**选择**

1.A。执行"myClass a(2)"时，会调用一次myClass的构造函数，传入参数2。执行"myClass b[2]"时，会调用两次myClass的构造函数，分别对b[0]和b[1]进行初始化。执行"\*p[2]"时，会创建一个指向myClass对象的指针数组p，但是不会调用构造函数。因此，总共调用myClass的构造函数3次。

2.A。A. 静态成员函数不但可以访问类的所有 public 数据成员，也可以访问类的所有 private 数据成员 是不正确的描述。静态成员函数只能访问类的静态数据成员和静态成员函数，不能访问非静态数据成员和非静态成员函数。

3.B。++和都是单目运算符，优先级相同，从左到右结合，所以是operator(operator++(i),k)。

4.C。用尾指针表示的单循环链表可以在O(1)时间内进行插入操作。

5.B。正确答案是B。

A选项错误，公有继承中，派生类只能访问基类中的公有成员和受保护成员，不能访问私有成员。

B选项正确，派生类对象的地址可以赋值给指向基类的指针，这是因为派生类包含了基类的所有成员，所以可以通过基类指针访问派生类对象的成员。

C选项错误，基类的友元不是派生类的友元，派生类需要自己定义自己的友元。

D选项错误，派生类不会隐式继承基类的构造函数和析构函数，需要自己定义。

6.C。根据p所指的对象类型而确定调用类A或类B中的函数func()。

7.A。T为int，后面不符合了

8.D。类模板的模板参数可以用作数据成员的类型、成员函数的类型和成员函数的参数类型。

9.D。派生类的虚函数与基类中对应的虚函数具有相同的函数名、参数个数和类型。D. 派生类的虚函数与基类中对应的虚函数具有相同的参数个数和类型是正确的描述。虚函数是一个动态绑定的成员函数，可以在基类中声明为虚函数，在派生类中进行重写。在运行时，会根据对象的实际类型来调用对应的虚函数，实现多态性。虚函数的调用是动态联编的。

10.A。ifstream默认打开方式为ios::in，即输入方式。A. ios::in 是 ifstream 流类的默认打开方式，用于读取文件。

**改错题1**

源代码：

class Node{

public:

int data;

Node \*next;

Node(int i) {

data = i;

next = nullptr;

}

}

class List {

private:

Node \*head;

static int nodeNum;

public:

List()

{

head = new Node();

}

void Insert(int k, int data);

};

List::nodeNum = 0;

void Insert(int k, int data) {

Node \*newNode = new Node(data);

Node \*p = head;

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

p = p->next;

newNode->next = p;

p->next = newNode;

}

这段代码存在一些问题，我来帮你指出并解决：

List::nodeNum = 0; 应该放在类定义外部，作为类静态成员变量的初始化。

void Insert(int k, int data) 函数定义应该加上 List::，即 void List::Insert(int k, int data)。

在 List() 构造函数中，应该将 head 指针初始化为 nullptr 而不是新建一个空节点。

在 Insert() 函数中，当 k 为 0 时应该特殊处理，即将 newNode 插入到 head 节点之后，然后将 head 指针指向 newNode。

在 Insert() 函数中，当 k 大于链表长度时应该特殊处理，即将 newNode 插入到链表末尾。

在 Insert() 函数中，应该在插入节点之前判断 p 是否为 nullptr，如果是则说明链表为空，直接将 newNode 插入到 head 节点之后，然后将 head 指针指向 newNode。

**修正后的代码：**

class Node {

public:

int data;

Node \*next;

Node(int i) {

data = i;

next = nullptr;

}

};

class List {

private:

Node \*head;

static int nodeNum;

public:

List() {

head = nullptr;

}

void Insert(int k, int data);

};

int List::nodeNum = 0;

void List::Insert(int k, int data) {

Node \*newNode = new Node(data);

Node \*p = head;

if (k == 0) {

newNode->next = head;

head = newNode;

return;

}

for (int i = 1; i < k && p != nullptr; i++) {

p = p->next;

}

if (p == nullptr) {

if (head == nullptr) {

head = newNode;

} else {

p = head;

while (p->next != nullptr) {

p = p->next;

}

p->next = newNode;

}

} else {

newNode->next = p->next;

p->next = newNode;

}

}

int main() {

List list;

list.Insert(0, 1);

list.Insert(1, 2);

list.Insert(2, 3);

list.Insert(1, 4);

return 0;

}

改错题2

错误1：第14行，构造函数调用了 A()，但是 A 类的构造函数需要一个 int 类型的参数。 错误2：第11行，using A::A(int a) 的语法不正确，应该使用冒号来继承父类的构造函数。

错误3：第28行，A 类中没有 f(int a) 函数，需要修改为 b1.f(10)。

错误4：第30行，a2 是 A 类型的指针，不能调用 B 类的成员函数，需要将 a2 的类型改为 B\*。

#include <iostream>

using namespace std;

class A {

protected:

int x;

public:

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

void f() { cout << "A" << endl; }

};

class B :public A {

public:

int y;

using A::A;

B(int a, int b) :A(a) { y = b; }

void f() { cout << "B" << endl; }

void showall() {

cout << "x: " << x << endl;

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

}

int getx() {

return x;

}

};

int main() {

B b1(10, 20);

cout << "x: " << b1.x << endl;

b1.f();

A\* a2 = new B(30, 40);

a2->f();

return 0;

}

**13.**

小明的年龄是15，成绩是90 (当前共1名学生)

李磊的年龄是16，成绩是80 (当前共2名学生)

张华的年龄是16，成绩是90 (当前共3名学生)

王康的年龄是14，成绩是60 (当前共4名学生)

平均成绩为80

14.

1010

这段程序的功能是将十进制数10转换为二进制数并输出。

程序首先定义了一个栈s，然后将十进制数10转换为二进制数并依次将每一位压入栈s中。具体地，程序通过循环将10不断除以2，每次将余数压入栈s中，直到商为0为止。这样，栈s中存储的就是10的二进制表示的每一位。

接着，程序通过循环将栈s中的元素依次弹出并输出，即可得到10的二进制表示。输出结果为：1010。

15.

Construction of Base Class

Construction of Derived Class

Fun1() in BaseClass is called!

Fun2() in DerivedClass is called!

解析：

首先创建一个DerivedClass对象d，因为DerivedClass继承自BaseClass，所以在创建DerivedClass对象时，会先调用BaseClass的构造函数，输出 "Construction of Base Class"，然后再调用DerivedClass的构造函数，输出 "Construction of Derived Class"。

接着将DerivedClass对象的地址赋给一个BaseClass指针pb，这是一个向上转型。

调用pb->Fun1()，由于Fun1()在BaseClass中不是虚函数，所以会调用BaseClass中的Fun1()，输出 "Fun1() in BaseClass is called!"。

调用pb->Fun2()，由于Fun2()在BaseClass中是虚函数，DerivedClass中也重写了Fun2()，所以会根据对象的实际类型调用DerivedClass中的Fun2()，输出 "Fun2() in DerivedClass is called!"。

16.

Animal(string n) called Cat(string name,string \_hair) called hair: short Animal(string n) called Cat(string name) called name: cat age: 10

程序首先创建了一个名为“cat”的猫对象p1，构造函数Cat(string name,string \_hair)被调用，输出“Cat(string name,string \_hair) called”，同时调用了父类Animal的构造函数Animal(string n)，输出“Animal(string n) called”。接着调用了print()函数，输出“hair: short”。

然后创建了一个名为“dog”的动物对象p2，构造函数Animal(string n,int a)被调用，输出“Animal(string n,int a) called”。接着将p1的地址赋给p2，即p2指向了p1，因为Cat类是Animal类的子类，所以可以这样做。最后调用了p2的print()函数，输出“name: cat”和“age: 10”。注意，这里调用的是Animal类的print()函数，而不是Cat类的print()函数，因为p2的类型是Animal指针。

17.

void merge(node \*h) {

node \*p1, \*p2;

if (h == NULL) return;

p1 = h;

p2 = h->next;

while (p2) {

p1->data += p2->data;

p1->next = p2->next;

delete p2;

p1 = p1->next;

if (p1 && p1->next)

p2 = p1->next;

else

p2 = NULL;

}

}

题目要求将链表上相邻的两个结点合并成一个结点，即将第 1 个结点与第 2 个结点合并，将第 3 个结点与第 4 个结点合并，……，如果链表上结点个数为奇数，则最后一个结点不合并，直接作为合并后链表上的最后一个结点。合并 2 个结点的含义是：将两个结点的数据成员 data 值相加。

根据题目要求，我们可以使用两个指针 p1 和 p2 分别指向相邻的两个结点，将它们的 data 值相加，然后将 p2 结点删除，将 p1 的 next 指针指向 p2 的 next 指针，最后将 p1 指向下一个结点，p2 指向下下个结点，继续进行合并操作，直到 p2 指向 NULL。

18.

class Rational {

int nume, deno;

public:

Rational(int x = 0, int y = 1) { nume = x; deno = y; }

void print();

Rational operator+(Rational a);

Rational operator-(Rational a);

};

void Rational::print() {

cout << nume << "/" << deno;

}

Rational Rational::operator+(Rational a) {

Rational r;

r.deno = a.deno \* deno;

r.nume = a.nume \* deno + a.deno \* nume;

return r;

}

Rational Rational::operator-(Rational a) {

Rational r;

r.deno = a.deno \* deno;

r.nume = nume \* a.deno - deno \* a.nume;

return r;

}

19.

#include<iostream>

using namespace std;

const double Pi = 3.14;

class circle {

public:

circle(double R);

double area() {

return Pi\*r\*r;

}

protected:

double r;

};

circle::circle(double R) {

r = R;

}

class cylinder :public circle {

protected:

double h;

public:

cylinder(double R, double H) : circle(R), h(H) {};

double area() {

return 2 \* Pi\*r\*h + 2 \* circle::area();

}

};

int main() {

cylinder c(1, 2);

cout << "The area of the cylinder c is " << c.area() << endl;

return 0;

}

20.

#include <fstream>

using namespace std;

struct date {

int year, month, day;

};

int main() {

date t = {2002, 2, 12};

ofstream outdate("date.txt");

if (!outdate) {

cerr << "\n 文件不能打开" << endl;

return 1;

}

outdate << t.year << " " << t.month << " " << t.day << endl;

outdate.close();

return 0;

}

(1) 使用 ofstream 类型的对象 outdate 打开文件 "date.txt"，并以写入模式打开。

(2) 将结构体变量 t 中的内容写入文件 outdate 中。需要注意的是，结构体变量中的每个成员变量之间需要用空格隔开，最后需要换行。写入完成后，需要关闭文件。

21.去除一个已有的 C++源程序文件中的所有的 for 语句。要求：

（1）在程序中由用户输入所要处理的 cpp 文件的文件名。

（2）如果该文件不存在返回一个提示信息

（3）去除程序中所有 for 语句后，得到的新程序存储到一个新文件中，新文件的文

件名是 new+源文件名,例如原文件名是 CPPfile.cpp,新文件是 newCPPfile,cpp。

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

int main() {

string filename;

cout << "Enter the filename: ";

cin >> filename;

ifstream fin(filename);

if (!fin) {

cout << "File not found." << endl;

return 0;

}

string newfilename = "new" + filename;

ofstream fout(newfilename);

string line;

while (getline(fin, line)) {

size\_t pos = line.find("for");

while (pos != string::npos) {

line.erase(pos, 3);

pos = line.find("for");

}

fout << line << endl;

}

fin.close();

fout.close();

cout << "New file created: " << newfilename << endl;

return 0;

**}**

**22.**

(1)void Exploration(long order[100][100], User\_Video record[10000]) {

for (int i = 0; i < 100; i++) {

int cnt = 0;

int j = 0;

while (cnt < 3 && j < 100) {

if (!record[i \* 100 + j].isClicked()) {

order[i][10 + cnt] = record[i \* 100 + j].getVideoID();

cnt++;

}

j++;

}

int randomIndex = rand() % 100;

order[i][13] = record[i \* 100 + randomIndex].getVideoID();

sort(order[i], order[i] + 14, [&](long a, long b) {

return record[i \* 100 + a].getScore() > record[i \* 100 + b].getScore();

});

}

}

这个函数的实现思路是，对于每个用户，找到他过去不常观看的三个视频，以及随机选择一个视频，插入到原有的六个视频中，重新生成 Top 10 推荐视频。

这个方法的合理性在于，通过插入用户过去不常观看的视频，可以让用户发现更多新的内容，从而增加用户的兴趣广度。同时，通过随机插入一个视频，可以保证用户看到的内容更加多样化，从而避免了“信息茧房”的问题。

(2) 开放性问题：

第一，作为一个用户，可以通过以下方式发现抖音推荐算法出现了不当行为：

* 观察推荐内容的多样性，如果推荐内容过于单一，可能是算法在进行大数据杀熟。
* 观察推荐内容的时长，如果推荐的内容时长过短，可能是算法在进行诱导沉迷。
* 观察推荐内容的质量，如果推荐的内容质量较低，可能是算法在进行卖流量。

第二，如果我是抖音运营商，可以在遵循法律和道德的前提下，实现自身的经济利益的方式包括：

* 通过提高广告收益，增加平台的收入。
* 通过推出新的商业模式，如直播带货、达人探店等，增加平台的用户粘性和流量。
* 通过提高用户体验，增加用户的留存率和活跃度，从而增加平台的价值。