# 勘误

### 2.6习题 一、选择题 第3题，选项D需要修改为：

A．goto B．Student C．123 D．2kld

### 2.6习题二、程序阅读题 第2题代码修改如下：

代码要修改为：

#include <iostream>

using namespace std;

int main()

{

int a = 6, b = 6;

if (a > 5)

{

a -= 1;

b += 1;

}

else

{

a += 1;

b -= 1;

}

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

cout << "b=" << b;

}

### 4.8习题 一、单项选择题 第2题 修改如下：

2．在下面循环语句中循环体执行的次数为（）。

int n=4;

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

if(i>n/2) break;

A．*1* B．*2* C．*3* D．*4*

### 11.8习题 一、单项选择题 第3题 选项C修改如下：

A．构造函数B．虚函数C．静态成员函数 D．友元函数

# 答案

## 1.5 习题

一、单项选择题

1.A

解析：C++程序源文件名通常以.cpp、.cxx或.C扩展名结尾。

2.C

解析：C++程序编译后的目标文件以.obj结尾。

3.B

解析：C++目标文件链接的可执行文件以.exe结尾。

4.B

解析：编写C++程序一般需要经过编辑、编译、链接和调试四个步骤。

5.C

解析：C++程序的基本单位为函数。

6.A

解析：程序的主函数名称为main。

7.D

解析：C++程序的基本模块为函数。

## 2.6 习题

一、单项选择题

1.B

解析：单行注释是以“//”开始，到本行结尾处结束。多行注释是以“/\*”开始，“\*/”结束。

2.A

解析：标识符命名规范为：以大写字母、小写字母或者下划线（\_）开始，不能以数字和数字之外的标点符号开始。可以以大写字母、小写字母、下划线（\_）或者数字（0~9）组成。大写字母和小写字母代表不同的标识符。不能使用C++关键字命名标识符。

3.B

解析：标识符命名规范为：以大写字母、小写字母或者下划线（\_）开始，不能以数字和数字之外的标点符号开始。可以以大写字母、小写字母、下划线（\_）或者数字（0~9）组成。大写字母和小写字母代表不同的标识符。不能使用C++关键字命名标识符。

4.B

C++语言的语句以英文分号（;）为结束符。

二、程序阅读题

1.

输出结果为：

c1=4

c2=5

解析：

“int a(2), b(3), c(4);”定义了变量a、b和c。他们的初始值分别为2，3，4。

“cout << "c1=" << c << endl;”输出变量c的值，其中“c1=”原样输出，所以输出结果为“c1=4”。

“c = a + b;”将变量a和b的值相加并赋值给变量c。变量c的值为5。

“cout << "c2=" << c << endl;”输出变量c的值，其中“c2=”原样输出，所以输出结果为“c2=5”。

2.

输出结果为：

a=5

b=7

解析：

“int a = 6, b = 6;”定义变量a和b，他们的初始值分别为6和6。

“if (a > 5)”判断a的值是否大于5，结果为真，所以执行if下的语句块。

“{a-= 1;b += 1;}”由于a大于5为真，所以执行该语句块。“a-= 1”会让变量a减1，所以a的值为5。“b += 1”会让变量b加1，所以b的值为7。

“else{a += 1;b -= 1;}”由于“a > 5”的值为真，所以不执行else分支的语句块。

“cout << "a=" << a << endl;” 输出变量c的值，其中“a=”原样输出，所以输出结果为“a=5”。

“cout << "b=" << b;” 输出变量c的值，其中“b=”原样输出，所以输出结果为“b=7”。

## 3.10 习题

一、单项选择题

1.D

解析：C++语言中基础数据类型包括整型、字符型、实型以及逻辑型。

2.B

解析：字符串常量需要用英文双引号引导。

3.D

解析：(m = a==b)先执行a==b，结果为1，再将1赋值给m。所以m的值为1。(n = c==d)先执行c==d，结果为1，再将1赋值给n。所以n的值为1。

4.C

解析：a += a \* a先执行a\*a结果为144，然后执行a+144，结果为156，最后执行a=156，所以a的值为156。

5.D

解析：程序运行中需要从键盘上输入多个数据时，各数据之间应使用空格或回车符号作为分隔符。

二、程序阅读题

1.

输出结果为：

0

2

4

解析：

int a,b声明了两个整型变量a和b

for (a=0,b=5;a<=b+1;a+=2,b--)for循环语句的条件。初始条件a=0,b=5。终止循环条件a<=b+1。迭代条件a+=2,b--。

cout<<a<<endl;每次循环输出a的值。

第1次循环：a的值为0，0<=b+1条件为真，输出a的值为0。执行a+=2,b--，a的值为2，b的值为4。

第2次循环：a的值为2，2<=b+1条件为真，输出a的值为2。执行a+=2,b--，a的值为4，b的值为3。

第3次循环：a的值为4，4<=b+1条件为真，输出a的值为4。执行a+=2,b--，a的值为6，b的值为2。

第4次循环：a的值为6，6<=b+1条件为假，跳出for循环，结束程序。

## 4.8 习题

一、单项选择题

1.C

解析：i是从0开始，所以，值为0时要进行一次循环。判断条件i<n，使用的是小于号，所以循环次数为n次。例如，i=0;i<2。此时，会循环2次。

2.D

解析：这道题中有两个判断条件可以决定，循环体的循环次数。其中“i<n”的条件，满足该条件的i值包括0、1、2、3，所以，该判断条件可以让循环体循环4次。而另外一个判断条件“if(i>n/2) break;”，满足该条件的i值也包括0、1、2、3，所以，该判断条件也可以让循环体循环4次。综合两个条件，可以判断，这道题的答案为D，循环体会执行4次。

3.A

解析：如果不确定循环的次数时都选择while循环。for循环一般都是在次数固定，条件明确的情况下使用。

4.C

解析：do…while循环的特点就是先执行一遍循环体才会去执行判断条件。

5.B

解析，当分if语句分支过多时，就可以考虑使用switch语句进行替代。所以，switch语句可以改写为if语句。而循环语句的特点是循环体要一致，不能实现多分支的执行。

二、程序阅读题

1.

输出结果为：

10

解析：

int i,j,k;声明了变量I，j和k。

for (i=0,j=10;i<=j;i++,j--)初始条件为i=0,j=10，判断条件为i<=j，迭代条件为i++,j--

k=i+j;循环体

cout<<k;循环体外输出最终的k值。

第1次循环：i的值为0，j的值为10，i<=j条件为真。执行k=i+j，k的值为10。执行i++,j--，i的值为1，j的值为9。

第2次循环：i的值为1，j的值为9，i<=j条件为真。执行k=i+j，k的值为10。执行i++,j--，i的值为2，j的值为8。

第3次循环：i的值为2，j的值为8，i<=j条件为真。执行k=i+j，k的值为10。执行i++,j--，i的值为3，j的值为7。

第4次循环：i的值为3，j的值为7，i<=j条件为真。执行k=i+j，k的值为10。执行i++,j--，i的值为4，j的值为6。

第5次循环：i的值为4，j的值为6，i<=j条件为真。执行k=i+j，k的值为10。执行i++,j--，i的值为5，j的值为5。

第6次循环：i的值为5，j的值为5，i<=j条件为真。执行k=i+j，k的值为10。执行i++,j--，i的值为6，j的值为4。

第7次循环：i的值为6，j的值为4，i<=j条件为假，跳出for循环，输出k的值，结束程序。

2.

输出结果为：

please input a number:5

a是奇数

解析：

int a;声明变量a

cout << "please input a number:";输出字符串“please input a number:”后等待用户输入数字。

cin >> a;将用户输入的数子存放到变量a中。

switch (a % 2)计算a % 2的值。

case 0: cout << "a是偶数" << endl;如果a % 2的值为0就输出字符串“a是偶数”。

default: cout << "a是奇数" << endl; 如果a % 2的值不为0就输出字符串“a是奇数”。

## 5.7 习题

一、单项选择题

1.C

解析：选项C没有指定数组长度

2.D

解析：选项D没有为单个字符添加花括号。

3.C

解析：定义二维数组时可以不指定第一维的长度，但是第二位的长度不可以省略。选项A省略了第二个维度。选项B没有通过初始化确定第一个维度的值，所以不能选。选项D的写法就是错的。

4.C

解析：数组变量a的排列如表所示。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 二维索引  一维索引 | 0 | 1 | 2 | 3 |
| 0 | 3 | 4 | 0 | 0 |
| 1 | 2 | 8 | 6 | 0 |
| 2 | 0 | 0 | 0 | 0 |

5.A

解析：数组变量a的排列如表所示。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 二维索引  一维索引 | 0 | 1 | 2 | 3 |
| 0 | 3 | 4 | 0 | 0 |
| 1 | 2 | 8 | 6 | 0 |
| 2 | 0 | 0 | 0 | 0 |

二、程序设计题

1.

代码如下所示。

#include <iostream>

using namespace std;

#include <math.h> //调用数学运算函数库

int main()

{

int num = 100; //最小的三位数

do

{

//num%10提取num的个位数；num/10%10提取num的十位数；num/100提取num的百位数；

if (pow(num % 10, 3) + pow(num / 10 % 10, 3) + pow(num / 100, 3) == num)

//pow（主体，幂的次方）

//对应个位数的三次方+十位数的三次方+百位数的三次方

{

cout << num << endl; //输出水仙花数

}

num++; //进行num=num+1

} while (num< 1000); //判断语句，判断是三位数

}

运行结果如下所示

153

370

371

407

解析：水仙花数要求如下所示：

第一：是三位数，所以其取值范围为100到999。最小值通过初始化确定。最大值通过比较运算符确定。

第二：该数字的个位的立方+十位的立方+百位的立方刚好等于该值。、

首先需要提取一个三位数的个位，十位和百位。假设要判断的数为num。那么num%10提取num的个位数；num/10%10提取num的十位数；num/100提取num的百位数。

求值的立方可以直接用数学函数pow(数字，次幂)。求num的三次方就是pow(num,3)。

判断数字每位的立方之和是否等于该数字的代码为：pow(num % 10, 3) + pow(num / 10 % 10, 3) + pow(num / 100, 3) == num。最后通过循环，依次判断100到999的数字是否为水仙花数即可。

2.

代码如下所示。

#include <iostream>

using namespace std;

int main()

{

int i, j, factorial, sum = 0; //初始化变量

for (i = 1; i <= 8; i++) //控制数字1到8

{

factorial = 1;

for (j = 1; j <= i; j++) //控制阶乘

{

factorial \*= j; //求阶乘

}

sum += factorial; //将两个阶乘的值相加

}

cout << sum << endl; //输出水仙花数

}

运行结果如下所示

46233

解析：求1到8的阶乘，需要使用两个for循环进行嵌套。外层循环用于控制数字为1到8。内层for循环用于求每个数字的阶乘，最后通过累加得到1到8阶乘的和。

## 6.10 习题

一、单向选择题

1.B

解析：double\*定义了函数的返回值类型为指针变量，返回值的为实数的地址值。所以选B。

2.B

解析：C++的程序都是从main函数开始到main函数结束。所以，选项A错误。函数可以没有返回值所以选项C错误。有调用的函数可以放在不同的文件中所以选项D错误。

3.C

解析：函数可以没有返回值。

4.A

解析：函数重载的函数名必须相同，参数的个数以及类型可以不同。

5.C

解析：函数模板不是具体的函数所以选项A错误。类型参数是可以根据需求设置参数的类型，与普通的参数类型不同所以选项B错误。函数模板定义的函数是需要确定具体的数据类型的，所以选项D错误。

二、程序设计题

1.代码如下所示

#include <iostream>

using namespace std;

#include <math.h> //调用数学运算函数库

void flower()

{

int num = 100; //最小的三位数

do

{

//num%10提取num的个位数；num/10%10提取num的十位数；num/100提取num的百位数；

if (pow(num % 10, 3) + pow(num / 10 % 10, 3) + pow(num / 100, 3) == num)

//pow（主体，幂的次方）

//对应个位数的三次方+十位数的三次方+百位数的三次方

{

cout << num << endl; //输出水仙花数

}

num++; //进行num=num+1

} while (num < 1000); //判断语句，判断是三位数

}

int main()

{

flower(); //调用水仙花函数

}

解析：只需要将水仙花的代码放在函数中，然后在main函数中直接调用该函数即可。

2.

代码如下所示。

#include <iostream>

using namespace std

void Day(int i,int j) //定义函数

{

int n=0; //天数

switch (j) //根据月份确定当月的天数

{

case 1: n = 31;

break;

case 2:

{

if (((i % 4 == 0) && (i % 100 != 00)) || (i % 400 == 0)) n = 29; //判断是否为闰年

else n = 28;

}

break;

case 3: n = 31;

break;

case 4: n = 30;

break;

case 5: n = 31;

break;

case 6: n = 30;

break;

case 7: n = 31;

break;

case 8: n = 31;

break;

case 9: n = 30;

break;

case 10: n = 31;

break;

case 11: n = 30;

break;

case 12: n = 31;

break;

}

cout << "该月的天数为" << n << endl; //输出天数

}

int main()

{

int i, j, n;

cout << "请输入年" << endl;

cin >>i; //获取年

cout << "请输入月" << endl;

cin >> j; //获取月

Day(i,j); //调用函数Day

return 0;

}

运行结果如下所示。

请输入年

2000

请输入月

2

该月的天数为29

解析：1、3、5、7、8、10、12月，每月31天；4、6、9、11月，每月30天；2月闰年29天，平年28天。对于多分支情况使用了switch语句实现。

其中，最特殊的是2月的天数要根据年来确定。闰年判断方法是：如果年号能被400整除，此年为闰年，如果年号能被4整除，而不能被100整除，此年为闰年；否则不是闰年，对应代码为“if (((i % 4 == 0) && (i % 100 != 00)) || (i % 400 == 0))”。

## 7.8 习题

一、单项选择题

1.C

解析：函数定义是可以定义形参为引用，在调用时使用实参进行调用。其它选项的都不是引用调用。

2.C

解析：除了选项C其它的选项语法都是错误的。选项A种new后面不能跟变量名，选项B中两侧的数据类型不符，选项D中不能连等对数组进行初始化。

3.C

解析：sizeof()函数求的是指定数据所占字节数。array和array[0]都是数组的第一个字节的地址，所以会返回数组中第一个字节在内存中字节数，也就是数组中每个元素所占的字节数。

4.A

解析：指针变量s存放的值是字符的首地址值。所以只需要将值直接赋值给同类型的指针变量p即可。选项B会把指针变量s的地址值赋值给p，而不是字符串的地址值。选项C传递的是字符串而不是地址值，相当于为p复制了一个新的字符串。p中的地址值与s中的地址值不同，所以指向的不是同一个字符串。选项D和选项B相同。

5.D

解析：x=\*pa+3相当于取了数组a的第一个元素的值，然后加3进行运算，最后赋值给变量x。

二、程序设计题

1.

代码如下所示。

#include<iostream>

using namespace std;

int main()

{

int i, n;

cin >> n;

cout << "n=" << "1"; //1是所有数字的第一个因子直接输出

for (i = 2; i <= n;) //根据输入的数控制循环次数

{

if (n % i == 0) //判断i是否为n因子

{

n = n / i; //修改n的值

cout << "\*" << i; //每次循环输出一个因子

continue;

}

i++;

}

cout << endl;

return 0;

}

运行结果如下所示。

12

n=1\*2\*2\*3

解析：整数a除以整数b(b≠0) 的商正好是整数而没有余数，我们就说b是a的因子。0不是0的因子。所以，首先需要判断n除以i是否有余数，从而判断i是否为n的因子。如果i是n的因子，此时要保留n除以i的商，然后再次用n除以i寻找因子。依次类推，一直到因子大于被除数为止。

2.

代码如下所示。

#include<iostream>

using namespace std;

int main()

{

double a = 1.0, sum = 0, pi;

int f = 1;

for (a = 1; (1.0/a)>0.000001; a += 2)//判断每一项是否小于0.000001，让除数通过累加变为奇数

{

sum += f \* 1.0 / a; //计算每一项的值

f = -f; //切换每一项的符号

}

pi = sum \* 4.0; //求出等式的值后乘以4得到π的值

cout << "pi="<<pi<<endl; //输出π的值

return 0;

}

运行结果如下所示。

pi=3.14159

解析：公式π/4=1-1/3+1/5-1/7+1/9…….中，将第1项看做1/1之后，每一项都是一个除法运算。除数为1和负1交替出现，被除数全部为奇数。所以使用f \* 1.0和f = -f;实现正负1的切换，使用a+=2迭代实现奇数依次出现。题目要求最后一项的值要小于0.000001，所以使用1.0/a>0.000001控制循环的次数。当最后一项的值小于等于0.000001后结束循环。

## 8.8 习题

一、单项选择题

1.A

解析：变量book为结构体指针变量，在访问成员变量是需要使用结构指向运算符“->”实现。或者使用间接运算符“\*”、一对小括号“()”和点成员运算符“.”访问成员变量。选项B直接使用了点成员运算符所以错误。选项C只使用间接运算符所以错误。选项D应该使用点成员运算符，但是使用了结构指向运算符所以错误。

2.A

解析：选项A中book小写，所以错误。

3.B

解析：选项B是为结构体变量book定义一个引用x。选项A是将地址值赋值给地址值是错误的。选项C语法错误。选项D无法为结构体直接定义引用，只能为结构体的变量定义引用。

二、程序设计题

1.

代码如下所示。

#include<iostream>

using namespace std;

struct Object { //定义结构体Object

int id; //id

string name; //姓名

}student; //声明变量student

int main()

{

cout << "请输出id" << endl;

cin >> student.id;

cout << "请输入姓名" << endl;

cin >> student.name;

cout << "用户输入的id为：" << student.id<< "，姓名为：" << student.name<<endl;

return 0;

}

运行结果如下所示。

请输出id

1

请输入姓名

张三

用户输入的id为：1，姓名为：张三

解析：定义结构体Object时需要添加id和name两个成员变量。在定义结构体的同时声明了结构体变量student。在主函数中通过键盘输入将id和姓名存放到student的成员变量中，最后进行输出。

2.

代码如下所示。

#include<iostream>

using namespace std;

struct Bank{ //定义结构体Object

string account; //账号

string IDcard; //身份证号

string name; //姓名

string address; //住址

double amount; //金额

}man; //声明变量man

int main()

{

cout << "请输入账号" << endl;

cin >> man.account;

cout << "请输入身份证号" << endl;

cin >> man.IDcard;

cout << "请输入姓名" << endl;

cin >> man.name;

cout << "请输入住址" << endl;

cin >> man.address;

cout << "请输入存款金额" << endl;

cin >> man.amount;

return 0;

}

解析：定义结构体Bank时需要添加五个成员变量。在定义结构体的同时声明了结构体变量man。在主函数中通过键盘输入将不同信息存放到对应的成员变量中。

## 9.8 习题

一、单项选择题

1.A

解析：面向对象分析方法（Object-Oriented Analysis，OOA）

2.B

解析：面向对象设计（Object-Oriented Design，OOD）

3.B

解析：尽管问题域中的事物是很复杂的，但是分析时并不需要了解和描述它们的一切，只需要分析研究其中与系统目标有关的事物及其本质性特征。

4.C

解析：类的定义中可以包含数据成员和函数成员。

5.B

解析：在类作用域中能够通过直接使用该类的公用成员名进行访问。

二、程序设计题

代码如下所示。

#include<iostream>

#include<string>

using std::cout;

using std::cin;

using std::endl;

using std::string;

class Book

{

//构造函数时从调用函数传过来参数，所以用pass的首字母p表示接受传参

public: Book(string pName, double pPrice, int pNumber);

void display(); //显示图书情况

void borrow(); //借书

void restore(); //还书

private: string name; //书名

double price; //价格

int number; //数量

};

Book::Book(string pName, double pPrice, int pNumber)

{

name = pName;

price = pPrice;

number = pNumber;

}

void Book::display()

{

cout << "书名" << name << "价格为：" << price << "目前拥有" << number << "本" << endl;

}

void Book::borrow()

{

number -= 1; //借书数量减1

cout << "目前拥有" << number << "本书。" << endl;

}

void Book::restore()

{

number = number + 1; //还书数量加1

cout << "目前拥有" << number << "本书" << endl;

}

int main()

{

Book book("四个苹果", 99.00, 6);

int choice; cout << "1-->展示所有图书信息" << endl;

cout << "2-->借书" << endl; cout << "3-->还书" << endl;

int contin = 1;

while(contin)

{

cout << "请输入你想干什么：";

cin >> choice;

switch(choice)

{

case 1:book.display();break;

case 2:book.borrow();break;

case 3:book.restore();break;

default:cout << "输入错误!";break;

}

cout <<"是否继续? 1-yes/0-no: ";

cin >> contin;

}

return 0;

}

运行结果如下所示。

1-->展示所有图书信息

2-->借书

3-->还书

请输入你想干什么：1

书名四个苹果价格为99目前拥有6本

是否继续? 1-yes/0-no: 1

请输入你想干什么：2

目前拥有5本书。

是否继续? 1-yes/0-no: 1

请输入你想干什么：3

目前拥有6本书

是否继续? 1-yes/0-no: 0

解析：使用类Book定义图书借阅系统，在类中通过成员变量实现书名、价格、数量等信息的定义，在通过类的方法实现查看图书信息，借书以及还书操作。最后在main函数中初始化图书馆的信息，在通过分支语句实现用户和系统的交互。

## 10.5 习题

1.B

解析：成员函数重载一元运算符时可以没有任何参数。

2.B

解析：成员函数重载双目元运算符时需要带有1个参数。

3.C

解析：非成员函数重载双目元运算符时需要带有2个参数。

4.A

解析：选项B中a和b写反了。选项C是分成员函数重载。选项D语法错误。

5.C

解析：选项A和选项B是成员函数重载。选项D写反了。

二、填空题

1. this指针指向的当前对象 成员函数参数

解析：利用成员函数对二元运算符重载,其左操作数为this指针指向的当前对象,右操作数为成员函数参数。

2. operator

解析：运算符函数中的关键字是operator，它和运算符一起组成该运算符函数的函数名。

三、程序设计题

1.

代码如下所示。

#include <iostream>

using namespace std;

class Money {

private:

int yuan; // 元

int jiao; // 角

int fen; // 分

public:

Money(int y = 0, int j = 0, int f = 0) : yuan(y), jiao(j), fen(f) {}

friend ostream& operator<<(ostream& os, const Money& m); // 友元函数，重载输出运算符

friend Money operator+(const Money& m1, const Money& m2); // 友元函数，重载加法运算符

friend Money operator-(const Money& m1, const Money& m2); // 友元函数，重载减法运算符

};

ostream& operator<<(ostream& os, const Money& m) {

os << "金额为" << m.yuan << "元" << m.jiao << "角" << m.fen << "分";

return os;

}

Money operator+(const Money& m1, const Money& m2) {

int sum\_fen = (m1.fen + m2.fen) % 10;

int carry\_jiao = (m1.fen + m2.fen) / 10;

int sum\_jiao = (m1.jiao + m2.jiao + carry\_jiao) % 10;

int carry\_yuan = (m1.jiao + m2.jiao + carry\_jiao) / 10;

int sum\_yuan = m1.yuan + m2.yuan + carry\_yuan;

return Money(sum\_yuan, sum\_jiao, sum\_fen);

}

Money operator-(const Money& m1, const Money& m2) {

int sum\_fen = (m1.fen - m2.fen) % 10;

int carry\_jiao = (m1.fen - m2.fen) / 10;

int sum\_jiao = (m1.jiao - m2.jiao - carry\_jiao) % 10;

int carry\_yuan = (m1.jiao - m2.jiao - carry\_jiao) / 10;

int sum\_yuan = m1.yuan - m2.yuan - carry\_yuan;

return Money(sum\_yuan, sum\_jiao, sum\_fen);

}

int main() {

Money m1(2, 3, 4); // 2元3角4分

Money m2(3, 7, 3); // 3元7角3分

Money m3 = m1 + m2; // 金额相加

Money m4 = m1 - m2; // 金额相减

cout << m1 << endl;

cout << m2 << endl;

cout << m3 << endl; //和

cout << m4<< endl; //差

return 0;

}

运行结果如下所示。

金额为2元3角4分

金额为3元7角3分

金额为6元0角7分

金额为-1元-4角1分

解析：使用类Money定义元，角和分，然后通过函数重载实现加法、减法和输出运算符的重载，最后在main函数中初始化对象并输出两个对象的和以及差值。

## 11.8 习题

一、单项选择题

1.D

解析：继承自基类的成员访问权限会发生对应的改变。

2.B

解析：在一个派生类中，其成员由两部分构成：一部分是从基类继承得到的，另一部分是自己定义的新成员。当基类本身也是一个派生类时，它必然包含了它的直接基类成员，这些成员相应地也存在于该基类的派生类中。因此，继承是具有传递性的。

3.A

解析：派生类不能继承基类的构造函数和析构函数

二、填空题

1.对象 析构函数

解析：构造函数是对象被创建时自动执行的，对象消失时自动执行的成员函数称为析构函数。

2. 子类 派生类 基类

解析：在继承机制下，当对象消亡时，编译系统先执行子类的析构函数，然后才执行派生类的析构函数，最后执行基类的析构函数。

3.派生类 基类

解析：如果类A继承了类B，则类A称为派生类，类B称为基类。

三、程序设计题

代码如下所示。

#include <iostream>

#include <string>

using namespace std;

class student

{

protected:

string num, name;

public:

student(string s\_num, string s\_name);

void show();

};

student::student(string s\_num, string s\_name) : num(s\_num), name(s\_name) {}

void student::show()

{

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

}

class student1 : public student

{

private:

int age;

string addr;

student monitor;

public:

student1(string s\_num, string s\_name, int s\_age, string s\_addr, string m\_num, string m\_name);

void show();

};

student1::student1(string s\_num, string s\_name, int s\_age, string s\_addr, string m\_num, string m\_name)

:student(s\_num, s\_name), monitor(m\_num, m\_name)

{//给monitor实例参数，令其调用构造函数

age = s\_age;

addr = s\_addr;

}

void student1::show()

{

cout << "This student is: " << endl;

student::show();//调用其基类的show函数，输出自身的数据

cout << "age: " << age << endl << "addr: " << addr << endl;

cout << "Class monitor is:" << endl;

monitor.show();//调用monitor实例的show函数

}

int main()

{

student1 stud1("10001", "LiMing", 20, "SD", "10001", "LiMing");

stud1.show();

return 0;

}

运行结果如下所示。

This student is:

num: 10001

name: LiMing

age: 20

addr: SD

Class monitor is:

num: 10001

name: LiMing

解析：定义了基类后在定义派生类。在定义派生类时要继承基类的成员。最后在main函数初始化类的对象，并为对象赋值。最后调用成员函数show展示对象的学生信息。

## 12.6 习题

一、单向选择题

1. D

解析：动态联编一直到程序运行时才能确定调用哪个函数。没有虚函数不能够实现动态联编，调用虚函数操作的是指向对象的指针或者对象的引用。

2.C

解析：一个基类的声明中有纯虚函数,该基类的派生类也可以为抽象类;

3. D

解析：带有纯虚函数的类称为抽象类。抽象类中的纯虚函数没有具体的定义，所以不能说明抽象类的对象。这也是抽象类的特性。

二、填空题

1.内联 inline

解析：为了降低函数调用的时间开销，建议将小的、调用频繁的函数定义为内联，方法是在函数类型前加上inline关键字。

2.虚基类

解析：如果一个类中有一个或多个纯虚函数，则这个类称为虚基类。

3.封装

解析：类完成了面向对象程序设计的封装特性。

4.纯虚函数

解析：纯虚函数是一个在基类中说明的虚函数，但未给出具体的实现，要求在其派生类中实现。

三、程序设计题

代码如下所示。

#include<iostream>

using namespace std;

// 定义数学常数pi

#define pi 3.1415926

// 创建抽象基类Area

class Area {

public:

virtual double GetArea() = 0;

};

// 创建抽象基类Volume

class Volume {

public:

virtual double GetVolume() = 0;

};

// 创建基类Circle圆类

class Circle :public Area {

public:

friend class Cylinder;

friend class Cone;

friend class Sphere;

Circle() {}

Circle(double r\_) { r = r\_; }

virtual double GetArea() { return pi \* r \* r; }

private:

double r;

};

// 创建圆柱类

class Cylinder :public Circle, public Volume {

public:

Cylinder() :Circle() {}

Cylinder(double r\_, double h\_) :Circle(r\_) { h = h\_; }

virtual double GetArea() { return 2 \* pi \* r \* r + 2 \* pi \* r \* h; }

virtual double GetVolume() { return pi \* r \* r \* h; }

private:

double h;

};

// 创建圆锥类

class Cone :public Circle, public Volume {

public:

Cone() :Circle() {}

Cone(double r\_, double h\_) :Circle(r\_) { h = h\_; }

virtual double GetArea() { return pi \* r \* r + 2 \* pi \* r \* sqrt(h \* h + r \* r) / 2; }

virtual double GetVolume() { return pi \* r \* r \* h / 3; }

private:

double h;

};

// 创建球类

class Sphere :public Circle, public Volume {

public:

Sphere() :Circle() {}

Sphere(double r\_) :Circle(r\_) {}

virtual double GetArea() { return 4 \* pi \* r \* r; }

virtual double GetVolume() { return 4 \* pi \* r \* r \* r / 3; }

};

int main() {

Circle circle(2);

cout << "创建一个圆对象" << endl;

cout << "半径 ：2.0" << endl;

cout << "面积 ：" << circle.GetArea() << endl << endl;

Cylinder cylinder(2, 3);

cout << "创建一个圆柱对象" << endl;

cout << "半径 ：2.0" << endl;

cout << "高度 ：3.0" << endl;

cout << "面积 ：" << cylinder.GetArea() << endl;

cout << "体积 ：" << cylinder.GetVolume() << endl << endl;

Cone cone(2, 3);

cout << "创建一个圆锥对象" << endl;

cout << "半径 ：2.0" << endl;

cout << "高度 ：3.0" << endl;

cout << "面积 ：" << cone.GetArea() << endl;

cout << "体积 ：" << cone.GetVolume() << endl << endl;

Sphere sphere(2);

cout << "创建一个球对象" << endl;

cout << "半径 ：2.0" << endl;

cout << "面积 ：" << sphere.GetArea() << endl;

cout << "体积 ：" << sphere.GetVolume() << endl << endl;

return 0;

}

运行结果如下所示。

创建一个圆对象

半径 ：2.0

面积 ：12.5664

创建一个圆柱对象

半径 ：2.0

高度 ：3.0

面积 ：62.8319

体积 ：37.6991

创建一个圆锥对象

半径 ：2.0

高度 ：3.0

面积 ：35.2207

体积 ：12.5664

创建一个球对象

半径 ：2.0

面积 ：50.2655

体积 ：33.5103

解析：创建的基类为圆，圆的半径会应用到求对应的圆柱体、圆锥体和球体的面积以及体积。所以，创建好基类后就可以派生出圆柱体、圆锥体和球体三个派生类。最后在main函数中分别使用对应的对象实现面积和体积的输出。

## 13.7 习题

一、单项选择题

1.C

解析：一个类可以有多个对象。

2.C

解析：构造函数的作用就是实现对象的初始化。

3.A

解析：在定义一个派生类时，若不使用保留字显式地规定采用何种继承方式，则默认为私有继承方式。

二、填空题

1.对象 继承

解析：1．类有两种用法：一种是类的实例化，即生成类的对象；另一种是通过继承派生出新的类。

2.不 和数量

解析：函数的形参在未被调用之前不分配空间，函数的形参的类型和数量要与实参的相同。

三、程序设计题

1.

代码如下所示。

#include <iostream>

template<typename T>

//定义模板

void arrayStats(T arr[], int size, T& maxVal, T& minVal, double& avgVal) {

if (size <= 0) {

std::cerr << "出现错误" << std::endl;

return; // 防止除零错误

}

maxVal = arr[0];

minVal = arr[0];

avgVal = arr[0];

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

if (arr[i] > maxVal) {

maxVal = arr[i];

}

if (arr[i] < minVal) {

minVal = arr[i];

}

avgVal += arr[i];

}

avgVal /= size;

}

int main() {

int arr[] = { 1, 2, 3, 4, 5, 6 }; //初始值

int maxVal, minVal;

double avgVal;

arrayStats(arr, 6, maxVal, minVal, avgVal);

std::cout << "最大值: " << maxVal << std::endl;

std::cout << "最小值: " << minVal << std::endl;

std::cout << "平均值: " << avgVal << std::endl;

return 0;

}

运行结果如下所示。

最大值: 6

最小值: 1

平均值: 3.5

解析：函数模板可以接受一个任意类型的数组、数组大小、以及最大值、最小值和平均值的引用参数。在函数内部，使用一个for循环遍历数组元素，分别计算最大值、最小值和平均值，并将结果存储在对应的引用参数中。

2.

代码如下所示。

#include <iostream>

template<typename T, int N>

void swapArray(T(&arr1)[N], T(&arr2)[N]) {

T temp[N];

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

temp[i] = arr1[i];

arr1[i] = arr2[i];

arr2[i] = temp[i];

}

}

int main() {

int arr1[] = { 1, 2, 3, 4, 5 };

int arr2[] = { 6, 7, 8, 9, 10 };

swapArray(arr1, arr2);

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

std::cout << arr1[i] << " ";

}

std::cout << std::endl;

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

std::cout << arr2[i] << " ";

}

std::cout << std::endl;

return 0;

}

运行结果如下所示。

6 7 8 9 10

1 2 3 4 5

解析：函数模板可以接受一个任意类型的数组。在函数内部，使用一个for循环遍历数组元素，并将两个数组之间的数据进行交换，最后在主函数中通过for循环进行了输出。

## 14.5 习题

一、填空题

1. stack queue priority\_queue

解析：三种STL容器适配器是stack、queue和priority\_queue。

2.不独立的 顺序 构造 析构

解析：适配器是不独立的，它依附于一个顺序容器上，它没有自己的构造函数和析构函数，而借用其实现类的对应函数。

二、简答题

1.

答：顺序容器分为**vector**（向量）、**deque**（双队列）、**list**（列表）。

**vector**（向量）属于顺序容器，用于容纳不定长线性序列（即线性群体），提供对序列的快速随机访问（也称直接访问）。数据结构很像一个数组，所以与其他容器相比，vector 能非常方便和高效访问单个元素，支持随机访问迭代子。向量是动态结构，它的大小不固定，可以在程序运行时增加或减少与数组不同，向量的内存用尽时，向量自动分配更大的连续内存区，将原先的元素复制到新的内存区，并释放旧的内存区。这是向量类的优点。

**deque**（双队列）双端队列是一种放松了访问权限的队列。元素可以从队列的两端入队和出队，也支持通过下标操作符“[]”进行直接访问。可以在双端队列起点上的插入和删除操作快

**list**（列表）主要用于存放双向链表，可以从任意一端开始遍历。列表还提供了拼接（splicing）操作，将一个序列中的元素从插入到另一个序列中。元素的插入和删除操作对list而言尤为高效。与vector和deque 相比，对元素的下标访问操作的低效是不能容忍的，因此 list 提供这类操作。

2.

答：迭代器不是指针，是类模板，表现的像指针。他只是模拟了指针的一些功能，通过重载了指针的一些操作符，->、\*、++、–等。迭代器封装了指针，是一个“可遍历STL（ Standard Template Library）容器内全部或部分元素”的对象， 本质是封装了原生指针，是指针概念的一种提升（lift），提供了比指针更高级的行为，相当于一种智能指针，他可以根据不同类型的数据结构来实现不同的++，–等操作。迭代器返回的是对象引用而不是对象的值，所以cout只能输出迭代器使用\*取值后的值而不能直接输出其自身。

三、程序设计题

代码如下所示。

#include <iostream>

#include <list>

#include <algorithm>

using namespace std;

int main()

{

// 初始化两个链表

list<int> list1 = { 10, 15, 2, 8, 22, 11 };

list<int> list2 = { 1, 5, 10, 22, 4, 8, 33, 6 };

// 将list2中的元素插入到list1中

list1.insert(list1.end(), list2.begin(), list2.end());

// 对链表进行排序

list1.sort();

// 删除重复元素

list1.unique();

// 输出结果

for (auto it = list1.begin(); it != list1.end(); ++it)

{

cout << \*it << " ";

}

cout << endl;

return 0;

}

运行结果如下所示。

1 2 4 5 6 8 10 11 15 22 33

解析：首先将两个数组插入到一个链表容器中，然后对列表中的内容进行排序，紧接着删除重复的元素，最后输出排序号的列表。

## 15.7 习题

一、单项选择题

1. D

解析：类的析构函数是在类的对象被释放的时候自动调用，它的作用是删除已经创建完成的对象。

2.C

解析：优先级依次为\*、+、=、>=。

3. D

解析：p是一个指针，用来存储a变量的地址，而100是存储在该地址的一个值用p表示的为-p。

二、填空题

1. this指针指向的当前对象 成员函数参数

解析：利用成员函数对二元运算符重载，其左操作数为this指针指向的当前对象，右操作数为成员函数参数。

2.对象

解析：文件的打开是通过使用类对象的成员函数来实现的。

3．不会

运算符重载后，原运算符的优先级和结合特性不会改变。

三、简答题

1.

答：在C++中，流(stream)是一个抽象的概念，表示数据的流动。流可以分为输入流和输出流两种类型，输入流用于从外部读取数据到程序中，而输出流用于将程序中的数据输出到外部。在C++中，输入流和输出流都是通过流对象来实现的。

流提取和插入是指将数据从流中读取或写入到变量中的操作。流提取是将流中的数据读取到变量中的过程，通常使用输入运算符 ">>" 实现。流插入是将变量的值写入到流中的过程，通常使用输出运算符 "<<" 实现。

I/O流是通过流对象来实现输入输出操作的。I/O流可以与多种类型的设备进行交，I/O流提供了一些成员函数，可以进行更加灵活的输入输出操作。

2.

答：在C++中，cerr和clog都是输出流对象，它们都用于将错误信息输出到标准错误输出设备。它们的区别如下所示：

cerr是一个非缓冲的流对象，而clog是一个缓冲的流对象。

cerr的输出流对象通常用于输出错误信息，而clog的输出流对象通常用于输出一般性的日志信息。

cerr和clog在使用方式上是相同的，都可以使用输出运算符 "<<" 将数据输出到流中。

## 16.6 习题

一、填空题

1.语句 语句

解析：C++程序将可能发生异常的块放在语句中，紧跟其后可放置若干个对应的语句中。

2.调用 定义

解析：throw表达式的行为有些像函数的调用，而catch子句则有些像函数的定义。

二、程序设计题

代码如下所示

#include<iostream>

#include<cmath>

using namespace std;

class line {

private:

double a, b, c;

public:

line() {//默认构造函数

a = 1;

b = 1;

c = 1;

}

line(line& ob) {//拷贝构造函数

a = ob.a;

b = ob.b;

c = ob.c;

}

void set() {

cout << "输入a,b,c的值：";

cin >> a >> b >> c;

try {

if (a == 0 && b == 0)

throw 1;

}

catch (int) {

cout << "发生异常：无法构成直线方程" << endl;

}

}

void display() {

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

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

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

}

friend void setpoint(line& ob1, line& ob2) {

double x, y;

try {

if (ob1.a / ob2.a == ob1.b / ob2.b && ob1.a / ob2.a == ob1.c / ob2.c)

throw 'a';//char

if (ob1.a / ob2.a == ob1.b / ob2.b && ob1.a / ob2.a != ob1.c / ob2.c)

throw 1;//int

x = (ob1.c \* ob2.b - ob2.c \* ob1.b) / (ob2.a \* ob1.b - ob1.a \* ob2.b);

y = (ob1.c \* ob2.a - ob2.c \* ob1.a) / (ob2.b \* ob1.a - ob1.b \* ob2.a);

if (fabs(x) >= pow(10, 8) || fabs(y) >= pow(10, 8))

throw "交点坐标的绝对值大于等于10^8";//const char \*

cout << "交点坐标：(" << x << "," << y << ")" << endl;

}

catch (int) {

cout << "发生异常：两直线平行" << endl;

}

catch (char) {

cout << "发生异常：两直线重合" << endl;

}

catch (const char\*) {

cout << "交点坐标的绝对值大于等于10^8" << endl;

}

}

};

int main(void) {

line ob1, ob2;

ob1.set();

ob1.display();

ob2.set();

ob2.display();

setpoint(ob1, ob2);

}

运行结果如下所示。

输入a,b,c的值：5 5 5

a = 5

b = 5

c = 5

输入a,b,c的值：3 4 5

a = 3

b = 4

c = 5

交点坐标：(1,-2)

运行程序抛出异常结果如下所示。

输入a,b,c的值：0 0 0

发生异常：无法构成直线方程

a = 0

b = 0

c = 0

输入a,b,c的值：1 2 3

a = 1

b = 2

c = 3

发生异常：两直线重合

解析：C++ 异常处理涉及到三个关键字：try、catch、throw。代码中会根据输入的值判断点是否可以构成直线，如果不能就抛出异常，如果可以就输出坐标位置。

## 17.4 习题

一、单项选择题

1.C

解析：．一个C++程序文件的扩展名为.cpp。.h是头文件，.c是C语言，.cp是数据库文件或声音文件。

2.A

解析：C++程序都是从main函数开始到main函数结束。所以，有main函数的文件被称为主文件。

3.D

解析：为了方便管理一般将类的定义存放与头文件中。需要使用对应类时可以引入对应的头文件。

4.A

解析：在多文件结构的程序中,通常把类中所有非内联函数的定义单独存放于主文件中。

5.B

解析：．消息的发送者不一定要完全了解接收者如何响应消息。

二、简答题

答：

C语言：C语言是一种高级编程语言，由Dennis Ritchie于1972年开发。它是一种过程式编程语言，通常用于系统编程、嵌入式开发、科学计算等领域。C语言具有高效、简洁、可移植等特点，是许多编程语言的基础。

C++语言：C++语言是一种基于C语言的面向对象编程语言，由Bjarne Stroustrup于1983年开发。它继承了C语言的许多特性，同时增加了面向对象的特性，如类、继承、多态等。C++语言可以用于开发各种类型的应用程序，包括系统软件、桌面应用程序、网络应用程序等。

VC++（Visual C++）：VC++是一款由Microsoft开发的C++集成开发环境（IDE），用于开发Windows应用程序。它包含了C++编译器、调试器、资源编辑器等工具，支持MFC、ATL、Win32等框架。VC++提供了丰富的工具和库，可以帮助开发者更快速地编写高质量的Windows应用程序。

MFC（Microsoft Foundation Class）：MFC是一组C++类库，用于开发Windows应用程序。它提供了大量的UI控件、数据结构、文件I/O等功能，可以更方便地进行Windows编程。MFC是VC++的一个重要组成部分，开发者可以使用MFC来开发Windows应用程序，从而更加方便地创建丰富的用户界面和功能。

总的来说，C和C++是编程语言，VC++是用于开发Windows应用程序的IDE，MFC是用于Windows应用程序开发的C++类库。开发者可以使用C和C++语言编写各种类型的应用程序，而使用VC++和MFC可以更加方便地进行Windows应用程序的开发。

## 18.5 习题

程序设计题

1.

代码如下所示。

#include <stdio.h>

#include <process.h>

#include <windows.h>

#include<iostream>

#include<cmath>

using namespace std;

HANDLE g\_hThreadEvent1;

HANDLE g\_hThreadEvent2;

int g\_nMainLoopNum = 0;

int g\_nSubLoopNum = 0;

#define LOOPNUM 50

#define MAINLOOPNUM 100

#define SUBLOOPNUM 10

unsigned int \_\_stdcall subThreadFun(void\* pData)

{

while (1) //子线程要一直监听信号量

{

//等待事件2的信号量，期间不往下执行P(event2)

WaitForSingleObject(g\_hThreadEvent2, INFINITE);

g\_nSubLoopNum++; //子线程执行次数

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

{

cout << "子线程编号是" << GetCurrentThreadId() << ",在循环第" << i << "次" << endl;

}

cout << " \*\*\*\*\*\*\*\*\*\*\*\*子线程正在执行第" << g\_nSubLoopNum << "次\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

//设置事件1的信号量，相当于执行主线程 v(event1)

ReleaseSemaphore(g\_hThreadEvent1, 1, NULL);

Sleep(100);

}

return 0;

}

int main()

{

g\_hThreadEvent1 = CreateSemaphore(NULL, 0, 1, NULL);

g\_hThreadEvent2 = CreateSemaphore(NULL, 1, 1, NULL);

\_beginthreadex(NULL, 0, subThreadFun, NULL, 0, NULL); //创建子线程

//相当于main为主线程

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

{

//一直等待线程事件1的信号量p(event1)

WaitForSingleObject(g\_hThreadEvent1, INFINITE);

g\_nMainLoopNum++; //主线程执行次数

for (int num = 1; num <= MAINLOOPNUM; num++)

{

cout << "主线程编号是" << GetCurrentThreadId() << ",在循环第"<< num<<"次" << endl;

}

cout << " \*\*\*\*\*\*\*\*\*\*\*\*主线程正在执行第"<< g\_nMainLoopNum <<"次\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<< endl;

Sleep(100);

if (i != LOOPNUM)

{

//给事件2设置信号量，就是让子线程执行 v(event2)

ReleaseSemaphore(g\_hThreadEvent2, 1, NULL);

}

else

break;

}

CloseHandle(g\_hThreadEvent1);

CloseHandle(g\_hThreadEvent2);

return 0;

}

运行结果如下所示。

子线程编号是 21096 ,在循环第 1 次

子线程编号是 21096 ,在循环第 2 次

子线程编号是 21096 ,在循环第 3 次

子线程编号是 21096 ,在循环第 4 次

子线程编号是 21096 ,在循环第 5 次

子线程编号是 21096 ,在循环第 6 次

子线程编号是 21096 ,在循环第 7 次

子线程编号是 21096 ,在循环第 8 次

子线程编号是 21096 ,在循环第 9 次

子线程编号是 21096 ,在循环第 10 次

\*\*\*\*\*\*\*\*\*\*\*\*子线程正在执行第1次\*\*\*\*\*\*\*\*\*\*\*\*\*\*

主线程编号是 27752 ,在循环第 1 次

主线程编号是 27752 ,在循环第 2 次

主线程编号是 27752 ,在循环第 3 次

主线程编号是 27752 ,在循环第 4 次

主线程编号是 27752 ,在循环第 5 次

主线程编号是 27752 ,在循环第 6 次

主线程编号是 27752 ,在循环第 7 次

主线程编号是 27752 ,在循环第 8 次

主线程编号是 27752 ,在循环第 9 次

……

主线程编号是 27752 ,在循环第 83 次

主线程编号是 27752 ,在循环第 84 次

主线程编号是 27752 ,在循环第 85 次

主线程编号是 27752 ,在循环第 86 次

主线程编号是 27752 ,在循环第 87 次

主线程编号是 27752 ,在循环第 88 次

主线程编号是 27752 ,在循环第 89 次

主线程编号是 27752 ,在循环第 90 次

主线程编号是 27752 ,在循环第 91 次

主线程编号是 27752 ,在循环第 92 次

主线程编号是 27752 ,在循环第 93 次

主线程编号是 27752 ,在循环第 94 次

主线程编号是 27752 ,在循环第 95 次

主线程编号是 27752 ,在循环第 96 次

主线程编号是 27752 ,在循环第 97 次

主线程编号是 27752 ,在循环第 98 次

主线程编号是 27752 ,在循环第 99 次

主线程编号是 27752 ,在循环第 100 次

\*\*\*\*\*\*\*\*\*\*\*\*主线程正在执行第50次\*\*\*\*\*\*\*\*\*\*\*\*\*\*

解析：代码中通过for循环嵌套依次执行对应次数的主线程和子线程。

2.

代码如下所示。

#include <Windows.h>

#include <iostream>

// 事件句柄 分别是 a b c

HANDLE hEvent[3] = { 0 };

DWORD WINAPI ThreadProc(LPVOID lpThreadParameter)

{

int now\_id = (int)lpThreadParameter;

int next\_id = (now\_id + 1) % 3;

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

{

WaitForSingleObject(hEvent[now\_id], -1); // 激发态时才能继续执行

printf("第%d遍:打印%c\n", i + 1, 'A' + now\_id);

if (next\_id == 0)

{

printf("\n");

}

SetEvent(hEvent[next\_id]); // 可以打印下一个了,下一个设置为激发态

ResetEvent(hEvent[now\_id]); // 设置为非激发态

if (i == 9)

{

CloseHandle(hEvent[now\_id]);

}

}

return 0;

}

int main()

{

// 第一个参数安全属性默认即可:NULL

// 第二个参数 TRUE表示程序员自己控制状态

// FALSE表示,WaitForSingleObject后自动改为非激发态

// 第三个参数为初始状态, TRUE表示激发态，FALSE表示非激发态

// 第四个参数为事件名称

hEvent[0] = CreateEvent(NULL, TRUE, TRUE, L"print\_a");

hEvent[1] = CreateEvent(NULL, TRUE, FALSE, L"print\_b");

hEvent[2] = CreateEvent(NULL, TRUE, FALSE, L"print\_c");

// 创建3线程

DWORD dThreadId1 = 0;

HANDLE hThread1 = CreateThread(NULL, NULL, ThreadProc, (LPVOID)0, NULL, &dThreadId1);

DWORD dThreadId2 = 0;

HANDLE hThread2 = CreateThread(NULL, NULL, ThreadProc, (LPVOID)1, NULL, &dThreadId2);

DWORD dThreadId3 = 0;

HANDLE hThread3 = CreateThread(NULL, NULL, ThreadProc, (LPVOID)2, NULL, &dThreadId2);

//它的意思是好戏开始

printf("Good Drama Start\n");

getchar();

return 0;

}

运行结果如下所示。

Good Drama Start

第1遍:打印A

第1遍:打印B

第1遍:打印C

第2遍:打印A

第2遍:打印B

第2遍:打印C

第3遍:打印A

第3遍:打印B

第3遍:打印C

第4遍:打印A

第4遍:打印B

第4遍:打印C

第5遍:打印A

第5遍:打印B

第5遍:打印C

第6遍:打印A

第6遍:打印B

第6遍:打印C

第7遍:打印A

第7遍:打印B

第7遍:打印C

第8遍:打印A

第8遍:打印B

第8遍:打印C

第9遍:打印A

第9遍:打印B

第9遍:打印C

第10遍:打印A

第10遍:打印B

第10遍:打印C

解析：代码中首先开启了3个线程，这三个线程的句柄是A、B和C。然后在通过循环输出10遍线程的ID。

## 19.6 习题

简答题

1.

答：静态库的代码是静态链接的，动态库的代码是动态链接的；静态库的代码被复制到可执行文件中，动态库的代码被加载到内存中；静态库在编译时就确定了代码，动态库在运行时才确定代码。另外，静态库的后缀名通常是“.a”或“.lib”，动态库的后缀名通常是“.so”或“.dll”。总的来说，静态库适用于程序比较小、部署简单的场景，动态库适用于程序比较大、需要共享库文件的场景。开发者需要根据具体的需求选择使用静态库还是动态库。

2．

答：

静态库的使用流程：

编写代码：开发者编写代码并将其编译成静态库文件。

链接静态库：将静态库与可执行文件进行链接，生成可执行文件。

运行程序：运行可执行文件，程序会使用静态库中的函数和数据。

动态库的使用流程：

编写代码：开发者编写代码并将其编译成动态库文件。

链接动态库：将动态库与可执行文件进行链接，生成可执行文件。

运行程序：运行可执行文件，程序会加载动态库，并调用其中的函数和数据。

查找动态库：程序在加载动态库时需要查找动态库的位置，可以使用环境变量、配置文件等方式进行设置。

更新动态库：如果动态库文件发生更新，可以直接替换原有的库文件，程序会自动加载新的库文件。

## 20.4 习题

程序设计题

1.

代码如下所示。

#include <iostream>

#include <vector>

using namespace std;

void reverse(int n, vector<int>& res)

{

if (n == 0)

{

return;

}

res.push\_back(n % 10);

reverse(n / 10, res);

}

int main()

{

int n = 12345;

vector<int> res;

reverse(n, res);

for (auto i : res)

{

cout << i << " ";

}

return 0;

}

运行结果如下所示

5 4 3 2 1

解析：代码中的reverse函数接受一个整数n和一个整型数组res作为参数，将n逆序后放入res数组中。如果n为0，则递归结束。否则，将n的个位数加入res数组中，然后递归处理n的十位数及以上的数字。在主函数中，我们将整数12345逆序放入res数组中，并输出该数组的元素。运行结果为：5 4 3 2 1。

2.

代码如下所示。

#include <iostream>

using namespace std;

bool isPalindrome(string s, int l, int r)

{

if (l >= r)

{

return true;

}

if (s[l] != s[r])

{

return false;

}

return isPalindrome(s, l + 1, r - 1);

}

int main() {

string s = "abcba";

if (isPalindrome(s, 0, s.length() - 1))

{

cout << s << " 是回文" << endl;

}

else

{

cout << s << " 不是回文." << endl;

}

return 0;

}

运行结果如下所示。

abcba 是回文

解析：代码中的isPalindrome函数接受一个字符串s和两个整数l和r作为参数，判断s从下标l到下标r的子串是否是回文串。如果s为空字符串或下标l大于等于下标r，则认为该子串是回文串。如果子串的第一个字符和最后一个字符不相等，则认为该子串不是回文串。否则，递归判断子串的第二个字符到倒数第二个字符是否是回文串。

3.

代码如下所示，

#include <iostream>

#include <vector>

using namespace std;

void primeFactors(int n, vector<int>& res) {

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

{

while (n % i == 0)

{

res.push\_back(i);

n /= i;

}

}

if (n > 1)

{

res.push\_back(n);

}

}

int main() {

int n = 435234;

vector<int> res;

primeFactors(n, res);

cout << n << " = ";

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

if (i != 0)

{

cout << " \* ";

}

cout << res[i];

}

return 0;

}

运行结果如下所示。

435234 = 2 \* 3 \* 17 \* 17 \* 251

解析：代码中的primeFactors函数接受一个整数n和一个整型数组res作为参数，将n分解成质因数并存储在res数组中。我们从2开始枚举可能的质因数i，如果i是n的因数，则将i加入res数组中，并将n除以i。然后，我们继续使用i作为因数，直到i大于n的平方根为止。如果此时n仍然大于1，则说明n本身是一个质数，将其加入res数组中即可。

## 21.10 习题

一、单项选择题

1.D

解析：顺序表中的元素在内存中是连续存储的，因此可以通过下标直接访问任意位置的元素，访问效率很高，时间复杂度为O(1)。

2.B

解析：冒泡排序算法时间复杂度不受数据初始状态影响，恒为*O*(*n*log*n*)。

3.D

解析：快速排序是一种高效的排序算法，其基本思想是通过一趟排序将待排记录分隔成独立的两部分，其中一部分记录的关键字均比另一部分记录的关键字小，然后再对这两部分记录分别进行快速排序，以达到整个序列有序的目的。所以，在一趟排序后不一定能将元素放置到指定位置。

4.C

解析：当待排序的序列中的元素已经排好序时，快速排序算法的最佳情况就是每次都能将待排序序列均匀地分成两份，此时递归树的深度为logn，每一层的时间复杂度为O(n)，因此总的时间复杂度为O(nlogn)。

5.D

解析：在有 n个结点且为 完全二叉树的排序中查找一键值，其平均比较次数量级个结点且为 完全二叉树的排序中查找一键值，其平均比较次数量级为*O*(log*n*)。

二、填空题

1.129

解析：n0=n2+1;n0=50;n2=49;n1=30;结点数129

2．0

解析：直接选择排序算法在最好情况下所做交换元素的次数为0。

3．归并排序

解析：占用辅助空间最多的是归并排序。

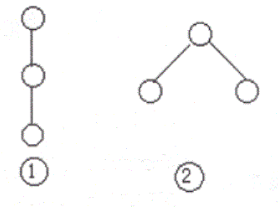
三、简答题

1.

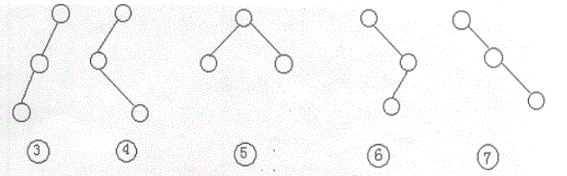
答：要达到这个目标，各种链表的要求不同如果是单链表，那么你必须知道一个在p前面的结点。如果不知道就就无法实现。如果是双链表和单循环链表则只要知道任意一个结点就可以了、时间复杂度全是O(n)。

2.

答：具有3个结点的树的形态为：①和②两种情况



具有3个结点的二叉树的形态为以下五种情况：



需要注意的是该面图②和图⑤是完全不同的结构，因为二叉树中的子树有左；右之分，而图②则无此要求。

## 22.5 习题

简答题

1．

答：

索引顺序的重要性：联合索引的查询效率与索引中列的顺序有关。在查询时，如果只涉及到联合索引中的第一个列，那么该联合索引的效率就会比较高；但如果查询中涉及到的列不是联合索引的第一个列，那么就可能会造成索引失效，需要遍历整个表进行查询，效率低下。因此，在创建联合索引时，需要考虑查询涉及到的列，并将其放在联合索引的前面。

索引列的选择：在创建联合索引时，需要选择合适的列作为索引列。通常情况下，选择那些经常用于查询、过滤和排序的列作为索引列。如果选择的列比较多，会增加索引的维护成本和存储成本，因此需要在效率和成本之间进行权衡。

索引的大小：联合索引的大小与索引列的数量和数据类型有关。如果索引列的数量比较多，数据类型比较长，那么索引的大小就会比较大，会占用较多的磁盘空间。因此，在创建联合索引时，需要考虑索引的大小，尽量选择那些占用空间较小的列作为索引列。

更新操作的影响：在进行更新操作时，如果更新的列恰好是联合索引的一部分，那么就会涉及到索引的更新和重构，会影响到系统的性能。因此，在创建联合索引时，需要考虑更新操作的影响，并根据实际情况进行权衡。

2．

答：

监控和分析MySQL的性能指标：了解MySQL的性能指标可以帮助我们及时发现和定位问题。可以使用MySQL自带的性能分析工具来监控和分析MySQL的性能指标。

优化查询语句：查询语句是MySQL性能问题的一个重要因素，因此需要对查询语句进行优化。

配置MySQL服务器参数：MySQL服务器参数对性能影响很大，因此需要根据实际情况对MySQL服务器参数进行调整。

优化表结构和数据类型：表结构和数据类型也会影响MySQL的性能。可以优化表结构，如采用垂直拆分、水平拆分、分区等技术，减小表的大小，提高查询效率。

使用缓存技术：使用缓存技术可以减轻MySQL的负担，提高系统的性能和可扩展性。

避免并发问题：可以通过使用事务、加锁、使用乐观锁等技术来避免并发问题，保证数据的一致性和可靠性。

定期维护MySQL：定期维护MySQL可以保证系统的稳定性和可靠性。

## 23.5 习题

简答题

1．

答：C/S架构（Client/Server Architecture），也叫客户端/服务器架构，是一种常见的分布式计算架构，它将系统划分为客户端和服务器两部分，客户端向服务器请求服务，服务器接受请求并提供服务。

C/S架构中，客户端和服务器之间通过网络进行通信，客户端发送请求并等待服务器的响应，服务器接收请求并进行处理，最终将结果返回给客户端。客户端通常是一个图形用户界面（GUI）程序，服务器则提供各种服务，如数据库服务、文件服务、应用程序服务等。

C/S架构的优势在于它可以将系统的各个部分分离开来，使客户端和服务器具有独立的开发和维护过程。客户端可以在不同的操作系统上运行，服务器也可以部署在不同的机器上，从而提高了系统的可扩展性和可靠性。

2．

答：互联网协议（Internet Protocol，简称IP）是互联网中最基本的协议之一，负责在网络中传输数据。IP协议定义了数据在网络中的传输方式和寻址方式，是互联网通信的基础。

互联网协议按照功能可以分为四层协议，分别是应用层、传输层、网络层和链路层。

应用层协议：应用层协议负责应用程序之间的通信。应用层协议包括HTTP、FTP、SMTP、POP3等，这些协议都是基于TCP或UDP协议实现的。应用层协议定义了数据格式和交互方式，例如HTTP协议定义了Web页面的请求和响应格式。

传输层协议：传输层协议负责端到端的数据传输。传输层协议包括TCP和UDP协议。TCP协议提供可靠的数据传输，可以保证数据的完整性和顺序性，但会造成较大的延迟。UDP协议则提供不可靠的数据传输，不保证数据的完整性和顺序性，但能够实现较低的延迟。

网络层协议：网络层协议负责数据在网络中的传输。网络层协议包括IP协议和ICMP协议。IP协议定义了数据在网络中的寻址方式和传输方式，是互联网通信的基础。ICMP协议则负责网络错误和控制消息的传输，例如PING命令使用的就是ICMP协议。

链路层协议：链路层协议负责数据在物理层的传输。链路层协议包括以太网协议、WiFi协议等。以太网协议是最常用的链路层协议，负责局域网内的数据传输，使用MAC地址进行寻址。

总的来说，互联网协议是互联网通信的基础，按照功能可以分为四层协议，包括应用层、传输层、网络层和链路层。不同层协议负责不同的功能，共同构成了互联网的通信体系。