

cin >> x;

bitDisplay(x);

x>>=4;

cout<<"Right 4-bit\n";

bitDisplay(x);

}

void bitDisplay(unsigned value)

{

unsigned c;

unsigned bitmask = 1<<31;

cout << value << " = \t";

for( c=1; c<=32; c++ )

{

cout << ( value&bitmask ? '1' : '0' );

value <<= 1;

if( c%8 == 0 )

cout << ' ';

}

cout << endl;

}

2．整数左移一位相当于将该数乘以2。编写一个函数

unsigned power2( unsigned number, unsigned pow );

使用移位运算计算number\*2pow，并以整数形式输出计算结果。注意考虑数据的溢出。

【解答】

unsigned power2( unsigned number, unsigned pow )

{

unsigned c=1;

unsigned bitmask = 1<<31;

while(c<31) //溢出判断

{

if( number&bitmask ) break; //查找最高位的1

c++;

bitmask>>=1;

}

if(pow<c)

return number<<pow;

else

{

cout<<"overflow!\n";

return 0;

}

}

3．使用按位异或（^）运算，可以不使用中间变量，快速交换两个变量的值。设计一个函数，实现快速交换两个整型变量的值。

【解答】

void Swap( int &A, int &B)

{

A=A^B;

B=A^B;

A=A^B;

}

4. 集合的元素通常是字符。设计程序，用无符号整数表示ASCII码字符集合，用位运算实现各种基本集合运算。

【解答】

ASCII码是0~127的整数，可以用长度为4的无符号整型数组表示集合，如教材例5-4所示。区别是，在输入集合元素时，需要把字符转换成整型数据，在输出操作中，把整型集合元素转换成字符型数据。

程序略。

5．使用结构类型表示复数。设计程序输入两个复数，可以选择进行复数的＋、－、×或÷运算，并输出结果。

【解答】

#include <iostream>

#include <iomanip>

using namespace std;

struct complex

{

double re, im;

};

int main()

{

complex a,b,c; char oper;

cout << "输入复数a的实部和虚部: ";

cin >> a.re >> a.im;

cout << "输入复数b的实部和虚部:";

cin >> b.re >> b.im;

cout << "输入运算符: ";

cin >> oper;

switch ( oper )

{

case '+': c.re=a.re+b.re; c.im=a.im+b.im; break;

case '-': c.re=a.re-b.re; c.im=a.im-b.im; break;

case '\*': c.re=a.re\*b.re-a.im\*b.im;

c.im=a.im\*b.re+a.re\*b.im; break;

case '/': c.re=(a.re\*b.re+a.im\*b.im)/(b.re\*b.re+b.im\*b.im);

c.im=(a.im\*b.re-a.re\*b.im)/(b.re\*b.re+b.im\*b.im);

break;

default: cout << "input error!" << endl;

return 0;

}

cout << "c=" << c.re;

cout << setiosflags( ios::showpos );

cout << c.im << "i" << endl;

return 0;

}

6．把一个班的学生姓名和成绩存放到一个结构数组中，寻找并输出最高分者。

【解答】

#include <iostream>

using namespace std;

int main()

{

struct data

{

char name[12];

double score;

}a[ ] = {"李小平",90,"何文章",66,"刘大安",87,"汪立新",93,"罗建国",78,

"陆丰收",81,"杨勇",85,"吴一兵",55,"伍晓笑",68,"张虹虹",93};

double max = a[0].score;

int i,n = sizeof(a) / sizeof(data);

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

if( a[i].score > max ) max = a[i].score;

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

if( a[i].score == max ) cout << a[i].name << endl;

}

7．使用结构表示*X*—*Y*平面直角坐标系上的点，编写程序顺序读入一个四边形的4个顶点坐标，判别由这个顶点的连线构成的图形是否为正方形、矩形或其他四边形。要求：定义求两个点距离的函数使用结构参数。

【解答】

#include <iostream>

#include <cmath>

using namespace std;

struct point

{

double x;

double y;

};

double d( point p1, point p2 )

{

return sqrt( pow( p1.x-p2.x,2 )+pow( p1.y-p2.y,2 ) );

}

int main()

{

int i; point p[5];

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

{ cout << "输入第" << i << "个顶点的横坐标和纵坐标: ";

cin >> p[i].x >> p[i].y;

}

if( fabs( d( p[1],p[2] ) - d( p[3],p[4] ))<=1e-8

&& fabs( d( p[1],p[4] ) - d( p[2],p[3] ))<=1e-8

&& fabs( d( p[1],p[3] ) - d( p[2],p[4] ))<=1e-8)

if( fabs( d( p[1],p[2] ) - d( p[2],p[3] ))<1e-8 )

cout << "四个顶点构成的图形为正方形！" << endl;

else cout << "四个顶点构成的图形为矩形！" << endl;

else cout << "四个顶点构成的图形为其它四边形！" << endl;

}

8．建立一个结点包括职工的编号、年龄和性别的单向链表，分别定义函数完成以下功能：

（1）遍历该链表输出全部职工信息；

（2）分别统计男、女职工的人数；

（3）在链表尾部插入新职工结点；

（4）删除指定编号的职工结点；

（5）删除年龄在60岁以上的男性职工或55岁以上的女性职工结点，并保存在另一个链表中。

要求：用主函数建立简单菜单选择，并测试程序。

【解答】

#include <iostream>

using namespace std;

struct employee

{

int num;

int age;

char sex;

employee \*next;

};

employee \*head, \*head1;

//建立单向链表

employee \*create()

{

employee \*head, \*p, \*pend;

char ch;

head = NULL;

cout << "\t输入数据?(y/n)"; cin >> ch;

if( ch == 'y' )

{

p = new employee;

cout << "\t编号:"; cin >> p->num;

cout << "\t年龄:"; cin >> p->age;

cout << "\t性别:"; cin >> p->sex;

}

else

goto L0;

while( ch == 'y' )

{

if( head == NULL ) head = p;

else pend->next = p;

pend = p;

cout << "\t输入数据?(y/n)"; cin>>ch;

if( ch == 'y' )

{

p = new employee;

cout << "\t编号:"; cin >> p->num;

cout << "\t年龄:"; cin >> p->age;

cout << "\t性别:"; cin >> p->sex;

}

}

pend->next = NULL;

L0: return head;

}

//显示单向链表中全部职工信息

void show( employee \*head )

{

employee \*p = head;

if( !head ) { cout << "\t空链表！" << endl; goto L1; }

cout << "\t链表中的数据是: \n";

while( p )

{

cout << '\t' << p->num << "," << p->age << "," << p->sex << endl;

p = p->next;

}

L1:

}

//统计男女职工人数

void count( employee \*head )

{

employee \*p = head;

int m, f;

m = 0; f = 0;

while( p )

{

if( p->sex == 'm' )

m++;

else

f++;

p = p->next;

}

cout << "\t男职工人数：" << m << endl;

cout << "\t女职工人数：" << f << endl;

}

//在链表尾部插入新结点

employee \*insert()

{

employee \*pend = head, \*p;

//在空链表尾部插入新结点

if( !head )

{

p = new employee;

cout << "\t编号:"; cin >> p->num;

cout << "\t年龄:"; cin >> p->age;

cout << "\t性别:"; cin >> p->sex;

head = p;

p->next = NULL;

return head;

}

//在链表尾部插入新结点

while( pend->next != NULL )

{

pend = pend->next;

}

p = new employee;

cout << "\t编号:"; cin >> p->num;

cout << "\t年龄:"; cin >> p->age;

cout << "\t性别:"; cin >> p->sex;

pend->next = p;

pend = p;

pend->next = NULL;

return head;

}

//删除指定编号的结点

employee \*del( int bh )

{

employee \*p, \*q;

if ( !head )

{

cout << "\t空链表！" << endl;

goto L2;

}

//删除链首结点

if( head->num == bh )

{

p = head;

head = head->next;

delete p;

cout << "\t结点已被删除！" << endl;

goto L2;

}

//删除非链首结点

q = head;

while( q->next != NULL )

{

if ( q->next->num == bh )

{

p = q->next; //待删除结点

q->next = p->next;

delete p;

cout << "\t结点已被删除！" << endl;

goto L2;

}

q = q->next;

}

cout << "\t找不到需删除结点！" << endl;

L2: return ( head );

}

//删除指定年龄段的结点，并把被删除结点保存在另一链表中

employee \*delcreate()

{

employee \*p, \*pd, \*p1, \*q;

int flag;

//建立新链表

if ( head == NULL )

{

cout << "\t空链表！" << endl;

goto L3;

}

head1 = NULL;

pd = new employee;

p = head;

flag = 0;

while ( p != NULL )

{

if( p->age >= 55 && p->age <=60 )

{

pd->num = p->num;

pd->age = p->age;

pd->sex = p->sex;

if( head1 == NULL )

head1 = pd;

else

p1->next = pd;

p1 = pd;

pd = new employee;

flag = 1;

}

p = p->next;

}

if ( flag == 0 )

{ cout << "\t没有需删除的结点！" << endl; goto L3; }

p1->next = NULL;

//显示新链表

cout <<"\t新链表中的数据是: \n";

p = head1;

while( p )

{

cout << '\t' << p->num << "," << p->age << "," << p->sex << endl;

p = p->next;

}

//删除指定年龄的结点

p = head;

q = p;

while ( p != NULL )

{

if( p->age >= 55 && p->age <= 60)

if( head->age == p->age )

{

pd = head; //待删除结点

head = head->next;

delete pd;

p = head;

continue;

}

else

if( p->next == NULL )

{

pd = p; //待删除结点

q->next = NULL;

delete pd;

goto L3;

}

else

{

pd = p; //待删除结点

q->next = p->next;

delete pd;

p = q->next;

continue;

}

q = p;

p = p->next;

}

L3: return ( head );

}

int main()

{

int choice, bh ;

L:

cout << "\n\t\t请键入操作选择\n" << endl;

cout << "\t 1 --- 建立单向链表" << endl;

cout << "\t 2 --- 显示单向链表中全部职工信息" << endl;

cout << "\t 3 --- 统计男女职工人数" << endl;

cout << "\t 4 --- 在职工尾部插入新结点" << endl;

cout << "\t 5 --- 删除指定编号的结点" << endl;

cout << "\t 6 --- 删除指定年龄的结点，并把被删除结点保存在另一链表中" << endl;

cout << "\t 0 --- 退出" << endl ;

cout << "\t\t";

cin >> choice ;

switch ( choice )

{

case 1 : head = create() ; goto L ;

case 2 : show( head ); goto L ;

case 3 : count( head ); goto L;

case 4 : head = insert(); goto L;

case 5 : cout << "\t输入需删除结点编号:";

cin >> bh;

head = del( bh ); goto L;

case 6 : head = delcreate(); goto L;

case 0 : cout << " \t退出程序的运行！\n" << endl ; break ;

default : cout << "\t输入错误，请重新输入！\n" << endl ; goto L;

}

}

9．输入一行字符，按输入字符的反序建立一个字符结点的单向链表，并输出该链表中的字符。

【解答】

#include <iostream>

using namespace std;

struct node

{

char ch;

node \*next;

};

void show( node \*head );

int main()

{

node \*head, \*p;

char c;

head = NULL;

while( (c = getchar()) != '\n' ) //输入一行字符

{

p = new node; //建立新结点

p->ch = c;

p->next = head; //插入表头

head=p;

}

show(head);

}

void show( node \*head ) //输出链表

{

node \*p = head;

cout << "链表中的字符是: \n";

while( p )

{ cout << p->ch;

p = p->next;

}

cout << endl;

}

10．设有说明语句：

struct List { int data; List \* next; };

List \*head;

head是有序单向链表的头指针。请编写函数：

void Count( List \* head );

计算并输出链表数据相同值的结点及个数。例如，若数据序列为：

2 3 3 3 4 5 5 6 6 6 6 7 8 9 9

则输出结果为：

data number

3 3

5 2

6 4

9 2

请用本章例5-11的程序生成有序链表，测试Count函数。

【解答】 略

11．用带头结点的有序单向链表可以存放集合，如图5.16所示。头结点不存放集合元素，仅为操作方便而设置。使用这种数据结构，设计集合的输入、输出和各种基本运算的函数。

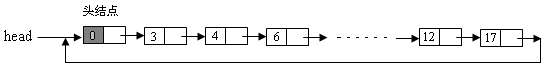


图5.16 带头结点的有序单向链表

【解答】 略

## 第6章 类与对象

#### 6.1 选择题

1．若有以下说明，则在类外使用对象objX成员的正确语句是（ C ）。

class X

{

int a;

void fun1();

public:

void fun2();

};

X objX;

（A）objX.a=0; （B）objX.fun1(); （C）objX.fun2(); （D）X::fun1();

2．若有以下说明，则对n的正确访问语句是（ B ）。

class Y

{

//… ;

public:

static int n;

};

int Y::n;

Y objY;

（A）n=1; （B）Y::n=1; （C）objY::n=1; （D）Y−>n

3．若有以下类Z说明，则函数fStatic中访问数据a错误的是（ C ）。

class Z

{

static int a;

public:

static void fStatic(Z&);

};

int Z::a=0;

Z objZ;

（A）void Z::fStatic() { objZ.a =1; }

（B）void Z::fStatic() { a = 1; }

（C）void Z::fStatic() { this−>a = 0; }

（D）void Z::fStatic() { Z::a = 0; }

4．若有以下类W说明，则函数fConst的正确定义是（ A ）。

class W

{

int a;

public:

void fConst(int&) const;

};

（A）void W::fConst( int &k )const { k = a; }

（B）void W::fConst( int &k )const { k = a++; }

（C）void W::fConst( int &k )const { cin >> a; }

（D）void W::fConst( int &k )const { a = k; }

5．若有以下类T说明，则函数fFriend的错误定义是（ C ）。

class T

{

int i;

friend void fFriend( T&, int );

};

（A）void fFriend( T &objT, int k ) { objT.i = k; }

（B）void fFriend( T &objT, int k ) { k = objT.i; }

（C）void T::fFriend( T &objT, int k ) { k += objT.i; }

（D）void fFriend( T &objT, int k ) { objT.i += k; }

6．在类定义的外部，可以被访问的成员有（ C ）。

（A）所有类成员 （B）private或protected的类成员

（C）public的类成员 （D）public或private的类成员

7．关于this指针的说法正确的是（ C ）。

（A）this指针必须显式说明 （B）定义一个类后，this指针就指向该类

（C）成员函数拥有this指针 （D）静态成员函数拥有this指针

8．说明一个类的对象时，系统自动调用（ B ）；撤销对象时，系统自动调用（ C ）。

（A）成员函数 （B）构造函数 （C）析构函数 （D）友元函数

9．下面对构造函数的不正确描述是（ C ）。

（A）用户定义的构造函数不是必须的 （B）构造函数可以重载

（C）构造函数可以有参数，也可以有返回值 （D）构造函数可以设置默认参数

10．下面对析构函数的正确描述是（ C ）。

（A）系统在任何情况下都能正确析构对象 （B）用户必须定义类的析构函数

（C）析构函数没有参数，也没有返回值 （D）析构函数可以设置默认参数

11．关于类的静态成员的不正确描述是（ D ）。

（A）静态成员不属于对象，是类的共享成员

（B）静态数据成员要在类外定义和初始化

（C）静态成员函数不拥有this指针，需要通过类参数访问对象成员

（D）只有静态成员函数可以操作静态数据成员

12．在下列选项中，（ C ）不是类的成员函数。

（A）构造函数 （B）析构函数 （C）友元函数 （D）复制构造函数

13．下面对友元的错误描述是（ D ）。

（A）关键字friend用于声明友元

（B）一个类中的成员函数可以是另一个类的友元

（C）友元函数访问对象的成员不受访问特性影响

（D）友元函数通过this指针访问对象成员

14．若class B中定义了一个class A的类成员A a，则关于类成员的正确描述是（ C ）。

（A）在类B的成员函数中可以访问A类的私有数据成员

（B）在类B的成员函数中可以访问A类的保护数据成员

（C）类B的构造函数可以调用类A的构造函数做数据成员初始化

（D）类A的构造函数可以调用类B的构造函数做数据成员初始化

#### 6.2 阅读下列程序，写出运行结果

1．

#include<iostream>

using namespace std;

class A

{

public :

int f1();

int f2();

void setx( int m ) { x = m; cout << x << endl; }

void sety( int n ) { y = n; cout << y << endl; }

int getx() { return x; }

int gety() { return y; }

private :

int x, y;

};

int A::f1()

{ return x + y; }

int A::f2()

{ return x - y; }

int main()

{

A a;

a.setx( 10 ); a.sety( 5 );

cout << a.getx() << '\t' << a.gety() << endl;

cout << a.f1() << '\t' << a.f2() << endl;

}

【解答】

10

5

10 5

15 5

2．

#include<iostream>

using namespace std;

class T

{

public :

T( int x, int y )

{

a = x; b = y;

cout << "调用构造函数1." << endl;

cout << a << '\t' << b << endl;

}

T( T &d )

{

cout << "调用构造函数2." << endl;

cout << d.a << '\t' << d.b << endl;

}

~T() { cout << "调用析构函数."<<endl; }

int add( int x, int y = 10 ) { return x + y; }

private :

int a, b;

};

int main()

{

T d1( 4, 8 );

T d2( d1 );

cout << d2.add( 10 ) << endl;

}

【解答】

调用构造函数1.

4 8

调用构造函数2.

4 8

20

调用析构函数.

调用析构函数.

3．

#include<iostream>

using namespace std;

class T

{

public:

T(int x) { a=x; b+=x; };

static void display(T c) { cout<<"a="<<c.a<<'\t'<<"b="<<c.b<<endl; }

private:

int a;

static int b;

};

int T::b=5;

int main()

{

T A(3), B(5);

T::display(A);

T::display(B);

}

【解答】

a=3 b=13

a=5 b=13

4．

#include<iostream>

using namespace std;

#include<cmath>

class Point

{

public :

Point( float x, float y )

{ a = x; b = y; cout<<"点( "<<a<<", "<<b<<" )"; }

friend double d( Point &A, Point &B )

{ return sqrt((A.a-B.a)\*(A.a-B.a)+(A.b-B.b)\*(A.b-B.b)); }

private:

double a, b;

};

int main()

{

Point p1( 2, 3 );

cout << " 到";

Point p2( 4, 5 );

cout << "的距离是：" << d( p1,p2 ) << endl;

}

【解答】

点（3，4）到点（4，5）的距离是：2.82843

5．

#include<iostream>

using namespace std;

class A

{

public :

A() { a = 5; }

void printa() { cout << "A:a = " << a << endl; }

private :

int a;

friend class B;

};

class B

{

public:

void display1( A t )

{ t.a ++; cout << "display1:a = " << t.a << endl; };

void display2( A t )

{ t.a --; cout << "display2:a = " << t.a << endl; };

};

int main()

{

A obj1;

B obj2;

obj1.printa();

obj2.display1( obj1 );

obj2.display2( obj1 );

obj1.printa();

}

【解答】

A:a = 5

display1:a = 6

display2:a = 4

A:a = 5

6．

#include<iostream>

using namespace std;

class A

{

public:

A(int x):a(x=0){ }

void getA(int &A) { A=a; }

void printA() { cout<<"a="<<a<<endl; }

private:

int a;

};

class B

{

public:

B(int x, int y):aa(x=0) { b = y; }

void getAB(int &A, int &outB) { aa.getA(A); outB=b; }

void printAB() { aa.printA(); cout<<"b="<<b<<endl; }

private:

A aa;

int b;

};

int main()

{

A objA;

int m=5;

objA.getA(m);

cout<<"objA.a="<<m<<endl;

cout<<"objB:\n";

B objB;

objB.getAB(12,56);

objB.printAB();

}

【解答】

objA:a=5

objB:

a=12

b=56

#### 6.3 思考题

1．结构与类有什么区别？如果把程序中定义结构的关键字struct直接改成class，会有什么问题？用教材中的一个例程试试看，想一想做什么修改能使程序正确运行？

【解答】

结构是数据的封装，类是数据和操作的封装。可以把结构看成是类的特例。结构和类都可以用关键字struct或class定义。区别是，struct定义的结构或类的全部成员都是公有的，用class定义的结构或类不做声明的成员是私有的。

若把struct改成class，只需要把全部成员定义为public就可以了。

2．有说明：

class A

{

int a;

double x;

public:

funMember();

};

A a1, a2, a3;

编译器为对象a1、a2和a3开辟了什么内存空间？它们有各自的funMember函数的副本吗？C++通过什么机制调用类的成员函数？

【解答】

开辟的存储空间有a1.a, a1.x, a2.a, a2.x, a3.a, a3.x。各对象没有funMember函数的副本，C++通过this指针调用成员函数。

3．C++提供了系统版本的构造函数，为什么还需要用户自定义构造函数？编写一个验证程序，说明自定义构造函数的必要性。

【解答】

类的默认构造函数可以建立基本类型数据成员的存储空间。基于以下两个原因，需要用户定义构造函数：

（1）对数据成员的值做指定初始化；

（2）类的数据是由指针管理的堆。

程序略。

4．试从定义方式、访问方式、存储性质和作用域4个方面来分析类的一般数据成员和静态数据成员的区别，并编写一个简单程序验证它。

【解答】

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 定义方式 | 访问方式 | 存储性质 | 作用域 |
| 一般数据成员 | 类中定义 | 对象.数据成员 | 局部数据 | 由访问属性public, protected, private决定 |
| 静态数据成员 | 类中声明，类外定义 | 对象.数据成员  类::数据成员 | 全局数据 |

程序略。

5．试从定义方式、调用方式两个方面来分析常成员函数、静态成员函数和友元函数的区别。考察例6-15，若class Goods的指针域：

Goods \* next;

被声明为私有（private）成员，程序会出现什么错误？做什么最小修改能使程序正确运行？

【解答】

|  |  |  |
| --- | --- | --- |
|  | **定义方式** | **调用方式** |
| 常成员函数 | 函数原型以const做后缀  this指针被约束为指向常量的常指针 | 与一般成员函数调用形式相同  对数据成员只读 |
| 静态成员函数 | 以static做函数原型前缀  没有this指针 | 通过类或对象调用  用于操作静态数据成员 |
| 友员函数 | 以friend做函数原型前缀  没有this指针 | 通过参数访问对象  可以访问对象的不同属性的成员 |

在例6-15中，若把next声明为私有数据成员，只须把有关指针操作的函数定义为友员函数就可以了：

friend void purchase( Goods \* &f, Goods \*& r, int w );

friend void sale( Goods \* & f , Goods \* & r );

6．设有：

class M

{

public: int a;

};

class N

{

public:

M m;

int b;

void fun()

{ /\*…\*/ }

};

int main()

{

N n;

N \*p = &n;

/\*…\*/

}

描述在N::fun中如何访问M类的数据成员a？在main函数中又如何访问对象n的全部数据成员？

【解答】

在N::fun中访问M类的数据成员a的形式是： m.a

在main函数中访问M类的数据成员的形式是： n.b，n.m.a

#### 6.4 编程题

1．定义一个Book（图书）类，在该类定义中包括以下数据成员和成员函数。

数据成员： bookname（书名）、price（价格）和number（存书数量）。

成员函数： display()显示图书的情况；borrow()将存书数量减1，并显示当前存书数量；restore()将存书数量加1，并显示当前存书数量。

在main函数中，要求创建某一种图书对象，并对该图书进行简单的显示、借阅和归还管理。

【解答】

#include <iostream>

using namespace std;

class Book

{

public:

void setBook(char\*,double,int);

void borrow();

void restore();

void display();

private:

char bookname[40];

double price;

int number;

};

//在类外定义Book类的成员函数

void Book::setBook(char \*name, double pri, int num)

{

strcpy(bookname, name);

price=pri;

number=num;

}

void Book::borrow()

{

if (number==0 )

{

cout << "已没存书，退出！" << endl;

abort();

}

number = number - 1;

cout << "借一次，现存书量为：" << number << endl;

}

void Book::restore()

{

number = number + 1;

cout << "还一次，现存书量为：" << number << endl;

}

void Book::display()

{

cout << "存书情况：" << endl

<< "bookname:" << bookname << endl

<< "price:" << price << endl

<< "number:" << number << endl;

}

int main()

{

char flag, ch;

Book computer;

computer.setBook( "c++程序设计基础" , 32, 1000 );

computer.display();

ch = 'y';

while ( ch == 'y' )

{

cout << "请输入借阅或归还标志(b/r)：";

cin >> flag;

switch ( flag )

{

case 'b': computer.borrow(); break;

case 'r': computer.restore();

}

cout << "是否继续？(y/n)";

cin >> ch;

}

computer.display();

}

2．定义一个Box（盒子）类，在该类定义中包括以下数据成员和成员函数。

数据成员：length（长）、width（宽）和height（高）。

成员函数：构造函数Box，设置盒子的长、宽和高3个初始数据；成员函数setBox对数据成员置值；成员函数volume 计算盒子的体积。

在main函数中，要求创建Box对象，输入长、宽、高，输出盒子的体积。

【解答】

#include <iostream>

using namespace std;

class BOX

{

public:

BOX( double l, double w, double h )

{ length = l;

width = w;

height = h;

}

void volume()

{ cout << "volume=" << length \* width \* height << endl;

}

private:

double length, width, height;

};

int main()

{

BOX box1( 1,3,5 );

box1.volume();

BOX box2( 2,4,6 );

box2.volume();

}

3．定义一个Student类，在该类定义中包括：一个数据成员（分数score）及两个静态数据成员（总分total和学生人数count）；成员函数scoretotalcount(double s) 用于设置分数、求总分和累计学生人数；静态成员函数sum()用于返回总分；静态成员函数average()用于求平均值。

在main函数中，输入某班同学的成绩，并调用上述函数求全班学生的总分和平均分。

【解答】

#include <iostream>

using namespace std;

class student

{

public:

void scoretotalcount( double s )

{

score = s;

total = total + score;

count++;

}

static double sum()

{

return total;

}

static double average()

{

return total / count;

}

private:

double score;

static double total;

static double count;

};

double student::total=0;

double student::count=0;

int main()

{

int i,n; double s;

cout << "请输入学生人数：";

cin >> n;

student stu;

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

{

cout << "请输入第" << i << "个学生的分数：";

cin >> s;

stu.scoretotalcount( s );}

cout << "总分：" << student::sum() << endl;

cout << "平均分：" << student::average() << endl;

}

4．定义一个表示点的结构类型Point和一个由直线方程*y* = *ax* + *b*确定的直线类Line。结构类型Point有两个成员x和y，分别表示点的横坐标和纵坐标。Line类有两个数据成员a和b，分别表示直线方程中的系数。Line类有一个成员函数print用于显示直线方程。友元函数setPoint(Line &l1,Line &l2)用于求两条直线的交点。在main函数中，建立两个直线对象，分别调用print函数显示两条直线的方程，并调用函数setPoint求这两条直线的交点。

【解答】

#include <iostream>

using namespace std;

struct point

{

double x; double y;

};

class line

{

public:

line( double u, double v )

{

a=u; b=v;

}

void print()

{

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

}

friend point setpoint(line &l1,line &l2);

private:

double a, b;

};

point setpoint( line &l1, line &l2 )

{

point p;

p.x=( l2.b-l1.b )/( l1.a-l2.a );

p.y=( l1.a\*l2.b-l2.a\*l1.b)/(l1.a-l2.a );

return p;

}

int main()

{

point setp;

line l1(2,3), l2(4,5);

cout<<"直线l1: ";

l1.print();

cout<<"直线l2: ";

l2.print();

setp=setpoint( l1,l2 );

cout<<"直线l1和直线l2的交点:("<<setp.x<<","<<setp.y<<")"<<endl;

}

5．用类成员结构修改习题6.4第4小题的程序，使其实现相同的功能。定义Point类和Line类，表示点和线；定义setPoint类，包含两个Line类成员和一个表示直线交点的Point成员，并定义类中求直线交点的成员函数。编写每个类相应的成员函数和测试用的主函数。

【解答】

略。

## 第7章 运算符重载

#### 7.1 选择题

1．在下列运算符中，不能重载的是（ B ）。

（A） ! （B）sizeof （C）new （D）delete

2．在下列关于运算符重载的描述中，（ D ）是正确的。

（A）可以改变参与运算的操作数个数 （B）可以改变运算符原来的优先级

（C）可以改变运算符原来的结合性 （D）不能改变原运算符的语义

3．在下列函数中，不能重载运算符的函数是（ B ）。

（A）成员函数 （B）构造函数 （C）普通函数 （D）友元函数

4．要求用成员函数重载的运算符是（ A ）。

（A）= （B）== （C）<= （D）++

5．要求用友元函数重载的ostream类输出运算符是（ ）。

（A）= （B）[] （C）<< （D）()

6．在下列关于类型转换的描述中，错误的是（ A ）。

（A）任何形式的构造函数都可以实现数据类型转换。

（B）带非默认参数的构造函数可以把基本类型数据转换成类类型对象。

（C）类型转换函数可以把类类型对象转换为其他指定类型对象。

（D）类型转换函数只能定义为一个类的成员函数，不能定义为类的友元函数。

#### 7.2 阅读下列程序，写出运行结果

1．

#include <iostream>

using namespace std;

class T

{

public :

T()

{ a = 0; b = 0; c = 0; }

T( int i, int j, int k )

{ a = i; b =j; c = k; }

void get( int &i, int &j, int &k )

{ i = a; j = b; k = c; }

T operator\* ( T obj );

private:

int a, b, c;

};

T T::operator\* ( T obj )

{

T tempobj;

tempobj.a = a \* obj.a;

tempobj.b = b \* obj.b;

tempobj.c = c \* obj.c;

return tempobj;

}

int main()

{

T obj1( 1,2,3 ), obj2( 5,5,5 ), obj3;

int a, b, c;

obj3 = obj1 \* obj2;

obj3.get( a, b, c );

cout<<"( obj1\*obj2 ): " <<"a = "<<a<<'\t'<<"b = "<<b<<'\t'<<"c = "<<c<<'\n';

(obj2\*obj3).get( a, b, c );

cout<<"( obj2\*obj3 ): " <<"a = "<<a<<'\t'<<"b = "<<b<<'\t'<<"c = "<<c<<'\n';

}

【解答】

( obj1 \* obj2 ): a = 5 b = 10 c = 15

( obj2 \* obj3 ): a = 25 b = 50 c = 75

2．

#include <iostream>

using namespace std;

class Vector

{

public:

Vector(){ }

Vector(int i,int j)

{ x = i; y = j; }

friend Vector operator+ ( Vector v1, Vector v2 )

{

Vector tempVector;

tempVector.x = v1.x + v2.x;

tempVector.y = v1.y + v2.y;

return tempVector;

}

void display()

{ cout << "( " << x << ", " << y << ") "<< endl; }

private:

int x, y;

};

int main()

{

Vector v1( 1, 2 ), v2( 3, 4 ), v3;

cout << "v1 = ";

v1.display();

cout << "v2 = ";

v2.display();

v3 = v1 + v2;

cout << "v3 = v1 + v2 = ";

v3.display();

}

【解答】

v1 = ( 1, 2 )

v2 = ( 3, 4 )

v3 = v1 + v2 = ( 4, 6 )

#### 7.3 思考题

1．一个运算符重载函数被定义为成员函数或友元函数后，在定义方式、解释方式和调用方式上有何区别？可能会出现什么问题？请用一个实例说明之。

【解答】

以二元运算符为例。

|  |  |  |  |
| --- | --- | --- | --- |
| **运算符重载** | **定义** | **解释** | **调用** |
| 成员函数 | Obj& operator op();  Obj operator *op*(object); | Obj.operator *op*()  ObjL.operator op(ObjR) | Obj *op* 或 *op* Obj  ObjL *op* ObjR  左操作数通过this指针指定，右操作数由参数传递 |
| 友员函数 | friend Obj & operator *op*(Obj &);  friend Obj operator *op*(Obj,Obj); | operator *op*(Obj)  operator *op*(ObjL,ObjR) | Obj *op* 或  *op* Obj  ObjL *op* ObjR  操作数均由参数传递 |

可能会出现的问题：

（1）运算符的左右操作数不同，须用友员函数重载；

（2）当运算符的操作需要修改类对象状态时，应用成员函数重载。

（3）友员函数不能重载运算符 = () [] ->

必须要用友员函数重载的运算符 >> <<

程序略。

2．类类型对象之间、类类型和基本类型对象之间用什么函数进行类型转换？归纳进行类型转换的构造函数和类型转换函数的定义形式、调用形式和调用时机。

【解答】

构造函数可以把基本类型、类类型数据转换成类类型数据；类类型转换函数可以在类类型和基本数据类型之间做数据转换。

|  |  |  |  |
| --- | --- | --- | --- |
|  | 定义形式 | 调用形式 | 调用时机 |
| 构造函数 | ClassX::ClassX(arg,arg1=E1,...,argn=En); | 隐式调用 | 建立对象、参数传递时 |
| 类类型转换函数 | ClassX::operator Type(); | 用类型符显式调用；  自动类型转换时隐式调用 | 需要做数据类型转换时 |

#### 7.4 编程题

1．分别使用成员函数和友元函数编写程序重载运算符“+”，使该运算符能实现两个字符串的连接。

【解答】

（1） 使用成员函数

#include <iostream>

#include<cstring>

using namespace std;

class s

{

public:

s(){ \*str = '\0'; }

s( char \*pstr ) { strcpy( str,pstr );

char \*gets() { return str; }

s operator+( s obj );

private:

char str[10];

};

s s::operator+( s obj )

{

strcat( str,obj.str );

return str; //或return \*this

}

int main()

{

s obj1( "Visual" ),obj2( " C++" ),obj3;

obj3 = obj1 + obj2;

cout << obj3.gets() << endl;

}

（2）使用友员函数

#include <iostream>

#include<cstring>

using namespace std;

class s

{ public:

s(){ \*str= '\0'; }

s( char \*pstr ) { strcpy( str,pstr ); }

char \*gets() { return str; }

friend s operator+( s obj1,s obj2 );

private:

char str[100];

};

s operator+( s obj1,s obj2 )

{

s tempobj;

strcat( tempobj.str,obj1.str );

strcat( tempobj.str,obj2.str );

return tempobj;

}

int main()

{

s obj1( "Visual" ),obj2( " C++" ),obj3;

obj3 = obj1 + obj2;

cout << obj3.gets() << endl;

}

2．定义一个整数计算类Integer，实现短整数 +、−、\*、/ 基本算术运算。要求：可以进行数据范围检查（−32 768～32 767，或自行设定），数据溢出时显示错误信息并中断程序运行。

【解答】

#include <iostream>

using namespace std;

class Integer

{

private:

short a;

public:

Integer (short n=0){ a=n;}

Integer operator +(Integer);

Integer operator -(Integer);

Integer operator \*(Integer);

Integer operator /(Integer);

Integer operator =(Integer);

void display()

{ cout<<a<<endl; }

};

Integer Integer::operator+(Integer x)

{

Integer temp;

if(a+x.a<-32768||a+x.a>32767)

{ cout<<"Data overflow!"<<endl; abort(); }

temp.a=a+x.a;

return temp;

}

Integer Integer::operator-(Integer x)

{

Integer temp;

if(a-x.a<-32768||a-x.a>32767)

{ cout<<"Data overflow!"<<endl; abort(); }

temp.a=a-x.a;

return temp;

}

Integer Integer::operator\*(Integer x)

{

Integer temp;

if(a\*x.a<-32768||a\*x.a>32767)

{cout<<"Data overflow!"<<endl; abort();}

temp.a=a\*x.a;

return temp;

}

Integer Integer::operator/(Integer x)

{

Integer temp;

if(a/x.a<-32768||a/x.a>32767)

{ cout<<"Data overflow!"<<endl; abort(); }

temp.a=a/x.a;

return temp;

}

Integer Integer::operator=(Integer x)

{

a=x.a;

return \*this;

}

int main()

{

Integer A(90),B(30),C;

cout<<"A=";A.display();

cout<<"B=";B.display();

C=A+B;

cout<<"C=A+B="; C.display();

C=A-B;

cout<<"C=A-B="; C.display();

C=A\*B;

cout<<"C=A\*B="; C.display();

C=A/B;

cout<<"C=A/B="; C.display();

}

3．定义一个实数计算类Real，实现单精度浮点数+、−、\*、/ 基本算术运算。要求：可以进行数据范围（−3.4×1038～3.4×1038，或自行设定）检查，数据溢出时显示错误信息并中断程序运行。

【解答】

#include <iostream>

using namespace std;

class Real

{

private:

double a;

public:

Real (double r=0){a=r;}

Real operator +(Real);

Real operator -(Real);

Real operator \*(Real);

Real operator /(Real);

Real operator =(Real);

void display()

{ cout<<a<<endl; }

};

Real Real::operator+(Real x)

{

Real temp;

if(a+x.a<-1.7e308||a+x.a>1.7e308)

{ cout<<"Data overflow!"<<endl; abort(); }

temp.a=a+x.a;

return temp;

}

Real Real::operator-(Real x)

{

Real temp;

if(a-x.a<-1.7e308||a-x.a>1.7e308)

{ cout<<"Data overflow!"<<endl; abort(); }

temp.a=a-x.a;

return temp;

}

Real Real::operator\*(Real x)

{

Real temp;

if(a\*x.a<-1.7e308||a\*x.a>1.7e308)

{ cout<<"Data overflow!"<<endl; abort(); }

temp.a=a\*x.a;

return temp;

}

Real Real::operator/(Real x)

{

Real temp;

if(a/x.a<-1.7e308||a/x.a>1.7e308)

{ cout<<"Data overflow!"<<endl; abort(); }

temp.a=a/x.a;

return temp;

}

Real Real::operator=(Real x)

{

a=x.a;

return \*this;

}

int main()

{

Real A(1.1),B(1.2),C;

cout<<"A=";A.display();

cout<<"B=";B.display();

C=A+B;

cout<<"C=A+B="; C.display();

C=A-B;

cout<<"C=A-B="; C.display();

C=A\*B;

cout<<"C=A\*B="; C.display();

C=A/B;

cout<<"C=A/B="; C.display();

}

4．假设有向量***X*** = ( *x*1, *x*2,…, *xn*) 和 ***Y*** = ( *y*1, *y*2, …, *yn*)，它们之间的加、减和乘法分别定义为：

***X*** + ***Y*** = ( *x*1 + *y*1*,* *x*2+ *y*2*,*…, *xn*+ *yn*)

***X*** − ***Y*** = ( *x*1 − *y*1*,* *x*2 − *y*2,…, *xn* − *yn* )

***X*** \* ***Y*** = *x*1 \* *y*1 + *x*2 \* *y*2 +,…,+ *xn*\* *yn*

编写程序定义向量类Vector，重载运算符“+”、“−”、“\*”和“=”，实现向量之间的加、减、乘、赋值运算；重载运算符“>>”、“<<”实现向量的输入、输出功能。注意检测运算的合法性。

【解答】

#include <iostream>

using namespace std;

class Vector

{

private:

double \*v;

int len;

public:

Vector(int size);

Vector(double \*,int);

~Vector();

double &operator[](int i);

Vector & operator =(Vector &);

friend Vector operator +(Vector &,Vector &);

friend Vector operator -(Vector &,Vector &);

friend double operator \*(Vector &,Vector &);

friend ostream & operator <<(ostream &output,Vector &);

friend istream & operator >>(istream &input,Vector &);

};

Vector::Vector (int size)

{

if(size<=0||size>=2147483647)

{ cout<<"The size of "<<size<<"is overflow!\n";

abort();

}

v=new double [size];

for(int i=0;i<size;i++) v[i]=0;

len=size;

}

Vector::Vector(double \*C,int size)

{

if(size<=0||size>=2147483647)

{ cout<<"The size of"<<size<<"is overflow!\n"<<endl;

abort();

}

v=new double[size];

len=size;

for(int i=0;i<len;i++) v[i]=C[i];

}

Vector::~Vector()

{

delete []v;

v=NULL; len=0;

}

double &Vector::operator[](int i)

{

if(i>=0 && i<len)

return v[i];

else

{ cout<<"The size of"<<i<<"is overflow!\n";

abort();}

}

Vector &Vector::operator =(Vector &C)

{

if(len==C.len)

{

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

v[i]=C[i];

return \*this;

}

else

{

cout<<"Operator = fail!\n";

abort();

}

}

Vector operator +(Vector &A,Vector &B) // 向量相加

{

int size=A.len ;

double \*T=new double[size];

if(size==B.len)

{

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

T[i]=A[i]+B[i];

return Vector (T,size);

}

else

{

cout<<"Operator + fail!\n";

abort();

}

}

Vector operator -(Vector &A,Vector &B) //向量相减

{

int size=A.len ;

double \*T=new double[size];

if(size==B.len)

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

T[i]=A[i]-B[i];

return Vector (T,size);

}

else

{

cout<<"Operator - fail!\n";

abort();

}

}

double operator \*(Vector &A,Vector &B) //向量相乘

{

int size=A.len ;

double s=0;

if( size==B.len )

{

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

s+=A[i]\*B[i];

return s;

}

else

{

cout<<"Operator \* fail!\n";

abort();

}

}

ostream & operator <<(ostream &output,Vector &A) //输出

{

int i;

output<<'(';

for( i=0;i<A.len-1;i++)

output<<A[i]<<',';

output<<A[i]<<')';

return output;

}

istream & operator >>(istream &input,Vector &A) //输入

{

for(int i=0;i<A.len;i++)

input>>A[i];

return input;

}

int main()

{

int k1,k2,k3;double t;

cout<<"Input the length of Vector A:\n";

cin>>k1;

Vector A(k1);

cout<<"Input the elements of Vector A:\n";

cin>>A;

cout<<"Input the length of Vector B:\n";

cin>>k2;

Vector B(k2);

cout<<"Input the elements of Vector B:\n";

cin>>B;

cout<<"Input the length of Vector C:\n";

cin>>k3;

Vector C(k3);

cout<<"A="<<A<<endl;

cout<<"B="<<B<<endl;

C=A+B;

cout<<"A+B="<<A<<"+"<<B<<"="<<C<<endl;

C=A-B;

cout<<"A-B="<<A<<"-"<<B<<"="<<C<<endl;

t=A\*B;

cout<<"A\*B="<<A<<"\*"<<B<<"="<<t<<endl;

}

5．定义一个类nauticalmile\_kilometer，它包含两个数据成员kilometer（千米）和meter（米）；还包含一个构造函数对数据成员进行初始化；成员函数print，用于输出数据成员kilometer和meter的值；类型转换函数operator double，实现把千米和米转换为海里（1海里=1.852千米）的功能。编写main函数，测试类nauticalmile\_kilometer。

【解答】

#include <iostream>

using namespace std;

const double n = 1.852; // 定义海里与千米和米的转换系数(1海里=1.852千米)

class nauticalmile\_kilometer

{

public:

nauticalmile\_kilometer( int km,double m )

{

kilometer = km; meter = m;

}

void print()

{

cout<<"kilometer="<<kilometer<<'\t'<<"meter="<<meter<<endl;

}

operator double();

private:

int kilometer;

double meter;

};

nauticalmile\_kilometer::operator double()

{

return ( meter / 1000 + double( kilometer )) / n;

}

int main()

{

nauticalmile\_kilometer obj( 100,50 );

obj.print();

cout << "nauticalmile=" << double( obj ) << endl;

}

6．定义一个集合类setColour，要求元素为枚举类型值。例如，

enum colour { red, yellow, blue, white, black };

集合类实现交、并、差、属于、蕴含、输入、输出等各种基本运算。设计main函数测试setColour类的功能。

【解答】

枚举类型数据用序值参与运算。定义枚举集合关键是对输入/输出操作进行显示转换。

程序略。

## 第8章 继承

#### 8.1 选择题

1．在C++中，类之间的继承关系具有（ C ）。

（A）自反性 （B）对称性 （C）传递性 （D）反对称性

2．在下列关于类的继承描述中，正确的是（ D ）。

（A）派生类公有继承基类时，可以访问基类的所有数据成员，调用所有成员函数

（B）派生类也是基类，所以它们是等价的

（C）派生类对象不会建立基类的私有数据成员，所以不能访问基类的私有数据成员

（D）一个基类可以有多个派生类，一个派生类可以有多个基类

3．当一个派生类公有继承一个基类时，基类中的所有公有成员成为派生类的（ A ）。

（A）public成员 （B）private成员 （C）protected成员 （D）友元

4．当一个派生类私有继承一个基类时，基类中的所有公有成员和保护成员成为派生类的（ B ）。

（A）public成员 （B）private成员 （C）protected成员 （D）友元

5．当一个派生类保护继承一个基类时，基类中的所有公有成员和保护成员成为派生类的（ C ）。

（A）public成员 （B）private成员 （C）protected成员 （D）友元

6．不论派生类以何种方式继承基类，都不能直接使用基类的（ B ）。

（A）public 成员 （B）private成员

（C）protected成员 （D）所有成员

7．在下列描述中，错误的是（ D ）。

（A）在基类中定义的public成员在公有继承的派生类中可见，也能在类外被访问

（B）在基类中定义的protected成员在私有继承的派生类中可见

（C）在基类中定义的公有静态成员在私有继承的派生类中可见

（D）访问声明可以在公有继承派生类中把基类的public成员声明为private成员

8．在C++中，可以被派生类继承的函数是（ A ）。

（A）成员函数 （B）构造函数 （C）析构函数 （D）友元函数

9．在创建派生类对象时，构造函数的执行顺序是（ D ）。

（A）对象成员构造函数→基类构造函数→派生类本身的构造函数

（B）派生类本身的构造函数→基类构造函数→对象成员构造函数

（C）基类构造函数→派生类本身的构造函数→对象成员构造函数

（D）基类构造函数→对象成员构造函数→派生类本身的构造函数

10．当不同的类具有相同的间接基类时，（ C ）。

（A）各派生类无法按继承路线产生自己的基类版本

（B）为了建立唯一的间接基类版本，应该声明间接基类为虚基类

（C）为了建立唯一的间接基类版本，应该声明派生类虚继承基类

（D）一旦声明虚继承，基类的性质就改变了，不能再定义新的派生类

#### 8.2 阅读下列程序，写出运行结果

1．

#include<iostream>

using namespace std;

class Base

{

public :

void get( int i,int j,int k,int l )

{

a = i; b = j; x = k; y = l;

}

void print()

{

cout << "a = "<< a << '\t' << "b = " << b << '\t'

<< "x = " << x << '\t' << "y = " << y << endl;

}

int a, b;

protected :

int x, y;

};

class A : public Base

{

public :

void get( int i, int j, int k, int l )

{

Base obj3;

obj3.get( 50, 60, 70, 80 );

obj3.print();

a = i; b = j; x = k; y = l;

u = a + b + obj3.a; v = y - x + obj3.b;

}

void print()

{

cout << "a = " << a << '\t' << "b = " << b << '\t'

<< "x = " << x << '\t' << "y = " << y << endl;

cout << "u = " << u << '\t' << "v = " << v << endl;

}

private:

int u, v;

};

int main()

{

Base obj1;

A obj2;

obj1.get( 10, 20, 30, 40 );

obj2.get( 30, 40, 50, 60 );

obj1.print();

obj2.print();

}

【解答】

a = 50 b = 60 x = 70 y = 80

a = 10 b = 20 x = 30 y = 40

a = 30 b = 40 x = 50 y = 60

u = 120 v = 70

2．

#include<iostream>

using namespace std;

class Base1

{

public :

Base1( int i )

{ cout << "调用基类Base1的构造函数:" << i << endl; }

};

class Base2

{

public:

Base2( int j )

{ cout << "调用基类Base2的构造函数:" << j << endl; }

};

class A : public Base1, public Base2

{

public :

A( int a,int b,int c,int d ):Base2(b),Base1(c),b2(a),b1(d)

{ cout << "调用派生类A的构造函数:" << a+b+c+d << endl; }

private :

Base1 b1;

Base2 b2;

};

int main()

{

A obj( 1, 2, 3, 4 );

}

【解答】

调用基类BASE1的构造函数:3

调用基类BASE2的构造函数:2

调用基类BASE1的构造函数:4

调用基类BASE2构造函数:1

调用派生类A的构造函数:10

3．

#include<iostream>

using namespace std;

class A

{

public :

A( int i, int j ) {a=i; b=j;}

void Add( int x, int y ) { a += x; b += y; }

void show() { cout << "("<<a<<")\t("<<b<<")\n"; }

private :

int a, b;

};

class B : public A

{

public :

B(int i, int j, int m, int n) : A( i, j ),x( m ), y( n ) {}

void show() { cout << "(" << x << ")\t(" << y << ")\n"; }

void fun() { Add( 3, 5 ); }

void ff() { A::show(); }

private :

int x, y;

};

int main()

{

A a( 1, 2 );

a.show();

B b( 3, 4, 5, 6 );

b.fun();

b.A::show();

b.show();

b.ff();

}

【解答】

(1) (2)

(6) (9)

(5) (6)

(6) (9)

4．

#include<iostream>

using namespace std;

class A

{

public :

A(const char \*s) { cout << s << endl; }

~A() {}

};

class B : virtual public A

{

public :

B(const char \*s1, const char \*s2) : A( s1 ) { cout << s2 << endl; }

};

class C : virtual public A

{

public :

C(const char \*s1, const char \*s2):A(s1) { cout << s2 << endl; }

};

class D : public B, public C

{

public :

D( const char \*s1,const char \*s2,const char \*s3,const char \*s4 ):

B( s1, s2 ), C( s1, s3 ), A( s1 )

{ cout << s4 << endl; }

};

int main()

{

D \*ptr = new D( "class A", "class B", "class C", "class D" );

delete ptr;

}

【解答】

class A

class B

class C

class D

#### 8.3 思考题

1．函数和类这两种程序模块都可以实现软件重用，它们之间有什么区别？

【解答】

函数是基于参数集的功能抽象模块，以调用方式实现软件重用，函数之间没有逻辑关系。

类是数据属性与操作的封装，以继承方式实现软件重用，类之间构成有向无回图的类格。

2．请按照类成员的访问特性、类层次的继承特点，制作一张表格，总结各种类成员在基类、派生类中的可见性和作用域。

【解答】

|  |  |  |  |
| --- | --- | --- | --- |
| 基类成员  派生类继承 | public | protected | private |
| public | 在派生类中访问特性不变。派生类和类外均可见，有作用域。 | 在派生类中访问特性不变。类体系中可见。 | 基类私有成员，仅在基类中可见。 |
| protected | 成为派生类保护段成员。在整个类体系中可见。 | |
| private | 成为派生类私有成员。仅在派生类和基类中可见。 | |
|  | 派生类不论以何种方式继承基类，基类所有成员在整个类体系有作用域。 | | |

3．若有以下说明语句：

class A

{

private : int a1;

public : int a2; double x;

/\*…\*/

};

class B : private A

{

private : int b1;

public int b2; double x;

/\*…\*/

};

B b;

对象b将会生成什么数据成员？与继承关系、访问特性、名字有关吗？

【解答】

对象b生成的数据成员有a1 a2 A::x b1 b2 B::x，共六个数据成员。数据成员的建立和继承关系、访问特性、名字无关。

4．若有以下说明语句：

class A

{

/\*…\*/

public : void sameFun();

};

class B : public A

{

/\*…\*/

public : void sameFun();

};

void comFun()

{

A a;

B b;

/\*…\*/

}

（1）若在B::sameFun中调用A::sameFun，语句格式如何？它将在什么对象上操作？

（2）在comFun中可以用什么方式调用A::sameFun和B::sameFun？语句格式如何？它们将可以在什么对象上操作？

【解答】

（1）若要在B::sameFun中调用A::sameFun，语句形式应为：

A::samefun(); //域作用符说明调用基类函数

调用的A::samefun将在B类对象上进行操作。

（2）在comFun中调用B::sameFun和A::sameFun的方式有：

a.A::sameFun(); //通过A类对象调用A::sameFun，在a类对象上操作。

b.sameFun(); //通过B类对象调用B::sameFun，在b类对象上操作。

b.A::sameFun(); //通过B类对象调用A::sameFun

//在b类对象上（对由基类继承下来的数据成员）操作。

5．有人定义一个教师类派生一个学生类。他认为“姓名”和“性别”是教师、学生共有的属性，声明为public，“职称”和“工资”是教师特有的，声明为private。在学生类中定义特有的属性“班级”和“成绩”。所以有：

class teacher

{

public :

char name[20]; char sex;

//…

private :

char title[20]; double salary;

};

class student : public teacher

{

//…

private :

char grade[20]; int score;

};

你认为这样定义合适吗？请给出你认为合理的类结构定义。

【解答】

不合适，这样导致数据冗余。合理的结构是提取它们共有的数据和操作定义一个基类，然后分别定义teacher和student作为派生类。

class person

{ protected:

char name[20]; char sex;

//……

};

class teacher : public teache

{ //……

private:

char title[20]; double salary;

};

class student : public teacher

{ //……

private :

char grade[20] ; int score;

};

6．在第6章的例6-21中，定义Student类包含了Date类成员。可以用继承方式把Student类定义为Date类的派生类吗？如何改写程序？请你试一试。

【解答】

可以用继承方式改写。程序略。

7．“虚基类”是通过什么方式定义的？如果类A有派生类B、C，类A是类B虚基类，那么它也一定是类C的虚基类吗？为什么？

【解答】

虚基类是在声明派生类时，指定继承方式时声明的，声明虚基类的一般形式为：

class 派生类名 ： virtual 继承方式 基类名

若类A是类B和类C的虚基类，但不一定是类D的虚基类，原因在于“虚基类”中的“虚”不是基类本身的性质。而是派生类在继承过程中的特性。关键字virtual只是说明该派生类把基类当作虚基类继承，不能说明基类其他派生类继承基类的方式

8．在具有虚继承的类体系中，建立派生类对象时，以什么顺序调用构造函数？请用简单程序验证你的分析。

【解答】

在具有虚继承的类体系中，建立派生类对象时先调用间接基类构造函数，再按照派生类定义时各个直接基类继承的顺序调用直接基类的构造函数，最后再对派生类对象自身构造函数。

另外，C++为了保证虚基类构造函数只被建立对象的类执行一次，规定在创建对象的派生类构造函数中只调用虚基类的构造函数和进行(执行)自身的初始化。参数表中的其他调用被忽略，即直接基类的构造函数只调用系统自带的版本，或调用自定义版本但不对虚基类数据成员初始化。

程序略。

#### 8.4 编程题

1．定义一个Rectangle类，它包含两个数据成员length和width，以及用于求长方形面积的成员函数。再定义Rectangle的派生类Rectangular，它包含一个新数据成员height和用来求长方体体积的成员函数。在main函数中，使用两个类，求某个长方形的面积和某个长方体的体积。

【解答】

#include <iostream>

using namespace std;

class rectangle

{

public:

rectangle( double l,double w )

{ length = l;width = w; }

double area()

{ return( length\*width ); }

double getlength()

{ return length; }

double getwidth()

{ return width; }

private:

double length;

double width;

};

class rectangular:public rectangle

{

public:

rectangular( double l,double w,double h ) : rectangle( l,w )

{ height = h; }

double getheight()

{ return height; }

double volume()

{ return area() \*height; }

private:

double height;

};

int main()

{

rectangle obj1( 2,8 );

rectangular obj2( 3,4,5 );

cout<<"length="<<obj1.getlength()<<'\t'<<"width="<<obj1.getwidth()<<endl;

cout<<"rectanglearea="<<obj1.area()<<endl;

cout<<"length="<<obj2.getlength()<<'\t'<<"width="<<obj2.getwidth();

cout<<'\t'<<"height="<<obj2.getheight()<<endl;

cout<<"rectangularvolume="<<obj2.volume()<<endl;

}

2．假设某销售公司有一般员工、销售员工和销售经理。月工资的计算办法是：

一般员工月薪=基本工资；

销售员工月薪=基本工资+销售额\*提成率；

销售经理月薪=基本工资+职务工资+销售额\*提成率。

编写程序，定义一个表示一般员工的基类Employee，它包含3个表示员工基本信息的数据成员：编号number、姓名name和基本工资basicSalary；

由Employee类派生销售员工Salesman类，Salesman类包含2个新数据成员：销售额sales和静态数据成员提成比例commrate；

再由Salesman类派生表示销售经理的Salesmanager类。Salesmanager类包含新数据成员：岗位工资jobSalary。

为这些类定义初始化数据的构造函数，以及输入数据input、计算工资pay和输出工资条print的成员函数。

设公司员工的基本工资是2000元，销售经理的岗位工资是3000元，提成率=5/1000。在main函数中，输入若干个不同类型的员工信息测试你的类结构。

#include <iostream>

using namespace std;

class Employee

{

public:

Employee( char Snumber[]="\0", char Sname[]="\0", double bSalary=2000 )

{

strcpy(number,Snumber);

strcpy(name,Sname);

basicSalary=bSalary;

}

void input()

{

cout << "编号：" ; cin >> number;

cout <<"姓名：" ; cin >> name;

}

void print()

{

cout<<"员工 ："<<name<<"\t\t编号："<<number<<"\t\t本月工资："<<basicSalary<<endl;

}

protected:

char number[5];

char name[10];

double basicSalary;

};

class Salesman: public Employee

{

public:

Salesman(int sal=0)

{ sales=sal; }

void input()

{

Employee::input();

cout<<"本月个人销售额：";

cin>>sales;

}

void pay()

{

salary = basicSalary+sales\*commrate;

}

void print()

{

pay();

cout<<"销售员 ："<<name<<"\t\t编号："<<number<<"\t\t本月工资："<<salary<<endl; }

protected:

static double commrate;

int sales;

double salary;

};

double Salesman :: commrate=0.005;

class Salesmanager : public Salesman

{

public:

Salesmanager(double jSalary=3000)

{

jobSalary = jSalary;

}

void input()

{

Employee::input();

cout<<"本月部门销售额：";

cin>>sales;

}

void pay()

{

salary = jobSalary + sales\*commrate;

}

void print()

{

pay();

cout<<"销售经理 ："<<name<<"\t\t编号："<<number<<"\t\t本月工资："<<salary<<endl; }

private:

double jobSalary;

};

int main()

{

cout<<"基本员工\n";

Employee emp1;

emp1.input();

emp1.print();

cout<<"销售员\n";

Salesman emp2;

emp2.input();

emp2.print();

cout<<"销售经理\n";

Salesmanager emp3;

emp3.input();

emp3.print();

}

3．试写出你所能想到的所有形状（包括二维的和三维的），生成一个形状层次类体系。生成的类体系以Shape作为基类，并由此派生出TwoDimShape类和ThreeDimShape类。它们的派生类是不同的形状类。定义类体系中的每一个类，并用main函数进行测试。

【解答】

略。

4．为第7章习题7.4第2小题和第3小题的Integer和Real类定义一个派生类IntReal：

class IntReal : public Integer, public Real;

使其可以进行+、−、\*、/、=的左、右操作数类型不同的相容运算，并符合原有运算类型转换的语义规则。

【解答】

略。

5．使用Integer类，定义派生类Vector类：

class Integer

{ //…

protected :

int n;

};

class Vector:public Integer

{ //…

protected :

int \*v;

};

其中，数据成员v用于建立向量，n为向量长度。要求：类的成员函数可以实现向量的基本算术运算。

【解答】

略。

6．用包含方式改写习题8.4第5题中的Vector类，使其实现相同的功能。

class Vector

{ //…

protected :

Integer \*v;

Integer size;

};

【解答】

略。

7．使用Vector类，定义派生类Matrix，实现矩阵的基本算术运算。

【解答】

略。

8．用包含方式改写习题8.4第7题的Matrix类，使其实现相同的功能。

【解答】

略。

## 第9章 虚函数与多态性

#### 9.1 选择题

1．在C++中，要实现动态联编，必须使用（ D ）调用虚函数。

（A）类名 （B）派生类指针 （C）对象名 （D）基类指针

2．在下列函数中，不能说明为虚函数的是（ C ）。

（A）私有成员函数 （B）公有成员函数 （C）构造函数 （D）析构函数

3．在派生类中重载一个虚函数时，要求函数名、参数的个数、参数的类型、参数的顺序和函数的返回值（ A ）。

（A）相同 （B）不同 （C）相容 （D）部分相同

4．在C++中，根据（ D ）识别类层次中不同类定义的虚函数版本。

（A）参数个数 （B）参数类型 （C）函数名 （D）this指针类型

5．虚析构函数的作用是（ C ）。

（A）虚基类必须定义虚析构函数 （B）类对象作用域结束时释放资源

（C）delete动态对象时释放资源 （D）无意义

6．在下列函数原型中，（ B ）声明了fun为纯虚函数。

（A）void fun()=0; （B）virtual void fun()=0;

（C）virtual void fun(); （D）virtual void fun(){ };

7．若一个类中含有纯虚函数，则该类称为（ C ）。

（A）基类 （B）纯基类 （C）抽象类 （D）派生类

8．假设 Aclass为抽象类，下列正确的说明语句是（ B ）。

（A）Aclass fun( int ); （B）Aclass \* p;

（C）int fun( Aclass ); （D）Aclass Obj;

9．在下列描述中，正确的是（ D ）。

（A）虚函数是没有实现的函数 （B）纯虚函数是返回值等于0的函数

（C）抽象类是只有纯虚函数的类 （D）抽象类指针可以指向不同的派生类

10．构造异质链表的意义是（ D ）。

（A）用数组组织类对象 （B）用链表组织类对象

（C）用抽象类指针指向派生类对象 （D）用抽象类指针构造派生类对象链表

#### 9.2 阅读下列程序，写出执行结果

1．

#include <iostream>

using namespace std;

class Bclass

{

public:

Bclass( int i, int j )

{

x = i;

y = j;

}

virtual int fun()

{

return 0;

}

protected:

int x, y;

};

class Iclass:public Bclass

{

public :

Iclass(int i, int j, int k):Bclass(i, j)

{

z = k;

}

int fun()

{

return ( x + y + z ) / 3;

}

private :

int z;

};

int main()

{

Iclass obj( 2, 4, 10 );

Bclass p1 = obj;

cout << p1.fun() << endl;

Bclass &p2 = obj;

cout << p2.fun() << endl;

cout << p2.Bclass :: fun() << endl;

Bclass \*p3 = &obj;

cout << p3 -> fun() << endl;

}

【解答】

0

5

0

5

2．

#include <iostream>

using namespace std;

class Base

{

public:

virtual void getxy( int i,int j = 0 )

{

x = i;

y = j;

}

virtual void fun() = 0;

protected:

int x , y;

};

class A : public Base

{

public:

void fun()

{

cout<<"x = "<<x<<'\t'<<"y = x \* x = "<<x\*x<<endl;

}

};

class B : public Base

{

public:

void fun()

{

cout << "x = " << x << '\t' << "y = " << y << endl;

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

}

};

int main()

{

Base \* pb;

A obj1;

B obj2;

pb = &obj1;

pb -> getxy( 10 );

pb -> fun();

pb = &obj2;

pb -> getxy( 100, 20 );

pb -> fun();

}

【解答】

x = 10 y = x\*x = 100

x = 100 y = 20

y = x / y = 5

#### 9.3　思考题

1．在C++中，使用类体系依靠什么机制实现程序运行时的多态？

在C++中，基类指针可以指向派生类对象，以及基类中拥有虚函数，是支持多态性的前提。程序通过用同一个基类指针访问不同派生类的虚函数重载版本实现程序运行时的多态。C++的虚特性负责自动地在程序运行时把基类指针的关联类型转换成当前指向对象的派生类类型。

另外，抽象类机制提供了软件抽象和可扩展性的手段，实现运行时的多态性。

2．如果一个类的虚函数被声明为私有成员函数，会有语法错误吗？当它作为基类时，可以在应用类体系时实现动态联编吗？请验证一下。

没有语法错误。但在应用类体系时无法实现动态编联和多态。因为私有成员函数只在类内可见，在类外无法调用，无法在类外通过基类指针实现多态。

程序略。

3．虚函数和纯虚函数的区别是什么？

【解答】

虚函数定义时冠以关键字virtual,本身有实现代码，作用是引导基类指针根据指向对象调用类体系中不同重载版本函数。

纯虚函数是指在说明时代码“为0”的虚函数，即纯虚函数本身并没有实现代码，必须通过它的派生类定义实现版本。

4．一个非抽象类的派生类是否可以为抽象类？利用例9-11进行验证，从Hex\_type类派生一个Hex\_format类，其中包含一个纯虚函数Show\_format，然后定义Hex\_format的派生类定义实现Show\_format。

【解答】

一个非抽象类的派生类可以为抽象类，即在派生类中定义了纯虚函数。

程序略。

#### 9.4　编程题

1．使用虚函数编写程序，求球体和圆柱体的体积及表面积。由于球体和圆柱体都可以看做由圆继承而来，因此，可以把圆类Circle作为基类。在Circle类中定义一个数据成员radius及两个虚函数area和volume。由Circle类派生Sphere类和Column类。在派生类中对虚函数area和volume重新定义，分别求球体和圆柱体的体积及表面积。

【解答】

#include <iostream>

using namespace std;

const double PI=3.14159265;

class circle

{

public:

circle(double r) { radius = r; }

virtual double area() { return 0.0; }

virtual double volume() { return 0.0; }

protected:

double radius;

};

class sphere:public circle

{

public:

sphere( double r ):circle( r ){}

double area()

{ return 4.0 \* PI \* radius \* radius; }

double volume()

{ return 4.0 \* PI \* radius \* radius \* radius / 3.0; }

};

class column:public circle

{

public:

column( double r,double h ):circle( r ) { height = h; }

double area()

{ return 2.0 \* PI \* radius \* ( height + radius ); }

double volume()

{ return PI \* radius \* radius \* height; }

private:

double height;

};

int main()

{

circle \*p;

sphere sobj(2);

p = &sobj;

cout << "球体:" << endl;

cout << "体积 = " << p->volume() << endl;

cout << "表面积 = " << p->area() << endl;

column cobj( 3,5 );

p = &cobj;

cout << "圆柱体:" << endl;

cout << "体积 = " << p->volume() << endl;

cout << "表面积 = " << p->area() << endl;

}

2．某学校教职工的工资计算方法为：

所有教职工都有基本工资；

教师月工资为固定工资+课时补贴。课时补贴根据职称和课时计算。例如，每课时教授补贴50元，副补贴30元，讲师补贴20元。

管理人员月薪为基本工资+职务工资；

实验室人员月薪为基本工资+工作日补贴。工作日补贴等于日补贴×月工作日数。

定义教职工抽象类，派生教师、管理人员和实验室类，编写程序测试这个类体系。

【解答】

#include <iostream>

using namespace std;

class staff

{

public:

staff ( double bSalary)

{

basicSalary=bSalary;

}

virtual void input() = 0;

virtual void output() = 0;

protected:

char name[30];

double basicSalary;

};

class teacher : public staff

{

public:

teacher( int basicsalary=3000 ) : staff( basicsalary ){ }

void input()

{

cout<<"姓名？";

cin>>name;

cout<<"职称 1，教授 2，副教授 3，讲师 （输入1，2 或 3）：";

cin>>title;

cout<<"课时?";

cin>>coursetime;

}

void output()

{

double salary;

switch(title)

{

case 1: salary = basicSalary+coursetime \*50; break;

case 2: salary=basicSalary+coursetime\*30; break;

case 3: salary=basicSalary+coursetime\*20;

}

cout<<"姓名："<<name<<"\t本月工资："<<salary<<endl;

}

protected:

int coursetime;

int title;

};

class manage : public staff

{

public:

manage( int basicsalary=2500 ) : staff( basicsalary ){ }

void input()

{

cout<<"姓名？";

cin>>name;

cout<<"职务工资？ ";

cin>>jobSalary;

}

void output()

{

double salary;

salary = basicSalary+jobSalary;

cout<<"姓名："<<name<<"\t本月工资："<<salary<<endl;

}

protected:

double jobSalary;

};

class technician : public staff

{

public:

technician( int basicsalary=2000 ) : staff( basicsalary ){ }

void input()

{

cout<<"姓名？";

cin>>name;

cout<<"工作日？";

cin>>workdays;

}

void output()

{

double salary;

salary = basicSalary+workdays\*20;

cout<<"姓名："<<name<<"\t本月工资："<<salary<<endl;

}

protected:

int workdays;

};

int main()

{

teacher t;

t.input();

t.output();

manage m;

m.input();

m.output();

technician h;

h.input();

h.output();

}

3．使用习题9.4第2小题中定义的教师类体系，编写程序，输入某月各种职称教师的工资信息，建立异质链表，输出每位教师的工资条，统计当月的总工资、平均工资、最高工资和最低工资。

【解答】

略

4．改写第8章习题8.4第3小题，把Shape类定义为抽象类，提供共同操作界面的纯虚函数。TwoDimShape类和ThreeDimShape类仍然是抽象类，只有第3层具体类才提供全部函数的实现。在测试函数中，使用基类指针实现不同派生类对象的操作。

【解答】

略

## 第10章 模板

#### 10.1 选择题

1．关于函数模板，描述错误的是（ A ）。

（A）函数模板必须由程序员实例化为可执行的函数模板

（B）函数模板的实例化由编译器实现

（C）一个类定义中，只要有一个函数模板，这个类就是类模板

（D）类模板的成员函数都是函数模板，类模板实例化后，成员函数也随之实例化

2．在下列模板说明中，正确的是（ C ）。

（A）template < typename T1, T2 >

（B）template < class T1, T2 >

（C）template < typename T1, typename T2 >

（D）template ( typedef T1, typedef T2 )

3．假设有函数模板定义如下：

template <typename T>

Max( T a, T b ,T &c)

{ c　= a + b; }

下列选项正确的是（ B ）。

（A）int x, y; char z; （B）double x, y, z;

Max( x, y, z ); Max( x, y, z );

（C）int x, y; float z; （D）float x; double y, z;

Max( x, y, z ); Max( x, y, z );

4．关于类模板，描述错误的是（ A ）。

（A）一个普通基类不能派生类模板

（B）类模板可以从普通类派生，也可以从类模板派生

（C）根据建立对象时的实际数据类型，编译器把类模板实例化为模板类

（D）函数的类模板参数需生成模板类并通过构造函数实例化

5． 建立类模板对象的实例化过程为（ C ）。

（A）基类→派生类 （B）构造函数→对象

（C）模板类→对象 （D）模板类→模板函数

6．在C++中，容器是一种（ D ）。

（A）标准类 （B）标准对象 （C）标准函数 （D）标准类模板

#### 10.2 阅读下列程序，写出运行结果

1．

#include <iostream>

using namespace std;

template <typename T>

void fun( T &x, T &y )

{

T temp;

temp = x; x = y; y = temp;

}

int main()

{

int i , j;

i = 10; j = 20;

fun( i, j );

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

double a , b;

a = 1.1; b = 2.2;

fun( a, b );

cout << "a = " << a << '\t' << "b = " << b << endl;

}

【解答】

i=20 j=10

a=2.2 b=1.1

2．

#include <iostream>

using namespace std;

template <typename T>

class Base

{

public:

Base( T i , T j ) { x = i; y = j; }

T sum() { return x + y; }

private:

T x, y;

};

int main()

{

Base<double> obj2(3.3,5.5);

cout << obj2.sum() << endl;

Base<int> obj1(3,5);

cout << obj1.sum() << endl;

}

【解答】

8.8

8

#### 10.3 思考题

1．抽象类和类模板都是提供抽象的机制，请分析它们的区别和应用场合。

【解答】

抽象类至少包含一个纯虚函数，纯虚函数抽象了类体系中一些类似操作的公共界面，它不依赖于数据，也没有操作定义。派生类必须定义实现版本。抽象类用于程序开发时对功能的统一策划，利用程序运行的多态性自动匹配实行不同版本的函数。

类模板抽象了数据类型，称为类属参数。成员函数描述了类型不同，逻辑操作相同的功能集。编译器用建立对象的数据类型参数实例化为模板类，生成可运行的实体。类模板用于抽象数据对象类型不同，逻辑操作完全相同类定义。这种数据类型的推导必须在语言功能的范畴之内的。

2．类属参数可以实现类型转换吗？如果不行，应该如何处理？

【解答】

类属参数不可以实现类型转换。为了解决参数隐式类型转换的问题，可以用类型参数把函数模板重载为非模板函数。

3．类模板能够声明什么形式的友元？当类模板的友元是函数模板时，它们可以定义不同形式的类属参数吗？请编写一个验证程序试一试。

【解答】

类模板可以声明的友员形式有：普通函数、函数模板、普通类成员函数、类模板成员函数以及普通类、类模板。

当类模板的友员是函数模板时，它们可以定义不同形式的类属参数。

程序略。

4．类模板的静态数据成员可以是抽象类型吗？它们的存储空间是什么时候建立的？请用验证程序试一试。

【解答】

类模板的静态数据成员可以是抽象类型。它们的存储空间在生成具体模板类的时候建立，即每生成一个模板类同时建立静态储存空间并做一次文件范围的初始化。

程序略。

#### 10.4 编程题

1．使用函数模板实现对不同类型数组求平均值的功能，并在main函数中分别求一个整型数组和一个浮点型数组的平均值。

【解答】

#include <iostream>

using namespace std;

template <typename T>

double average( T \*array,int size )

{

T sum = 0;

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

sum += array[i];

return sum / size;

}

int main()

{

int a[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

double b[] = { 1.1, 2.2, 3.3, 4.4, 5.5, 6.6, 7.7, 8.8, 9.9, 10 };

cout << "Average of array a :" << average( a,10 ) << endl;

cout << "Average of array b :" << average( b,10 ) << endl;

}

2．建立结点，包括一个任意类型数据域和一个指针域的单向链表类模板。在main函数中使用该类模板建立数据域为整型的单向链表，并把链表中的数据显示出来。

【解答】

#include <iostream>

using namespace std;

template <typename T>

class List

{

public:

List( T x ) { data = x; }

void append( List \*node )

{

node->next = this;

next = 0;

}

List \*getnext() { return next; }

T getdata() { return data; }

private:

T data;

List \*next;

};

int main()

{

int i, idata, n, fdata;

cout << "输入结点的个数：";

cin >> n;

cout << "输入结点的数据域：";

cin >> fdata;

List <int> headnode( fdata );

List <int> \*p, \*last;

last = &headnode;

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

{

cin >> idata;

p = new List <int>( idata );

p->append( last );

last = p;

}

cout << "链表已经建立！" << endl;

cout << "链表中的数据为：" << endl;

p = &headnode;

while( p )

{

cout << p->getdata() << endl;

p = p->getnext();

}

}

3．定义类模板T\_Counter，实现基本类型数据的+、−、\*、=、>>、<< 运算；类模板T\_Vector，实现向量运算；类模板T\_Matrix，实现矩阵运算。请分析使用类模板建立T\_Counter、T\_Vector、T\_Matrix对象和使用类继承体系建立IntReal、Vector、Matrix对象（见第8章习题8.4第4、5、7小题）的语法区别和运算功能区别。

【解答】

略。

4．学习MSDN Library中Visual C++的STL，应用容器和算法，实现一个简单的人员信息管理系统。

【解答】

略。

## 第11章 输入/输出流

#### 11.1 选择题

1．在下列流类中，可以用于处理文件的是( D )。

（A）ios （B）iostream （C）strstream （D）fstream

2．在下列选项中，（ B ）是istream类的对象。

（A）cerr （B）cin （C）clog （D）cout

3．用于处理字符串流的是（ A ）。

（A）strstream （B）ios （C）fstream （D）iostream

4．能够从输入流中提取指定长度的字节序列的函数是（ C ）。

（A）get （B）getline （C）read （D）cin

5．能够把指定长度的字节序列插入到输出流中的函数是（ B ）。

（A）put （B）write （C）cout （D）print

6．getline函数的功能是从输入流中读取（ C ）。

（A）一个字符 （B）当前字符 （C）一行字符 （D）指定若干个字节

7．在下列选项中，用于清除基数格式位设置以十六进制数输出的语句是（ B ）。

（A）cout<<setf( ios::dec, ios::basefield );

（B）cout<<setf( ios::hex, ios::basefield );

（C）cout<<setf( ios::oct, ios::basefield );

（D）cin>>setf( ios::hex, ios::basefield );

8．下列格式控制符，既可以用于输入，又可以用于输出的是（ A ）。

（A）setbase （B）setfill （C）setprecision （D）setw

9．要求打开文件 D:\file.dat，并能够写入数据，正确的语句是（ D ）。

（A）ifstream infile("D:\\file.dat", ios::in );

（B）ifstream infile("D:\\file.dat", ios::out );

（C）ofstream outfile("D:\\file.dat", ios::in );

（D）fstream infile("D:\\file.dat", ios::in|ios::out );

10．能实现删除文件功能的语句是（ A ）。

（A）ofstream fs("date.dat", ios::trunc );

（B）ifstream fs("date.dat", ios::trunc );

（C）ofstream fs("date.dat", ios::out );

（D）ifstream fs("date.dat", ios::in );

11．设已定义浮点型变量data，以二进制代码方式把data的值写入输出文件流对象outfile中，正确的语句是（ C ）。

（A）outfile.write((double \*) &data, sizeof(double));

（B）outfile.write((double \*) &data, data);

（C）outfile.write((char \*) &data, sizeof(double));

（D）outfile.write((char \*) &data, data);

12．二进制数据文件流fdat读指针移到文件头的语句是（ A ）。

（A）fdat.seekg( 0, ios::beg); （B）fdat.tellg( 0, ios::beg );

（C）fdat.seekp( 0, ios::beg); （D）fdat.tellp( 0, ios::beg );

#### 11.2 阅读下列程序，写出运行结果

1．

#include<iostream>

using namespace std;

int main()

{

double x = 123.456;

cout.width( 10 );

cout.setf( ios::dec, ios::basefield );

cout<<x<<endl;

cout.setf( ios::left );

cout<<x<<endl;

cout.width( 15 );

cout.setf( ios::right , ios::left );

cout<<x<<endl;

cout.setf( ios::showpos );

cout<<x<<endl;

cout<<-x<<endl;

cout.setf( ios::scientific );

cout<<x<<endl;

}

【解答】

123.456

123.456

123.456

+123.456

-123.456

+1.234560e+002

2．

#include<iostream>

using namespace std;

int main()

{

double x = 123.45678;

cout.width( 10 );

cout<<( "#" );

cout<<x<<endl;

cout.precision( 5 );

cout<<x<<endl;

cout.setf( ios::showpos );

cout<<x<<endl;

cout.setf( ios::scientific );

cout<<x<<endl;

}

【解答】

#123.457

123.46

+123.46

+1.23457e+002

3．

#include<iostream>

#include <iomanip>

using namespace std;

int main()

{

double x = 123.456789;

cout<<setiosflags( ios::fixed | ios::showpos )<<x<<endl;

cout<<setw( 12 )<<setiosflags( ios::right );

cout<<setprecision( 3 )<<-x<<endl;

cout<<resetiosflags( ios::fixed | ios::showpos )<<setiosflags( ios::scientific );

cout<<setprecision( 5 )<<x<<endl;

}

【解答】

+123.456789

-123.457

1.23457e+002

4．写出文件D:\f1.txt中的内容和屏幕显示的结果。

#include<iostream>

#include<fstream>

using namespace std;

int main()

{

int i;

ofstream ftxt1;

ftxt1.open( "D:\\f1.txt", ios::out );

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

ftxt1<<i<<' ';

ftxt1.close();

ifstream ftxt2;

ftxt2.open( "D:\\f1.txt", ios::in );

while( !ftxt2.eof() )

{

ftxt2>>i>>i;

cout<<i<<endl;

}

}

【解答】

D:\f1.txt：

1 2 3 4 5 6 7 8 9

屏幕显示：

2

4

6

8

9

5．以下程序使用了习题11.2第4小题中生成的文件D:\f1.txt。写出程序运行后屏幕显示的结果。

#include<iostream>

#include<fstream>

using namespace std;

int main()

{

int i;

ifstream f1( "d:\\f1.txt", ios::in );

fstream f2;

f2.open( "d:\\f2.dat", ios::out|ios::binary );

while(!f1.eof())

{

f1>>i;

i = i\*5;

f2.write( ( char\* ) &i, sizeof( int ) );

}

f1.close();

f2.close();

f2.open( "d:\\f2.dat", ios::in|ios::binary );

do

{

f2.read( ( char\* ) &i, sizeof( int ) );

cout<<i<<" ";

}while( i<30 );

cout<<endl;

f2.close();

}

【解答】

5 10 15 20 25 30

#### 11.3 思考题

1．在Visual C++中，流类库的作用是什么？有人说，cin是键盘，cout是显示器，这种说法正确吗？为什么？

【解答】

在Visual C++中，流类库是一个程序包，作用是实现对象之间的数据交互。“cin是键盘，cout是显示器”的说法不正确。cin和cout分别是istream和ostream的预定义对象，默认连接标准设备键盘、显示器，解释从键盘接受的信息，传送到内存；把内存的信息解释传送到显示器。所以称为标准流对象。程序可以对cin、cout重定向，连接到用户指定的设备，例如指定的磁盘文件。

2．什么叫文件？C++读/写文件需要通过什么对象？有些什么基本操作步骤？

【解答】

任何一个应用程序运行，都要利用内存储器存放数据。这些数据在程序运行结束之后就会消失。为了永久的保存大量数据，计算机用外存储器（如磁盘和磁带）保存数据。各种计算机应用系统通常把一些相关信息组织起来保存在外存储器中，并用一个名字（称为文件名）加以标识，称为文件。

C++读/写文件需要用到文件流对象。

文件操作的三个主要步骤是：打开文件、读/写文件、关闭文件流。

打开文件包括建立文件流对象，与外部文件关联，指定文件的打开方式。

读/写文件是按文件信息规格、数据形式与内存交互数据的过程。

关闭文件包括把缓冲区数据完整地写入文件，添加文件结束表示符，切断流对象和外部文件的连接。

3．一个已经建立的文本文件可以用二进制代码方式打开操作吗？一个二进制数据文件可以用文本方式打开吗？为什么？写一个程序试一试。

【解答】

一个已经建立的文本文件可以用二进制方式打开操作。但必须以字符类型数据读取数据然后转换成需要的类型数据才有意义。通常一个二进制文件用文本方式打开是没有意义的，除非这个二进制文件全部是用字符类型数据建立的。因为文本文件是以可读形式ASC码存放数据的，二进制文件直接用计算机表示数据的二进制形式存放数据，它们之间解释方式不同。

程序略。

#### 11.4 编程题

1．以表格形式输出当*x* = 1°，2°，…，10°时sin*x*、cos*x*和tan*x*的值。要求：输出时，数据的宽度为10，左对齐，保留小数点后5位。

【解答】

#include <iostream>

#include <cmath>

#include <iomanip>

using namespace std;

int main()

{

int x; double a;

cout << "x sin(x) cos(x) tg(x)" << endl; //输出表头

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

{

a = x \* 3.14159265 / 180; //角度转换为弧度

cout << setw(3) << setiosflags( ios::left );

cout << setiosflags( ios::fixed );

cout << setprecision(5);

cout << x;

cout << setw(10) << sin(a);

cout << setw(10) << cos(a);

cout<<setw(10)<<sin(a)/cos(a)<<endl;

}

}

2. 建立一个文本文件，从键盘输入一篇短文存放在文件中。短文由若干行构成，每行不超过80个字符。

【解答】

#include <iostream>

#include <fstream>

using namespace std;

int main()

{

char filename[20];

fstream outfile;

cout << "Please input the name of file :\n";

cin >> filename ;

outfile.open( filename, ios::out );

if ( !outfile )

{

cerr << "File could not be open." << endl;

abort();

}

outfile << "This is a file of students\n";

outfile << "Input the number, name, and score.\n";

outfile << "Enter Ctrl-Z to end input? ";

outfile.close();

}

3．读出由习题11.4第2小题建立的文本文件，显示在屏幕上并统计该文件的行数。

【解答】

#include <iostream>

#include <fstream>

using namespace std;

int main()

{

char filename[20];

fstream infile;

cout << "Please input the name of file :\n";

cin >> filename ;

infile.open( filename, ios::in );

if ( !infile )

{

cerr << "File could not be open." << endl;

abort();

}

char textline[80];

int i = 0;

while ( !infile.eof() )

{

infile.getline( textline,sizeof( textline ));

cout << textline << endl;

++i;

}

infile.close();

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

}

4．读出一个作业cpp文件，删除全部注释内容，即以“/\*……\*/”相括的文本和以“//”开始到行末的文本，生成一个新的cpp文件。

【解答】

略。

5．建立某单位职工通讯录的二进制数据文件，文件中的每个记录包括：职工编号、姓名、电话号码、邮政编码和住址。

【解答】

#include <iostream>

#include<fstream>

using namespace std;

struct txrec

{

char no[6];

char name[20];

char tel[9];

char postc[7];

char addr[30];

};

int main()

{

int n,i;

txrec gzrec;

char filename[20];

fstream outfile;

cout << "请输入通讯录文件名:" ;

cin >> filename ;

outfile.open( filename, ios::out|ios::binary );

if ( !outfile )

{

cerr << "文件不能打开！" << endl ;

abort();

}

cout << "请输入职工人数:" ;

cin >> n;

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

{

cout << "请输入第"<< i <<"个职工的编号:" ;

cin >> gzrec.no ;

cout << "请输入第"<< i <<"个职工的姓名:" ;

cin >> gzrec.name ;

cout << "请输入第"<< i <<"个职工的电话号码:" ;

cin >> gzrec.tel ;

cout << "请输入第"<< i <<"个职工的邮政编码:" ;

cin >> gzrec.postc ;

cout << "请输入第"<< i <<"个职工的通信地址:" ;

cin >> gzrec.addr ;

outfile.write( ( char\* )&gzrec,sizeof( txrec )) ;

}

outfile.close() ;

}

6．从键盘输入职工的编号，在由习题11.4第5小题所建立的通讯录文件中查找该职工资料。查找成功后，显示职工的姓名、电话号码、邮政编码和住址。

【解答】

#include <iostream>

#include<fstream>

using namespace std;

struct txrec

{

char no[6];

char name[20];

char tel[9];

char postc[7];

char addr[30];

};

int main()

{

struct txrec gzrec; int i;

char filename[20], num[6];

fstream infile;

cout << "请输入通讯录文件名:";

cin >> filename ;

infile.open( filename, ios::in|ios::binary );

if ( !infile )

{

cerr << "文件不能打开！" << endl;

abort();

}

infile.seekg( 0,ios::end );

long posend = infile.tellp();

infile.seekg( 0,ios::beg );

cout << "请输入职工编号:" ;

cin >> num;

do

{

infile.read(( char \* )&gzrec,sizeof( txrec ));

} while ( strcmp( gzrec.no,num ) != 0 && long(infile.tellp()) != posend );

if ( strcmp( gzrec.no,num ) == 0 )

{

cout << "该职工的记录找到了！" << endl;

cout << "编号:" << gzrec.no << endl;

cout << "姓名:"<< gzrec.name << endl;

cout << "电话号码:"<< gzrec.tel << endl;

cout << "邮政编码:" << gzrec.postc << endl;

cout << "通信地址:" << gzrec.addr << endl;

}

else

{

cout << "该职工的记录找不到！" << endl;

}

infile.close();

}

7．设有两个按升序排列的二进制数据文件f和g，将它们合并生成一个新的升序二进制数据文件h。

【解答】

#include <iostream>

#include<fstream>

using namespace std;

int main()

{

int data1,data2;

fstream infile1,infile2,outfile;

infile1.open( "d:\\vc\\f.dat", ios::in|ios::binary);

if ( !infile1 )

{

cerr << "文件不能打开！" << endl ;

abort();

}

infile1.seekg(0,ios::end);

long posend1 = infile1.tellp();

infile2.open( "d:\\vc\\g.dat", ios::in|ios::binary);

if ( !infile2 )

{

cerr << "文件不能打开！" << endl ;

abort();

}

infile2.seekg( 0,ios::end );

long posend2 = infile2.tellp();

outfile.open( "d:\\vc\\h.dat", ios::out|ios::binary );

infile1.seekg( 0,ios::beg );

infile2.seekg( 0,ios::beg );

while ( long(infile1.tellp()) != posend1 && long(infile2.tellp()) != posend2 )

{

infile1.read(( char \* )&data1,sizeof( int ));

infile2.read(( char \* )&data2,sizeof( int ));

if( data1<data2 )

{

outfile.write(( char \* )&data1,sizeof( int ));

infile2.seekg(-int(sizeof( int )),ios::cur );

}

else

{

outfile.write(( char \* )&data2,sizeof( int ));

infile1.seekg( -int(sizeof(int)),ios::cur );

}

}

while ( long(infile1.tellp()) != posend1 )

{

infile1.read(( char \* )&data1,sizeof( int ));

outfile.write(( char \* )&data1,sizeof( int ));

}

while ( long(infile2.tellp()) != posend2 )

{

infile2.read(( char \* )&data2,sizeof( int ));

outfile.write(( char \* )&data2,sizeof( int ));

}

cout << "文件合并已完成！" << endl;

infile1.close();

infile2.close();

outfile.close();

}

8．编写一个函数，使用数据文件测试在第10章习题10.4第3小题完成的T\_Counter类体系。准备一个文件inputdat用于输入数据，把程序运行结果显示在屏幕上并写入文件outputdat中。

【解答】

略。

## 第12章 异常处理

#### 12.1 阅读下列程序，写出运行结果

1．

#include<iostream>

using namespace std;

int a[ 10 ] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

int fun( int i );

int main()

{

int i ,s = 0;

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

{

try

{

s = s + fun( i );

}

catch( int )

{

cout<<"数组下标越界！"<<endl;

}

}

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

}

int fun( int i )

{

if ( i >= 10 )

throw i;

return a[i];

}

【解答】

数组下标越界!

S=55

2．

#include<iostream>

using namespace std;

void f();

class T

{

public:

T()

{

cout<<"constructor"<<endl;

try

{

throw "exception";

}

catch( char \* )

{

cout<<"exception1"<<endl;

}

throw "exception";

}

~T()

{

cout<<"destructor";

}

};

int main()

{

cout<<"main function "<<endl;

try

{

f();

}

catch( char \* )

{

cout<<"exception2"<<endl;

}

cout<<"main function "<<endl;

}

void f()

{

T t;

}

【解答】

main function

constructor

exception1

exception2

main function

#### 12.2 思考题

1．对一个应用是否一定要设计异常处理程序？异常处理的作用是什么？

【解答】

一个应用不一定要设计异常处理程序。异常处理以结构化思想把异常检测与异常处理分离，增加了程序的可读性，便于大型软件的开发。

2．什么叫抛出异常？catch可以获取什么异常参数？是根据异常参数的类型还是根据参数的值处理异常？请编写测试程序验证。

【解答】

C++异常处理通过三个关键字实现：throw、try和catch。被调用函数按指定条件检测到异常条件的存在，用throw一个数值，称为抛出一个异常。这个函数仅仅做了throw，而不去处理错误。在上层调用函数中使用try语句检测函数调用是否引发异常，被检测到的各种异常由catch语句捕获并作相应的处理。catch只是根据异常参数的类型（不管具体数值）处理异常。

3．什么是不唤醒机制？这种机制有什么好处？请举例说明。

【解答】

不唤醒机制是指抛出异常后，调用链上的所有模块都终止执行，不返回异常抛出点。这种机制的好处是把函数的正常功能设计和异常处理设计分离，便于结构化处理。

程序略。

#### 12.3 编程题

1．从键盘上输入*x*和*y*的值，计算*y* = ln( 2*x* – *y* ) 的值，要求用异常处理“负数求对数”的情况。

【解答】

#include <iostream>

#include <cmath>

using namespace std;

double f( double x,double y );

int main()

{

double x,y;

try

{

cout << "输入x和y的值：";

cin >> x >> y;

cout << f( x,y ) << endl;

}

catch( char \* )

{

cout << "负数不能求对数！" << endl;

}

}

double f( double x,double y )

{

if( 2\*x-y < 0 )

throw "error";

else

return log( 2\*x - y );

}

2．程序中，典型的异常有：内存不足以满足new的请求、数组下标越界、运算溢出、除数为0或无效函数参数等。简单描述程序应该如何用异常处理的方法处理这些情况。

【解答】

略。

3．把12.2.4节中的代码补充成完整的测试程序并运行。

【解答】

略。