四、程序填充

1. 斐波那契数列的第1和第2个数分别为0和1 ，从第三个数开始，每个数等于其前两个数之和。求斐波那契数列中的前20个数，要求每行输出5个数。

#include<iostream.h>

void main() {

int f,f1,f2,i;

cout<<” 斐波那契数列：\n”;

f1=0; f2=1;

cout<<setw(6)<<f1<<setw(6)<<f2;

for(i=3;i<=20;i++) {

f=\_\_\_\_\_\_(1)\_\_\_\_\_\_;

cout<<setw(6)<<f;

if(\_\_\_\_\_(2)\_\_\_\_\_\_) cout<<endl;

f1=f2;

f2=\_\_\_\_(3)\_\_\_\_\_\_\_;

}

cout<<endl;

}

(1) (2) (3)

2. 计算的值。

#include<iostream.h>

void main()

{

double x,p1=1,p2=1,s=0;

int i,j=1;

cout<<"输入x的值:";

cin>>x;

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

p1\*=\_\_\_(1)\_\_\_\_\_;

p2\*=\_\_\_\_(2)\_\_\_\_;

s+=j\*p1/p2; //j的值为(-1)i+1

j=\_\_\_\_(3)\_\_\_\_;

}

cout<<s<<endl;

}

(1) (2) (3)

3. 打印出2至99之间的所有素数(即不能被任何数整除的数)。

#include<iostream.h>

#include<math.h>

void main()

{

int i,n;

for(n=2; \_\_\_(1)\_\_\_; n++) {

int temp=int(sqrt(n)); //求出n的平方根并取整

for(i=2; \_\_\_(2)\_\_\_; i++)

if(n%i==0) \_\_\_(3)\_\_\_;

if(i>temp) cout<<n<<' ';

}

cout<<'\n';

}

(1) (2) (3)

4. 采用辗转相除法求出两个整数的最大公约数。

#include<iostream.h>

void main()

{

int a,b;

cout<<"请输入两个正整数:";

cin>>a>>b;

while(a<=0 || \_\_(1)\_\_\_) {cout<<"重新输入:"; cin>>a>>b;}

while(b) {

int r;

r=a%b;

\_\_\_(2)\_\_\_; \_\_\_(3)\_\_\_; //分别修改a和b的值

}

cout<<a<<endl; //输出最大公约数

}

(1) (2) (3)

5. 把从键盘上输入的一个大于等于3的整数分解为质因子的乘积。如输入24时得到的输出结果为“2 2 2 3”，输入50时得到的输出结果为“2 5 5”，输入37时得到的输出结果为“37”。

#include<iostream.h>

void main()

{

int x;

cout<<"请输入一个整数，若小于3则重输:";

do cin>>x; while(\_\_\_(1)\_\_\_);

int i=2;

do{

while(\_\_\_(2)\_\_\_) {

cout<<i<<' ';

x/=i;

}

\_\_\_(3)\_\_\_;

}while(i<x);

if(x!=1) cout<<x;

cout<<endl;

}

(1) (2) (3)

6. 下面函数是求两个整型参数a和b的最小公倍数。

int f2(int a, int b)

{

int i=2, p=1;

do {

while(a%i==0 && \_\_\_(1)\_\_\_) {

p\*=i; a/=i; b/=i;

}

\_\_\_(2)\_\_\_;

}while(a>=i && \_\_\_(3)\_\_\_);

return p\*a\*b;

}

(1) (2) (3)

7. 在输出屏幕上打印出一个由字符’\*’组成的等腰三角形，该三角形的高为5行，从上到下每行的字符数依次为1,3,5,7,9。

#include<iostream.h>

void main()

{

int i,j;

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

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

if(j<=5-i || \_\_\_(2)\_\_\_) cout<<’ ’;

else \_\_\_(3)\_\_\_;

cout<<endl;

}

}

(1) (2) (3)

8. 统计字符串中英文字母个数的程序。

#include <iostream.h>

int count (char str[]);

void main(){

char s1[80];

cout <<”Enter a line:”;

cin >>s1;

cout <<”count=”<<count(s1)<<endl;

}

int count(char str[]){

int num=0; //给统计变量赋初值

for(int i=0;str[i];i++)

if (str[i]>=’a’ && str[i]<=’z’ ||\_\_\_(1)\_\_\_ )

\_\_\_(2)\_\_\_;

\_\_\_(3)\_\_\_;

}

(1) (2) (3)

9. 主函数调用一个fun函数将字符串逆序。

#include<iostream.h>

#include<string.h>

\_\_\_(1)\_\_\_;

void main( ) {

char s[80];

cin>>s;

\_\_\_(2)\_\_\_;

cout<<”逆序后的字符串:”<<s<<endl ;

}

void fun(char ss[]) {

int n=strlen(ss);

for(int i=0; \_\_\_(3)\_\_\_\_; i++) {

char c=ss[i];

ss[i]=ss[n–1–i];

ss[n–1–i]=c;

}

}

(1) (2) (3)

10. 从一个字符串中删除所有同一个给定字符后得到一个新字符串并输出。

#include<iostream.h>

const int len=20;

void delstr(char a[],char b[],char c);

void main() {

char str1[len],str2[len];

char ch;

cout<<"输入一个字符串:";

cin>>str1;

cout<<"输入一个待删除的字符:";

cin>>ch;

delstr(str1,str2,ch);

cout<<str2<<endl;

}

void delstr(char a[],char b[],char c)

{

int j=0;

for(int i=0; \_\_\_(1)\_\_\_; i++)

if(\_\_\_(2)\_\_\_) b[j++]=a[i];

b[j]=\_\_\_(2)\_\_\_;

}

(1) (2) (3)

11. 采用指针访问方式从键盘给数组a[N]输入数据，然后对元素值重新按逆序存放并输出。

#include <iostream.h>

const int N=8;

void main()

{

int a[N],\*p,\*q;

for(p=a; p<a+N; p++) \_\_\_(1)\_\_\_;

p=a;q=a+N-1;

while(p<q) {

int r=\*p; \*p=\*q; \*q=r;

\_\_\_(2)\_\_\_; \_\_\_(3)\_\_\_;

}

for(p=a;p<a+N; p++)

cout<<\*p<<' ';

cout<<endl;

}

(1) (2) (3)

12. 从键盘上输入一个正整数，然后把它转换成的二进制数的每一位存放到一维数组中，最后输出该二进制数。注意二进制数的存放是按照从低位到高位的次序进行的。

#include <iostream.h>

void main()

{

int x;

cout<<"输入一个整数:";

cin>>x;

int a[20],k=0,r;

do {

r=x%2;

a[k++]=r;

x=\_\_\_(1)\_\_\_;

} while(\_\_\_(2)\_\_\_);

for(--k;k>=0;k--) \_\_\_(3)\_\_\_;

cout<<endl;

}

(1) (2) (3)

13. 对数组a[n]按升序进行的选择排序算法

void SelectSort(int a[], \_\_\_(1)\_\_\_)

{

int i,j,k;

for(i=1;i<n;i++) { //进行n-1次选择和交换

k=i-1;

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

if(a[j]<a[k]) \_\_\_(2)\_\_\_;

int x=a[i-1]; a[i-1]=a[k]; \_\_\_(3)\_\_\_;

}

}

(1) (2) (3)

14. 对数组a[n]按升序进行的插入排序算法

void InsertSort(\_\_\_(1)\_\_\_, int n)

{

int i,j,x;

for(i=1;i<n;i++) { //进行n-1次循环

x=a[i];

for(j=i-1;j>=0;j--) //为x顺序向前寻找合适的插入位置

if(x<a[j]) \_\_\_(2)\_\_\_;

else \_\_\_(3)\_\_\_;

a[j+1]=x;

}

}

(1) (2) (3)

15. 对按从小到大排列的有序数组a[n]进行二分查找x的算法，若查找成功返回该元素下标，否则返回-1。

int BinarySearch(int a[],int x)

{

int low=0, high=N-1; //定义并初始化区间下界和上界变量

int mid; //定义保存中点元素下标的变量

while(low<=high) {

mid=\_\_\_(1)\_\_\_;

if(x==a[mid]) \_\_\_(2)\_\_\_;

else if(x<a[mid]) high=mid-1;

else \_\_\_(3)\_\_\_;

}

return -1;

}

(1) (2) (3)

16. 用插入排序方法对table指针数组中size个指针所指向的字符串进行按升序排序的算法。

void sort(char \*table[], int size){

for(int i=1,\_\_\_(1)\_\_\_; i++){

char \*p=table[i];

for(int j=i-1; j>=0 ; j--)

if(strcmp(p,table[j])<0) \_\_\_(2)\_\_\_;

else break;

table[j+1]=\_\_\_(3)\_\_\_;

}

}

(1) (2) (3)

17. 假定有定义为“struct NODE{int data; NODE\* next;};”，下面算法根据table数组中的n个元素建立一个表头指针为L的链表，链表中结点值的顺序与数组元素值的顺序相同。

void f5(NODE\*& L, int table[], int n)

{

if(n<=0) {L=NULL; return;}

L=new NODE; //生成附加的头结点

int i=0;

NODE\* p=L;

while(\_\_\_(1)\_\_\_) {

p=p->next=\_\_\_(2)\_\_\_;

p->data=\_\_\_(3)\_\_\_;

i++;

}

p->next=NULL; //把最后一个结点的指针域置空

p=L;

L=L->next; //使L指向链表的第一个带值的结点

delete p;

}

(1) (2) (3)

18. 假定有定义为“struct NODE{int data; NODE\* next;};”，下面算法根据table数组中的n个元素建立一个表头指针为L的链表，链表中结点值的顺序与数组元素值的顺序正好相反。

void f6(NODE\*& L, int table[], int n)

{

L=NULL;

if(n<=0) return;

int i=0;

NODE\* p;

while(\_\_\_(1)\_\_\_) {

p=new NODE;

p->data=\_\_\_(2)\_\_\_;

p->next=L;

\_\_\_(3)\_\_\_;

i++;

}

}

(1) (2) (3)

19. 假定有定义为“struct NODE{int data; NODE\* next;};”，下面算法是依次显示输出以L为表头指针的链表中各结点的值。

void f7(NODE\* L)

{

for(\_\_\_(1)\_\_\_; p!=NULL; \_\_\_(2)\_\_\_)

cout<<\_\_\_(3)\_\_\_ <<' ';

cout<<endl;

}

(1) (2) (3)

20. 假定有定义为“struct NODE{int data; NODE\* next;};”，下面算法是把以L为表头指针的链表中各结点依次按相反次序链接并返回新链表的表头指针。

NODE\* f8(NODE\* L)

{

if(L==NULL) return NULL;

NODE \*p=NULL, \*q=L, \*t;

while(q!=NULL) {

t=q;

q=\_\_\_(1)\_\_\_;

t->next=\_\_\_(2)\_\_\_;

p=t;

}

\_\_\_(3)\_\_\_;

}

(1) (2) (3)

21. 已知一维数组类ARRAY的定义如下，ARRAY与普通一维数组区别是：其重载的运算符[ ]要对下标是否越界进行检查。

class ARRAY{

int \*v; //指向存放数组数据的空间

int s; //数组大小

public:

ARRAY(int a[], int n);

~ ARRAY(){delete []v;}

int size(){ return s;}

int& operator[](int n);

};

\_\_\_(1)\_\_\_ operator[](int n) //[ ]的运算符成员函数定义

{

if(n<0 || \_\_\_(2)\_\_\_) {cerr<<"下标越界！"; exit(1);}

return \_\_\_(3)\_\_\_;

}

(1) (2) (3)

22. 已知一维数组类ARRAY的定义如下，构造函数的作用是把参数n的值赋给s，给v动态分配长度为n的数组空间，接着利用数组参数a初始化v所指向的数组。

class ARRAY{

int \*v; //指向存放数组数据的空间

int s; //数组大小

public:

ARRAY(int a[], int n);

~ ARRAY(){delete []v;}

int size(){ return s;}

int& operator[](int n);

};

\_\_\_(1)\_\_\_ ARRAY(int a[], int n)

{

if(n<=0) {v=NULL;s=0;return;}

s=n;

v=\_\_\_(2)\_\_\_;

for(int i=0; i<n; i++) \_\_\_(3)\_\_\_;

}

(1) (2) (3)

23. 下面是一维数组类ARRAY的定义，ARRAY与普通一维数组区别是：(a)用()而不是[]进行下标访问，(2)下标从1而不是从0开始，(c)要对下标是否越界进行检查。

class ARRAY{

int \*v; //指向存放数组数据的空间

int s; //数组大小

public:

ARRAY(int a[], int n);

~ ARRAY(){delete []v;}

int size(){ return s;}

int& operator()(int n);

}; \_\_\_(1)\_\_\_ operator()(int n)

{ // ()的运算符函数定义

if(\_\_\_(2)\_\_\_) {cerr<<"下标越界！"; exit(1);}

return \_\_\_(3)\_\_\_;

}

(1) (2) (3)

24. 已知一个类的定义如下：

#include<iostream.h>

class AA {

int a[10];

int n;

public:

void SetA(int aa[], int nn); //用数组aa初始化数据成员a，

//用nn初始化数据成员n

int MaxA(); //从数组a中前n个元素中查找最大值

void SortA(); //采用选择排序的方法对数组a中前n个元素

//进行从小到大排序

void InsertA();//采用插入排序的方法对数组a中前n个元素进行从小到大排序

void PrintA(); //依次输出数组a中的前n个元素

};

该类中MaxA()函数的实现如下，请在标号位置补充适当的内容。

int \_\_\_\_(1)\_\_\_\_\_

{

int x=a[0];

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

if(a[i]>x) \_\_\_(2)\_\_\_;

\_\_\_(3)\_\_\_;

}

(1) (2) (3)

25. 已知一个类的定义如下：

#include<iostream.h>

class AA {

int a[10];

int n;

public:

void SetA(int aa[], int nn); //用数组aa初始化数据成员a，

//用nn初始化数据成员n

int MaxA(); //从数组a中前n个元素中查找最大值

void SortA(); //采用选择排序的方法对数组a中前n个元素

//进行从小到大排序

void InsertA();//采用插入排序的方法对数组a中前n个元素进行从小到大排序

void PrintA(); //依次输出数组a中的前n个元素

};

void AA::SortA()

{

int i,j;

for(i=0; \_\_\_(1)\_\_\_; i++) {

int x=a[i], k=i;

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

if(a[j]<x) \_\_\_(2)\_\_\_

a[k]=a[i];

\_\_\_(3)\_\_\_;

}

}

(1) (2) (3)

26. 已知一个类的定义如下：

#include<iostream.h>

class AA {

int a[10];

int n;

public:

void SetA(int aa[], int nn); //用数组aa初始化数据成员a，

//用nn初始化数据成员n

int MaxA(); //从数组a中前n个元素中查找最大值

void SortA(); //采用选择排序的方法对数组a中前n个元素

//进行从小到大排序

void InsertA();//采用插入排序的方法对数组a中前n个元素进行从小到大排序

void PrintA(); //依次输出数组a中的前n个元素

};

void \_\_\_(1)\_\_\_

{

int i,j;

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

int x=a[i];

for(j=i-1; j>=0; j--)

if(x<a[j]) \_\_\_(2)\_\_\_;

else \_\_\_(3)\_\_\_;

a[j+1]=x;

}

}

(1) (2) (3)

27. 已知一个类的定义如下：

#include<iostream.h>

class AA {

int a[10];

int n;

public:

void SetA(int aa[], int nn); //用数组aa初始化数据成员a，

//用nn初始化数据成员n

int MaxA(); //从数组a中前n个元素中查找最大值

void SortA(); //采用选择排序的方法对数组a中前n个元素

//进行从小到大排序

void InsertA();//采用插入排序的方法对数组a中前n个元素进行从小到大排序

void PrintA(); //依次输出数组a中的前n个元素

//最后输出一个换行

};

使用该类的主函数如下：

void main()

{

int a[10]={23,78,46,55,62,76,90,25,38,42};

AA x;

\_\_\_(1)\_\_\_;

int m=\_\_\_(2)\_\_\_;

\_\_\_(3)\_\_\_;

cout<<m<<endl;

}

该程序运行结果为:

23 78 46 55 62 76

78

(1) (2) (3)

28. 已知一个类的定义如下：

#include<iostream.h>

class AA {

int a[10];

int n;

public:

void SetA(int aa[], int nn); //用数组aa初始化数据成员a，

//用nn初始化数据成员n

int MaxA(); //从数组a中前n个元素中查找最大值

void SortA(); //采用选择排序的方法对数组a中前n个元素

//进行从小到大排序

void PrintA(); //依次输出数组a中的前n个元素，

//最后输出一个换行

};

使用该类的主函数如下：

void main()

{

int a[10]={23,78,46,55,62,76,90,25,38,42};

\_\_\_(1)\_\_\_;

x.SetA(a,8);

int \_\_\_(2)\_\_\_;

\_\_\_(3)\_\_\_;

x.PrintA();

cout<<m<<endl;

}

该程序运行结果为:

23 25 46 55 62 76 78 90

90

(1) (2) (3)

29. 已知一个利用数组实现栈的类定义如下：

const int ARRAY\_SIZE=10;

class Stack {

public:

void Init() {top=-1;} //初始化栈为空

void Push(int newElem); //向栈中压入一个元素

int Pop(); //从栈顶弹出一个元素

bool Empty() { //判栈空

if(top==-1) return true;else return false;}

int Depth() {return top+1;} //返回栈的深度

void Print();

//按照后进先出原则依次输出栈中每个元素，直到栈空为止

private:

int elem[ARRAY\_SIZE]; //用于保存栈元素的数组

int top; //指明栈顶元素位置的指针

};

void Stack::Push(int newElem) {

if(\_\_\_(1)\_\_\_) {

cout<<"栈满!"<<endl;

exit(1); //中止运行

}

\_\_\_(2)\_\_\_;

elem[top]=\_\_\_(3)\_\_\_;

}

(1) (2) (3)

30. 已知一个利用数组实现栈的类定义如下：

const int ARRAY\_SIZE=10;

class Stack {

public:

void Init() {top=-1;} //初始化栈为空

void Push(int newElem); //向栈中压入一个元素

int Pop(); //从栈顶弹出一个元素

bool Empty() { //判栈空

if(top==-1) return true;else return false;}

int Depth() {return top+1;} //返回栈的深度

void Print();

//按照后进先出原则依次输出栈中每个元素，直到栈空为止

private:

int elem[ARRAY\_SIZE]; //用于保存堆栈元素的数组

int top; //指明栈顶元素位置的指针

};

该类的Pop和Print函数的实现分别如下：

\_\_\_(1)\_\_\_ {

if(top==-1) {

cout<<"栈空!"<<endl;

exit(1); //中止运行

}

return \_\_\_(2)\_\_\_;

}

void Stack::Print() {

while(!Empty())

cout<<\_\_\_(3)\_\_\_ <<' ';

}

(1) (2) (3)

31．class A {

int a;

public:

A() {a=0;}

\_\_\_(1)\_\_\_{} //定义构造函数，用参数aa初始化数据成员a

};

main() {

\_\_\_(2)\_\_\_; //定义类A的指针对象p

\_\_\_(3)\_\_; //用p指向动态对象并初始化为整数5

}

(1) (2) (3)

32．class A {

char \*a;

public:

\_\_\_(1)\_\_\_ //定义无参构造函数，使a的值为空

A(char \*aa) {

a=\_\_\_(2)\_\_\_;

strcpy(a,aa); //用aa所指字符串初始化a所指向的动态存储空间

}

\_\_\_(3)\_\_\_ //定义析构函数，删除a所指向的动态存储空间

};

(1) (2) (3)

33．class A {

int a,b;

public:

A(int aa=0, int bb=0) \_\_\_(1)\_\_\_ {} //分别用aa和bb对应初始化a和b

};

main() {

\_\_\_(2)\_\_\_ ; //定义类A的对象x并用5初始化，同时定义y并用x初始化

\_\_\_(3)\_\_\_ ; //定义p指针，使之指向对象x

}

(1) (2) (3)

34．class A {

int a,b;

public:

\_\_\_(1)\_\_\_ //定义构造函数，使参数aa和bb的默认值为0，

//在函数体中用aa初始化a，用bb初始化b

};

main() {

A \*p1, \*p2;

\_\_\_(2)\_\_\_ ; //调用无参构造函数生成由p1指向的动态对象

\_\_\_(3)\_\_\_ ; //调用带参构造函数生成由p2指向的动态对象，

//使a和b成员分别被初始化为4和5

}

(1) (2) (3)

35. #include<iostream.h>

#include<stdlib.h>

class A {

int \*a; int n; int MaxLen;

public:

A(): a(0), n(0), MaxLen(0) {}

A(int \*aa, int nn, int MM) {

n=nn;

MaxLen=MM;

if(n>MaxLen) exit(1);

\_\_\_(1)\_\_\_; //由a指向长度为MaxLen的动态数组

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

}

~A() {delete []a;}

int GetValue(int i) \_\_\_(2)\_\_\_ //函数体返回a[i]的值

};

void main()

{

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

A r(b,10,10);

int i,s=0;

for(i=0; i<10; i++) \_\_\_(3)\_\_\_ ; //把r对象的a数据成员中的每个

//元素值依次累加到s中

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

}

(1) (2) (3)

36. #include<iostream.h>

#include<stdlib.h>

class A {

int \*a; int n; int MaxLen;

public:

A(): a(0), n(0), MaxLen(0) {}

A(int \*aa, int nn, int MM) {

n=nn;

MaxLen=MM;

if(n>MaxLen) exit(1);

a=new int[MaxLen];

\_\_\_(1)\_\_\_; //以i为循环变量把aa数组中每个元素值

//传送给a数组的对应元素中

}

~A();

int GetValue(int i) {return a[i];} //函数体返回a[i]的值

};

\_\_\_(2)\_\_\_ //析构函数的类外定义

void main()

{

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

A r(b,10,10);

int i,s=0;

\_\_\_(3)\_\_\_ ; //以i为循环变量，把r对象的a数据成员中的

//每个元素值依次累加到s中

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

}

(1) (2) (3)

37. 一种类定义如下:

class Goods

{

private:

char gd\_name[20]; //商品名称

int weight; //商品重量

static int totalweight; //同类商品总重量

public:

Goods (char\*str,int w){ //构造函数

strcpy(gd\_name,str);

weight=w;

totalweight+=weight;

}

~ Goods (){totalweight -= weight;}

char\* GetN(){\_\_\_(1)\_\_\_;} //返回商品名称

int GetW(){return weight;}

\_\_\_(2)\_\_\_ GetTotal\_Weight() { //定义静态成员函数返回总重量

\_\_\_(3)\_\_\_;

}

}

(1) (2) (3)

38.

class Point

{

private:

int x, y;

public:

Point(){x=y=0;}

Point(int x0,int y0) {x=x0;y=y0;}

int GetX() { return x; }

int GetY() { return y; }

void Print(){cout<<"Point("<<x<<","<<y<<")"<<endl;}

\_\_\_(1)\_\_\_; //友元函数声明

\_\_\_(2)\_\_\_; //友元函数声明

};

Point operator+(Point& pt,int dd)

//加号操作符重载函数，实现Point类对象与整数加法

{

Point temp=pt;

temp.x+=dd;

temp.y+=dd;

return temp;

}

Point operator+(Point& pt1,Point& pt2)

//加号操作符重载函数，实现两个Point类对象的加法

{

Point temp=pt1;

temp.x+=pt2.x;

\_\_\_(3)\_\_\_;

return temp;

}

(1) (2) (3)

39. 在下面一段类定义中， Derived类是由直接基类Base 1和Base 2所派生的，Derived类包含有两个间接基类BaseBase，在初始化函数Init中，需要把x1和x2的值分别赋给属于基类Base1的x成员和属于基类Base2的x成员。

class BaseBase {

protected:

int x;

public:

BaseBase(){ x = 1;}

};

class Base1: public BaseBase {

public:

Base1(){}

};

class Base2: public BaseBase {

public:

Base2(){}

};

class Derived: \_\_\_(1)\_\_\_

{

public:

Derived() {}

void Init(int x1, int x2) {

\_\_\_(2)\_\_\_; \_\_\_(3)\_\_\_;

}

void output() {cout<<Base1::x<<' '<<Base2::x<<endl;}

};

(1) (2) (3)

40. 在下面一段类定义中， Derived类公有继承了基类Base。需要填充的函数由注释内容给出了功能。

class Base

{

private:

int mem1,mem2; //基类的数据成员

public:

Base(int m1,int m2) {

mem1=m1; mem2=m2;

}

void output(){cout<<mem1<<' '<<mem2<<' ';}

//...

};

class Derived: public Base

{

private:

int mem3; //派生类本身的数据成员

public:

//构造函数,由m1和m2分别初始化mem1和mem2，由m3初始化mem3

Derived(int m1,int m2, int m3);

//输出mem1,mem2和mem3数据成员的值

void output(){

\_\_\_(1)\_\_\_; cout<<mem3<<endl;

}

//...

};

Derived::Derived(int m1,int m2, int m3): \_\_\_(2)\_\_\_ {\_\_\_(3)\_\_\_;}

(1) (2) (3)

41. 在下面一段类的定义中，需要填充的函数由注释内容给出了功能。

class Point //定义坐标点类

{

public:

int x,y; //点的横坐标和纵坐标

Point(){x=0;y=0;}

Point(int x0,int y0) {x=x0; y=y0;}

int X(){return x;}

int Y(){return y;}

void PrintP(){cout<<"Point:("<<x<<","<<y<<")"<<endl;}

};

class Line: public Point //利用坐标点类定义直线类

{

private:

class Point pt1,pt2; //直线的两个端点

public:

Line(Point pts, Point pte); //构造函数，分别用参数初始化对应的端点

double Dx(){return pt2.x-pt1.x;}

double Dy(){return pt2.y-pt1.y;}

double Length(){ //计算直线的长度

return sqrt(\_\_\_(1)\_\_\_);

};

void PrintL(); //输出直线的两个端点和直线长度

};

Line::Line(Point pts, Point pte) \_\_\_(2)\_\_\_

void Line::PrintL()

{

cout<<"1st ";

pt1.PrintP();

cout<<"2nd ";

pt2.PrintP();

cout<<"The Length of Line: "<<\_\_\_(3)\_\_\_ <<endl;

}

(1) (2) (3)

42. 在下面一段类的定义中，自行车类的虚基类为车辆类，机动车类的虚基类也为车辆类，摩托车类的基类为自行车类和机动车类，类之间均为公有继承。

class vehicle //车辆类

{

private:

int MaxSpeed; //最大车速

int Weight; //车重

public:

vehicle(){MaxSpeed=0; Weight=0;};

virtual void Run() {cout<<"A vehicle is running!"<<endl;}

};

class bicycle : \_\_\_(1)\_\_\_ //自行车类

{

private:

int Height; //车高

public:

bicycle(){};

void Run() {cout<<"A bicycle is running!"<<endl;}

};

class motorcar : \_\_\_(2)\_\_\_ //机动车类

{

private:

int SeatNum; //乘人数

public:

motorcar(){};

void Run() {cout << "A motorcar is running!" << endl; }

};

class motorcycle: \_\_\_(3)\_\_\_ //摩托车类

{

public:

motorcycle (){};

void Run() {cout<<"A motorcycle is running!"<<endl;}

};

(1) (2) (3)