**项目实验**

**一、程序设计题（共10小题，100分）**

|  |  |
| --- | --- |
| 题型得分 | 0 |

1. 【描述】  
   有两个字符串str和substr，str的字符个数不超过10，substr的字符个数为3。将substr插入到str中ASCII码最大的那个字符后面，若有多个最大则只考虑第一个。  
   【输入】  
   输入包括若干行，每一行为一组测试数据，格式为  
   str substr  
   【输出】  
   对于每一组测试数据，输出插入之后的字符串。  
   【输入示例】  
   abcab eee  
   12343 555  
   【输出示例】  
   abceeeab  
   12345553（10分）

**我的答案：**

#include "iostream"

#include "string"

#include "algorithm"

using namespace std;

int main(){

while (1){

string str="",substr="";

cin>>str;

cin>>substr;

int index=0;

char maxChar=NULL;

for(int i=0;i<str.length();i++){

if(str[i]>maxChar) maxChar=str[i];

}

for(int i=0;i<str.length();i++){

if(str[i]==maxChar){

index=i;

break;

}

}

str.insert(index+1,substr);

cout<<str<<endl;

}

return 0;

}

|  |  |
| --- | --- |
| 题目得分 |  |

**参考答案：**

#include <iostream>

#include <string>

using namespace std;

int main(){

string str, substr;

// 处理多组数据

while(cin >> str) {

cin >> substr;

// 用len遍历str，获得长度、最大ASCII字符及其下标

int len = 0;

char maxChar = 0;

int maxIdx = 0;

// 遍历数组，并获得相关信息

len = str.length();

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

if(str[i] > maxChar) {

maxChar = str[i];

maxIdx = i;

}

}

str.insert(maxIdx + 1, substr);

cout << str << endl;

}

return 0;

}

1. 【描述】  
   某个科室的病房分为重症和普通，只有当病人的疾病严重程度超过了入住重症病房的最低严重值，才可以安排入住重症病房。   
   现在要求设计一个程序，给病人安排好病房。疾病的严重程度用0到10来表示，0表示小毛病，10表示非常严重。  
   【输入】  
   第一行输入病人的个数m（m < 50），以及安排住入重症病房的最低严重值a。  
   紧接着m行，每行表示病人编号（三位，用0补齐）及其疾病的严重程度（浮点数，1位小数）。  
   【输出】  
   每个病人的疾病严重程度都不一样。输出要求按照病人的严重程度输出住在重症病房里的病人的编号。  
   【注意】  
   如果当前所有病人的严重程度并不满足住在重症病房里，则输出“None.”。  
   【输入示例】  
   10 7.55  
   006 6.5  
   005 8.0  
   004 3.5  
   009 8.5  
   011 7.0  
   043 9.5  
   003 5.0  
   103 6.0  
   112 4.0  
   118 9.0  
   【输出示例】  
   043 9.5  
   118 9.0  
   009 8.5  
   005 8.0  
   【提示】  
   可以定义如下结构类型：  
   struct Person {  
       int no;             // 病人的编号  
       double num;   // 病人病情严重程度  
   };（10分）

**我的答案：**

#include "algorithm"

bool compare(Person a,Person b){

return a.num>b.num;

}

int main(){

int N;

double num;

cin>>N>>num;

Person person[N];

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

cin>>person[i].no>>person[i].num;

}

sort(person,person+N,compare);

int flag=0;

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

if(person[i].num>=num){

flag=1;

cout<<setw(3)<<setfill('0')<<person[i].no<<" ";

cout<<fixed<<setprecision(1)<<person[i].num<<endl;

}

}

if(!flag) cout<<"None.";

return 0;

}

|  |  |
| --- | --- |
| 题目得分 |  |

**参考答案：**

#include <iostream>

#include <iomanip>

using namespace std;

struct Person {

int no; // 病人的编号

double num; // 病人病情严重程度

};

int main() {

Person p[55]; // 表示重症病人的结构数组

int m; // m个病人

double a; // 入住重症病房的最低严重值

int n; // 临时变量，表示当前输入的病人的编号

double t; // 临时变量，表示当前输入的病人的严重值

int len = 0; // 表示当前已经遇到的重症病人的人数

Person temp;

cin >> m >> a;

for(int i = 0; i < m; ++i) { // 读入m个人的信息并确定是否重症病人

cin >> n >> t; // 读入病人的编号和严重值

if(t > a) {

p[len].no=n;

p[len].num=t;

len++;

}

}

if(len==0) { // 假如没有重症病人

cout << "None." << endl;

return 0;

}

int flag;

for(int i = 1; i < len; ++i) { // 对重症病人按严重值排序 ，冒泡算法

flag = 1;

for(int j = 0; j < len - i; ++j) {

if(p[j].num < p[j+1].num) {

flag = 0;

temp = p[j];

p[j] = p[j+1];

p[j+1] = temp;

}

}

if(flag)

break;

}

for(int i = 0; i < len; ++i) // 按重症病人的严重值由大到小输出他们的信息。

cout << setw(3) << setfill( '0') << p[i].no << " "

<< fixed << setprecision(1) << p[i].num << endl;

return 0;

}

1. 【描述】  
   将输入的学生信息输出。  
   【输入】  
   第一行是整数t，表明数据组数。  
   在每组数据中：  
   第一行先是整数n(n<100),表示有n个学生。  
   接下来有n行，每行表示一个学生。先是一个无空格的字符串，表示姓名，然后是一个非负整数，表示学号。中间用单个空格隔开。  
   姓名长度不超过100字符，学号小于1000。  
   【输出】  
   按照输入的顺序，输出每个学生的信息。先输出学号，再输出姓名，中间用单个空格隔开。  
   一组数据处理完后，要输出一行 "\*\*\*\*"。  
   【样例输入】  
   2  
   3  
   Tom 12  
   Jack 20  
   Marry 89  
   2  
   Jade 78  
   White 76  
   【样例输出】  
   12 Tom  
   20 Jack  
   89 Marry  
   \*\*\*\*  
   78 Jade  
   76 White  
   \*\*\*\*（10分）

**我的答案：**

class Student{

public:

Student() {}

Student(const string &name, int score);

void Read(){

cin>> this->name;

cin>> this->score;

}

void Print(){

cout<< this->score<<" "<< this->name<<endl;

}

string name;

int score;

};

Student::Student(const string &name, int score) : name(name), score(score) {}

|  |  |
| --- | --- |
| 题目得分 |  |

**参考答案：**

#include <iostream>

#include <string>

using namespace std;

// 此处填写你的代码

class Student {

private:

string name;

int num;

public:

void Read() {

cin >> name >> num;

}

void Print() {

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

}

Student() { }

Student(string s, int n):name(s), num(n) { }

};

// 填写代码结束

int main() {

int t;

cin >> t;

Student s("Tom", 12);

while(t--) {

int n;

cin >> n;

Student st;

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

st.Read();

st.Print();

}

cout << "\*\*\*\*" << endl;

}

return 0;

}

1. 【描述】  
   小明是一个熊孩子，他很想知道某一个日期是星期几。他知道你正在学习C++，所以想请你写一个程序，实现一个Date类。Date类的定义如下：
2. class Date {
3. public:
4. Date(int year, int month, int day);
5. int getWeekday() const;
6. private:
7. int year;
8. int month;
9. int day;
10. };

成员变量year、month、day分别代表年月日。现在你要实现Date函数和getWeekday函数，并编写main函数。getWeekday函数，如果日期合法，返回1～7中某个数值，表示星期一到星期天中某一天（其中1为星期一），如果日期不合法，则返回-1。  
【输入】  
输入包含多组数据。每行包括三个整数year（2000≤y≤9999)、month、day。输入数据以0 0 0结束。  
【输出】  
每组数据输出一行，每行包括一个整数。1表示星期一，7表示星期日，等等，若数据非法，则输出-1。  
0 0 0不需输出  
【输入示例】  
2013 2 26  
2013 2 29  
0 0 0  
【输出示例】  
2  
-1  
【提示】2000年1月1日是星期六。  
求星期几：（6 + 2000年1月1日至输入日期的天数 - 1）% 7 + 1

（10分）

**我的答案：**

class Date{

public:

Date(int year, int month, int day);

int getWeekday(){

if((month>12||month<1)||(day>31 || day<1)){

return -1;

}

if(month==1||month==12){

month+=12;

year--;

}

int re=(day+2\*month+3\*(month+1)/5+year+year/4-year/100+year/400)%7;

re+=1;

return re;

}

private:

int year;

int month;

int day;

};

Date::Date(int year, int month, int day):year(year),month(month),day(day){}

|  |  |
| --- | --- |
| 题目得分 |  |

**参考答案：**

#include <iostream>

using namespace std;

class Date {

public:

Date(int year, int month, int day);

int getWeekday() const;

private:

int year;

int month;

int day;

};

bool isLeapYear(int year) {

return (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0);

}

Date::Date(int year, int month, int day) {

this->year = year;

this->month = month;

this->day = day;

}

int Date::getWeekday() const {

const int DAYS\_PER\_MONTH[13] = {

0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31

};

if(year < 2000 || year > 9999)

return -1;

if(month < 1 || month > 12)

return -1;

if(day < 1 || day > DAYS\_PER\_MONTH[month])

return -1;

int sum = 0;

// 处理年

for(int i = 2000; i < year; ++i) {

if(isLeapYear(i))

sum += 366;

else

sum += 365;

}

// 处理月

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

sum += DAYS\_PER\_MONTH[i];

if(isLeapYear(year))

if(month > 2 || month == 2)

sum += 1;

// 处理日

sum += day - 1; // 除去2000年1月1日

return (6 + sum - 1) % 7 + 1;

}

int main() {

int year, month, day;

while(cin >> year >> month >> day) {

if(year == 0 && month == 0 && day == 0)

break;

Date d(year, month, day);

cout << d.getWeekday() << endl;

}

return 0;

}

1. 【描述】  
   输入整数n（0<=n<=100000），以及整数i，j（0<=i、j<31，且i不等于j），输出整数k，要求：  
   ①k的第i位和n相同；  
   ②第j位和n不同；  
   ③其他位都是0。  
   这里提到的位，指的是将n表示成二进制数后的位，最低位（最右边）是第0位  
   【输入】  
   第一行是整数t，表示数据组数。  
   每组输入数据一行，包含三个整数n，i和j。  
   【输出】  
   对每组输入数据，输出一行：符合题目要求的整数k。  
   【样例输入】  
   2  
   23 4 3  
   3 0 1  
   【样例输出】  
   24  
   1

（10分）

**我的答案：**

#include "iostream"

#include "algorithm"

#include "iomanip"

#include "string"

#include "cmath"

#include "algorithm"

#include "vector"

using namespace std;

string int\_binary(int n){

string str="";

for(int i=n;i;i/=2){

str+=(i%2?'1':'0');

}

reverse(str.begin(),str.end());

str.insert(0,"00000000000000000000000000");

return str;

}

int binary\_int(string str) {

int sum = 0,k = 1,t;

for (int i = str.size() - 1;i >= 0;i--) {

t = str[i] - '0';

sum += k \* t;

k \*= 2;

}

return sum;

}

int main(){

int t;

cin>>t;

while (t--){

int n,i,j;

cin>>n>>i>>j;

string str\_n=int\_binary(n);

string str\_k;

int length=str\_n.length();

for(int i=0;i<str\_n.length();i++){

str\_k+='0';

}

str\_k[length-i-1]=str\_n[length-i-1];

str\_k[length-j-1] = str\_n[length-j-1] == '0' ? '1' : '0';

cout<<binary\_int(str\_k)<<endl;

}

return 0;

}

|  |  |
| --- | --- |
| 题目得分 |  |

**参考答案：**

#include <iostream>

using namespace std;

int main() {

int t;

cin >> t;

while(t--) {

int n, i, j;

cin >> n >> i >> j;

// 第j位和n不同

int m = ( 1 << j ) ^ n;

// 第i位和n相同

int k = ((1 << i) & n) | ((1 << j) & m);

cout << k << endl;

}

return 0;

}

1. 【描述】  
   输入若干学生信息，把他们的名字按成绩从低到高排序输出。  
   【输入】  
   第一行是整数 t，表示数据组数。  
   每组输入数据第一行是整数n，表示学生的数量。  
   接下来n行，每行是一个学生的信息。学生信息分为两部分，先是姓名，姓名由字母组成，没有空格。然后是个整数，表示分数。姓名和分数之间用一个空格隔开。  
   所有学生分数都不相同。姓名长度不超过100字符，分数范围从0到100。  
   【输出】  
   对每组输入数据，按分数从低到高输出学生姓名，每个姓名占一行。每组数据的最后输出一行“\*\*\*\*"。  
   【样例输入】  
   2  
   3  
   Tom 27  
   Jack 32  
   Jone 18  
   2  
   Will 91  
   Jane 97  
   【样例输出】  
   Jone  
   Tom  
   Jack  
   \*\*\*\*  
   Will  
   Jane  
   \*\*\*\*（10分）

**我的答案：**

/\* 请在此处编写Student类 \*/

/\* 请在此处编写Student类 \*/

class Student

{

private:

int score;

char name[100];

public:

bool operator <(Student& s)

{

return this->score < s.score;

};

friend ostream& operator <<(ostream& o, Student& s)

{

o << s.name;

return o;

};

friend istream& operator >>(istream& i, Student& s)

{

/\* 自行补充对输入数据合法性的判断 \*/

i >> s.name >> s.score;

return i;

}

};

|  |  |
| --- | --- |
| 题目得分 |  |

**参考答案：**

#include <iostream>

#include <string>

#include <algorithm>

#include <vector>

using namespace std;

// 此处填写你的代码

class Student {

public:

string name;

int score;

};

bool operator<(const Student &s1, const Student &s2) {

return s1.score < s2.score;

}

istream &operator>>(istream &i, Student &s) {

i >> s.name >> s.score;

return i;

}

ostream &operator<<(ostream &o,const Student &s) {

o << s.name;

return o;

}

// 填写代码结束

int main() {

int t;

vector<Student> v;

cin >> t;

while(t--) {

int n;

cin >> n;

Student st;

v.clear();

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

cin >> st;

v.push\_back(st);

}

sort(v.begin(),v.end());

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

cout << v[i] << endl;

cout << "\*\*\*\*" << endl;

}

return 0;

}

1. 【描述】  
   求一个正整数各位数字之和。要求定义和调用函数：int sumDigits(int n)，该函数返回一个正整数各位数字之和。  
   【输入】  
   输入一个正整数。  
   【输出】  
   输出该正整数各位数字之和。  
   【输入示例】  
   3456  
   【输出示例】  
   18  
   【来源】  
   《程序设计基础——以C++为例》第3章实验2强化练习。（10分）

**我的答案：**

int sumDigits(int n){

int sum=0;

while (n){

sum+=n%10;

n/=10;

}

return sum;

}

|  |  |
| --- | --- |
| 题目得分 |  |

**参考答案：**

#include <iostream>

using namespace std;

int sumDigits(int n);

int main() {

int n;

cin >> n;

cout << sumDigits(n) << endl;

return 0;

}

int sumDigits(int n) {

int remainder, result = 0;

while (n != 0) {

remainder = n % 10;

result += remainder;

n = n / 10;

}

return result;

}

1. 【描述】  
   下面是一个图书的单价表：  
   《计算概论》28.9元/本、《数据结构与算法》32.7元/本、《数字逻辑》45.6元/本、《C++程序设计教程》78元/本、《人工智能》35元/本、《计算机体系结构》86.2元/本、《编译原理》27.8元/本、《操作系统》43元/本、《计算机网络》56元/本、《Java程序设计》65元/本。  
   给定每种图书购买的数量，计算应付的总费用。  
   【输入】  
   第一行包含一个正整数n（0 < n <100），表示有n组测试数据；接下来n行，每行包含一组测试数据。每组测试数据包含10个整数（大于等于0、小于等于100），分别表示购买的《计算概论》、《数据结构与算法》、《数字逻辑》、《C++程序设计教程》、《人工智能》、《计算机体系结构》、《编译原理》、《操作系统》、《计算机网络》、《Java程序设计》的数量（以本为单位），整数以用空格间隔。  
   【输出】  
   对于每组测试数据，输出一行，表示应付的总费用，结果保留2位小数。  
   【输入示例】  
   2  
   1 5 8 10 5 1 1 2 3 4  
   3 5 6 3 100 1 1 0 1 0  
   【输出示例】  
   2140.20  
   4427.80  
   （10分）

**我的答案：**

#include "iostream"

#include "algorithm"

#include "iomanip"

#include "string"

#include "cmath"

#include "algorithm"

#include "vector"

using namespace std;

int main() {

int n;

double price[]={28.9,32.7,45.6,78,35,86.2,27.8,43,56,65};

int num[10];

cin>>n;

while (n--){

double sum=0;

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

cin>>num[i];

sum+=(num[i]\*price[i]);

}

cout<<fixed<<setprecision(2)<<sum<<endl;

}

return 0;

}

|  |  |
| --- | --- |
| 题目得分 |  |

**参考答案：**

#include <iostream>

#include <iomanip>

using namespace std;

int main() {

double prices[10] = {28.9, 32.7, 45.6, 78, 35, 86.2, 27.8, 43, 56, 65};

int n;

cin >> n;

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

double sum = 0, count;

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

cin >> count;

sum += count \* prices[i];

}

cout << fixed << setprecision(2) << sum << endl;

}

return 0;

}

1. 【描述】  
   对一个整数n，如果其各个位数的数字相加得到的数m能整除n，则称n为自整除数。例如21，21%(2+1)==0，所以21是自整除数。现求出从10到n（n<100）之间的所有自整除数。  
   【输入】  
   输入一个整数n（10≤n<100）。  
   【输出】  
   按从小到大的顺序输出所有大于等于10、小于等于n的自整除数，每行一个自整除数。  
   【输入示例】  
   47  
   【输出示例】  
   10  
   12  
   18  
   20  
   21  
   24  
   27  
   30  
   36  
   40  
   42  
   45（10分）

**我的答案：**

#include "iostream"

#include "algorithm"

#include "iomanip"

#include "string"

#include "cmath"

#include "algorithm"

#include "vector"

using namespace std;

int main() {

int n;

cin>>n;

for(int i=10;i<=n;i++){

int a,b;

int j=i;

a=i%10;

b=i/10;

if((j%(a+b))==0){

cout<<j<<endl;

}

}

return 0;

}

|  |  |
| --- | --- |
| 题目得分 |  |

**参考答案：**

#include<iostream>

using namespace std;

int main() {

int n;

cin >> n;

for(int i = 10; i <= n; i++){

int temp = i;

int sum = 0;

while(temp != 0) {

sum += temp % 10;

temp /= 10;

}

if(i % sum == 0)

cout << i << endl;

}

return 0;

}

1. 【描述】  
   用手机发短信，一般一条短信资费为0.1元，但限定每条短信的内容在70个字以内（包括70个字）。如果你所发送的一条短信超过了70个字，则大多数手机会按照每70个字一条短信的限制把它分割成多条短信发送。假设已经知道你当月所发送的每条短信的字数，试统计一下你当月短信的总资费。  
   【输入】  
   第一行是整数n，表示当月短信总条数，其余n行每行一个整数，表示各条短信的字数。  
   【输出】  
   当月短信总资费，单位为元，结果保留1位小数。  
   【输入示例】  
   10  
   39  
   49  
   42  
   61  
   44  
   147  
   42  
   72  
   35  
   46  
   【输出示例】  
   1.3（10分）

**我的答案：**

#include "iostream"

#include "algorithm"

#include "iomanip"

#include "string"

#include "cmath"

#include "algorithm"

#include "vector"

using namespace std;

int main(){

int n;

double sum=0;

cin>>n;

int a[n];

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

cin>>a[i];

}

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

while(a[i] > 0) {

sum+=0.1;

a[i] /= 70;

}

}

cout<<fixed<<setprecision(1)<<sum+0.1;

return 0;

}

|  |  |
| --- | --- |
| 题目得分 |  |

**参考答案：**

#include <iostream>

#include <iomanip>

#include <cmath>

using namespace std;

int main() {

int n;

cin >> n;

double sum = 0;

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

int word;

cin >> word;

if (word <= 70)

sum += 0.1;

else

sum += ceil(1.0 \* word / 70) \* 0.1;

}

cout << fixed << setprecision(1) << sum << endl;

return 0;

}

**《C++程序设计实践》实验十一**

**一、程序设计题（共5小题，100分）**

1. 【描述】  
   编写程序，输入若干个正整数，输入-1时输入结束，可以简化修改本章教材提供的链表类LinkedList，用单向链表组织输入的正整数。要求链表按照结点中整数值的大小从大到小排序，不包括最后标识结束的-1。输出单向链表。  
   【输入】  
   一系列正整数，输入-1表示结束，-1不是输入的数据的一部分。  
   【输出】  
   按照结点中整数值的大小从大到小输出所有的整数，每个整数后面跟一个空格以与后面的整数区分，最后的整数后面没有空格。  
   【输入示例】  
   1 3 5 2 -1  
   【输出示例】  
   5 3 2 1  
   【来源】  
   《程序设计基础——以C++为例》第9章实验3。

（20分）

**我的答案：**

template <typename T>

class Node {

public:

Node() {

next = NULL;

}

Node(const T &value) {

this->value = value;

next = NULL;

}

T value;

Node<T> \*next;

};

template <typename T>

class LinkedList {

public:

LinkedList();

void add(const T &value);

void print() const;

private:

Node<T> \*head;

int size;

};

template <typename T>

LinkedList<T>::LinkedList() {

head = NULL;

size = 0;

}

template <typename T>

void LinkedList<T>::add(const T &value) {

Node<T> \*temp = new Node<T>(value);

Node<T> \*current = head;

Node<T> \*previous = NULL;

while(current != NULL && current->value > value) {

previous = current;

current = current->next;

}

if(current == head)

head = temp;

else

previous->next = temp;

temp->next = current;

++size;

}

template <typename T>

void LinkedList<T>::print() const {

Node<T> \*current = head;

while(current != NULL) {

cout << current->value << " ";

current = current->next;

}

cout << endl;

}

|  |  |
| --- | --- |
| 题目得分 |  |

**参考答案：**

#include <iostream>

using namespace std;

template <typename T>

class Node {

public:

Node() {

next = NULL;

}

Node(const T &value) {

this->value = value;

next = NULL;

}

T value;

Node<T> \*next;

};

template <typename T>

class LinkedList {

public:

LinkedList();

void add(const T &value);

void print() const;

private:

Node<T> \*head;

int size;

};

template <typename T>

LinkedList<T>::LinkedList() {

head = NULL;

size = 0;

}

template <typename T>

void LinkedList<T>::add(const T &value) {

Node<T> \*temp = new Node<T>(value);

Node<T> \*current = head;

Node<T> \*previous = NULL;

while(current != NULL && current->value > value) {

previous = current;

current = current->next;

}

if(current == head)

head = temp;

else

previous->next = temp;

temp->next = current;

++size;

}

template <typename T>

void LinkedList<T>::print() const {

Node<T> \*current = head;

while(current != NULL) {

cout << current->value << " ";

current = current->next;

}

cout << endl;

}

int main() {

LinkedList<int> list;

int num;

cin >> num;

while(num > 0) {

list.add(num); // 输入的正整数按从大到小的顺序添加到链表中

cin >> num;

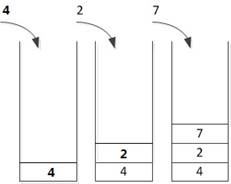
}

list.print(); // 输出链表

return 0;

}

1. **【**描述**】**栈可以应用于算术表达式求值。这里对算术表达式做如下简化：运算符为+、-、\*、/、%；操作数为单数字（0～9）非负整数。  
   例如：  
   (2+5)\*3-8/3  
   上述形式的算术表达式也称中缀表达式，因为每个运算符出现在它的两个操作数之间。编译器在求算术表达式值时，往往将中缀表达式转换为后缀表达式。后缀表达式也称逆波兰表达式，由波兰数学家Jan Lukasiewicz发明，指运算符出现在操作数后面的不含括号的算术表达式。  
   中缀表达式：a+b\*c，其后缀表达式为：a b c \* +。因为\*的优先级比+高，所以先计算b\*c，即b c \*；+的操作数是a和b c \*，即a b c \* +。  
   中缀表达式：(a+b)\*c，其后缀表达式为：a b + c \*。先计算圆括号中a+b，即a b +；\*的操作数是a b +和c，即a b + c \*。  
   中缀表达式：(a\*b+c)/d+e，其后缀表达式为：a b \* c + d / e +。先计算圆括号中a\*b+c，即a b \* c +；/的操作数是a b \* c +和d，即a b \* c + d /；+的操作数是a b \* c + d /和e，即a b \* c + d / e +。  
   后缀表达式求值使用一个存放操作数的栈。求值过程顺序扫描后缀表达式，每次遇到操作数就将它入栈；遇到运算符时，就从栈中弹出两个操作数进行计算，并将结果入栈。到扫描结束，留在栈顶的操作数就是所求算术表达式的值。  
   例如：  
   4 2 7 \*+  
   如下图一所示，顺序读取操作数4、2、7，并入栈。如下图二所示，读取运算符\*，操作数7、2出栈，计算2\*7，结果为14并入栈。如下图三所示，读取运算符+，操作数14、4出栈，计算4+14，结果为18并入栈。扫描后缀表达式后，栈中惟一元素即为最终结果。

****  
****  
****

本章教材组合向量类Vector实现了一个Stack类模板。请组合栈类Stack，声明并实现一个PostfixEvaluation类，求后缀表达式的值。PostfixEvaluation类包括：

* + string类型的私有数据成员postfixExpression，存放后缀表达式。
  + Stack类型的私有数据成员operandStack，存放操作数（单数字非负整数）。
  + 私有成员函数getOperands，连续从栈顶出栈两个操作数。
  + 私有成员函数calculate，求出栈的两个操作数的计算结果，并将结果入栈。
  + 私有成员函数isOperator，确定输入的运算符是有效运算符（+、-、\*、/、%）
  + 有参构造函数。新建一个后缀表达式对象。
  + 访问器函数getPostfixExpression，获取后缀表达式。
  + 更改器函数setPostfixExpression，设置新的后缀表达式。
  + 成员函数evaluate，计算并返回后缀表达式的值。

PostfixEvaluation类如下：  
class PostfixEvaluation {  
public:  
    PostfixEvaluation(const string&postfixExpression);                 // 构造函数  
    stringgetPostfixExpression() const;                                          // 获取后缀表达式  
    void setPostfixExpression(const string&postfixExpression);    // 设置后缀表达式  
    int evaluate();                            // 计算并返回后缀表达式值  
private:  
    string postfixExpression;           // 存放要计算的后缀表达式  
    Stack<int> operandStack;        // 存放操作数  
    void getOperands(int &left, int&right);             // 操作数出栈  
    int calculate(int left, int right, char op) const;   // 求操作数运算值  
    bool isOperator(char ch) const;                         // 是否是运算符  
};  
【输入】  
输入一个后缀表达式。  
【输出】  
输出后缀表达式的值。  
【输入示例1】  
4 2 7 \* +  
【输出示例1】  
18  
【输入示例2】  
8 4 0 % +  
【输出示例2】  
Divisor cannot be zero!  
【输入示例3】  
2 3 ^ 4 +  
【输出示例3】  
Illegal input!  
【输入示例4】  
4 2 / \*  
【输出示例4】  
Too many operators!  
【输入示例5】  
1 2 3 +  
【输出示例5】  
Too many operands!  
【来源】  
《程序设计基础——以C++为例》第9章实验2。

（20分）

**我的答案：**

class PostfixEvaluation {

public:

PostfixEvaluation(const string &postfixExpression); // 构造函数

string getPostfixExpression() const; // 获取后缀表达式

void setPostfixExpression(const string &postfixExpression); // 设置后缀表达式

int evaluate(); // 计算并返回后缀表达式值

private:

string postfixExpression; // 存放要计算的后缀表达式

Stack<int> operandStack; // 存放操作数

void getOperands(int &left, int &right); // 操作数出栈

int calculate(int left, int right, char op) const; // 求操作数运算值

bool isOperator(char ch) const; // 是否是运算符

};

void PostfixEvaluation::getOperands(int &left, int &right) {

if(operandStack.isEmpty())

throw runtime\_error("Too many operators!");

right = operandStack.pop();

if(operandStack.isEmpty())

throw runtime\_error("Too many operators!");

left = operandStack.pop();

}

int PostfixEvaluation::calculate(int left, int right, char op) const {

int value;

switch(op) {

case '+':value = left + right;

break;

case '-':value = left - right;

break;

case '\*':value = left \* right;

break;

case '/':if(right == 0)

throw runtime\_error("Divisor cannot be zero!");

value = left / right;

break;

case '%':if(right == 0)

throw runtime\_error("Divisor cannot be zero!");

value = left % right;

break;

}

return value;

}

bool PostfixEvaluation::isOperator(char ch) const {

return ch == '+' || ch == '-' || ch == '\*' || ch == '/' || ch == '%';

}

PostfixEvaluation::PostfixEvaluation(const string &postfixExpression) {

this->postfixExpression = postfixExpression;

}

string PostfixEvaluation::getPostfixExpression() const {

return postfixExpression;

}

void PostfixEvaluation::setPostfixExpression(const string &postfixExpression) {

this->postfixExpression = postfixExpression;

}

int PostfixEvaluation::evaluate() {

int left, right, value;

char ch;

for(string::size\_type i = 0; i < postfixExpression.size(); ++i) {

ch = postfixExpression[i];

if(isdigit(ch))

operandStack.push(ch - '0');

else if(isOperator(ch)) {

getOperands(left, right);

operandStack.push(calculate(left, right, ch));

}

else if(!isspace(ch))

throw runtime\_error("Illegal input!");

}

value = operandStack.pop();

if(!operandStack.isEmpty())

throw runtime\_error("Too many operands!");

return value;

}

|  |  |
| --- | --- |
| 题目得分 |  |

**参考答案：**

#include <iostream>

#include <stdexcept>

#include <string>

using namespace std;

template <typename T>

class Vector {

public:

Vector(int size); // 构造函数

Vector(int size, const T &value); // 构造函数

Vector(const Vector<T> &v); // 拷贝构造函数

virtual ~Vector(); // 析构函数

const Vector<T> &operator=(const Vector<T> &right); // 重载赋值运算符

T &operator[](int index); // 重载下标运算符

T operator[](int index) const; // 重载下标运算符

int getSize() const;

void resize(int size);

private:

T \*pVector; // 指针，指向存放数组元素的动态分配内存空间

int size; // 数组长度

};

template <typename T>

Vector<T>::Vector(int size) {

if(size > 0)

this->size = size;

else

throw invalid\_argument("数组长度必须是正整数!");

pVector = new T[size];

}

template <typename T>

Vector<T>::Vector(int size, const T &value) {

if(size > 0)

this->size = size;

else

throw invalid\_argument("数组长度必须是正整数!");

pVector = new T[size];

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

pVector[i] = value;

}

template <typename T>

Vector<T>::Vector(const Vector<T> &v) {

size = v.size;

pVector = new T[size];

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

pVector[i] = v.pVector[i];

}

template <typename T>

Vector<T>::~Vector() {

delete[] pVector;

}

template <typename T>

const Vector<T> &Vector<T>::operator=(const Vector<T> &right) {

if(this != &right) {

if(size != right.size) {

delete[] pVector;

size = right.size;

pVector = new T[size];

}

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

pVector[i] = right.pVector[i];

}

return \*this;

}

template <typename T>

T &Vector<T>::operator[](int index) {

if(index < 0 || index > size - 1)

throw out\_of\_range("数组下标超出允许范围!");

return pVector[index];

}

template <typename T>

T Vector<T>::operator[](int index) const {

if(index < 0 || index > size - 1)

throw out\_of\_range("数组下标超出允许范围!");

return pVector[index];

}

template <typename T>

int Vector<T>::getSize() const {

return size;

}

template <typename T>

void Vector<T>::resize(int size) {

if(size > 0) {

if(this->size != size) {

T \*old = pVector;

pVector = new T[size];

int newSize = (this->size > size) ? size : this->size;

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

pVector[i] = old[i];

this->size = size;

delete[] old;

}

}

else

throw invalid\_argument("数组长度必须是正整数!");

}

template <typename T>

class Stack {

public:

Stack(int size = 16); // 构造函数

Stack(const Stack<T> &v); // 拷贝构造函数

void clear(); // 将栈设置为空栈

bool isEmpty() const; // 判断栈是否为空

void push(T value); // 入栈

T pop(); // 出栈

T peek() const; // 获取栈顶元素

private:

Vector<T> data; // 存放栈元素

int top; // 记录栈顶位置

};

template <typename T>

Stack<T>::Stack(int size):data(size) {

clear();

}

template <typename T>

Stack<T>::Stack(const Stack<T> &v)

:data(v.data), top(v.top) { }

template <typename T>

void Stack<T>::clear() {

top = 0;

}

template <typename T>

bool Stack<T>::isEmpty() const {

return top == 0;

}

template <typename T>

void Stack<T>::push(T value) {

if(top >= data.getSize())

data.resize(2 \* data.getSize());

data[top++] = value;

}

template <typename T>

T Stack<T>::pop() {

return data[--top];

}

template <typename T>

T Stack<T>::peek() const {

return data[top - 1];

}

class PostfixEvaluation {

public:

PostfixEvaluation(const string &postfixExpression); // 构造函数

string getPostfixExpression() const; // 获取后缀表达式

void setPostfixExpression(const string &postfixExpression); // 设置后缀表达式

int evaluate(); // 计算并返回后缀表达式值

private:

string postfixExpression; // 存放要计算的后缀表达式

Stack<int> operandStack; // 存放操作数

void getOperands(int &left, int &right); // 操作数出栈

int calculate(int left, int right, char op) const; // 求操作数运算值

bool isOperator(char ch) const; // 是否是运算符

};

void PostfixEvaluation::getOperands(int &left, int &right) {

if(operandStack.isEmpty())

throw runtime\_error("Too many operators!");

right = operandStack.pop();

if(operandStack.isEmpty())

throw runtime\_error("Too many operators!");

left = operandStack.pop();

}

int PostfixEvaluation::calculate(int left, int right, char op) const {

int value;

switch(op) {

case '+':value = left + right;

break;

case '-':value = left - right;

break;

case '\*':value = left \* right;

break;

case '/':if(right == 0)

throw runtime\_error("Divisor cannot be zero!");

value = left / right;

break;

case '%':if(right == 0)

throw runtime\_error("Divisor cannot be zero!");

value = left % right;

break;

}

return value;

}

bool PostfixEvaluation::isOperator(char ch) const {

return ch == '+' || ch == '-' || ch == '\*' || ch == '/' || ch == '%';

}

PostfixEvaluation::PostfixEvaluation(const string &postfixExpression) {

this->postfixExpression = postfixExpression;

}

string PostfixEvaluation::getPostfixExpression() const {

return postfixExpression;

}

void PostfixEvaluation::setPostfixExpression(const string &postfixExpression) {

this->postfixExpression = postfixExpression;

}

int PostfixEvaluation::evaluate() {

int left, right, value;

char ch;

for(string::size\_type i = 0; i < postfixExpression.size(); ++i) {

ch = postfixExpression[i];

if(isdigit(ch))

operandStack.push(ch - '0');

else if(isOperator(ch)) {

getOperands(left, right);

operandStack.push(calculate(left, right, ch));

}

else if(!isspace(ch))

throw runtime\_error("Illegal input!");

}

value = operandStack.pop();

if(!operandStack.isEmpty())

throw runtime\_error("Too many operands!");

return value;

}

int main() {

string expression;

getline(cin, expression);

PostfixEvaluation exp(expression);

try {

cout << exp.evaluate() << endl;

}

catch(runtime\_error &ex) {

cout << ex.what() << endl;

}

return 0;

}

1. 【描述】  
   本章教材组合向量类Vector实现了一个Stack类模板。也可以用链表的方式实现栈。利用本章教材提供的链表类LinkedList，组合链表类LinkedList来实现一个新的Stack类模板。  
   template <typename T>  
   class Stack {  
   public:  
       Stack();  
       void clear();  
       bool isEmpty() const;  
       void push(const T &value);  
       T pop();  
       T peek() const;  
   private:  
       LinkedList<T> data;  
   };  
   【输入】  
   一系列正整数并入栈，输入-1表示结束，-1不是输入的数据的一部分。  
   【输出】  
   输出栈中所有的整数，每个整数后面跟一个空格以与后面的整数区分。  
   【输入示例】  
   1 3 5 2 -1  
   【输出示例】  
   2 5 3 1  
   【来源】  
   《程序设计基础——以C++为例》第9章实验4。

（20分）

**我的答案：**

template <typename T>

class Stack {

public:

Stack();

void clear();

bool isEmpty() const;

void push(const T &value);

T pop();

T peek() const;

private:

LinkedList<T> data;

};

template <typename T>

Stack<T>::Stack() { }

template <typename T>

void Stack<T>::clear() {

data.clear();

}

template <typename T>

bool Stack<T>::isEmpty() const {

return data.isEmpty();

}

template <typename T>

void Stack<T>::push(const T &value) {

data.addLast(value);

}

template <typename T>

T Stack<T>::pop() {

T temp = data.getLast();

data.removeLast();

return temp;

}

template <typename T>

T Stack<T>::peek() const {

return data.getLast();

}

|  |  |
| --- | --- |
| 题目得分 |  |

**参考答案：**

#include <iostream>

#include <stdexcept>

using namespace std;

template <typename T>

class Node {

public:

Node() {

next = NULL;

}

Node(const T &value) {

this->value = value;

next = NULL;

}

T value;

Node<T> \*next;

};

template <typename T>

class LinkedList {

public:

LinkedList();

void addFirst(const T &value);

void addLast(const T &value);

void add(int index, const T &value);

void removeFirst();

void removeLast();

void removeAt(int index);

T getFirst() const;

T getLast() const;

void clear();

bool isEmpty() const;

int getSize() const;

void print() const;

private:

Node<T> \*head, \*tail;

int size;

};

template <typename T>

LinkedList<T>::LinkedList() {

head = tail = NULL;

size = 0;

}

template <typename T>

void LinkedList<T>::addFirst(const T &value) {

Node<T> \*temp = new Node<T>(value);

temp->next = head;

head = temp;

++size;

if(tail == NULL)

tail = head;

}

template <typename T>

void LinkedList<T>::addLast(const T &value) {

if(tail == NULL)

head = tail = new Node<T>(value);

else {

tail->next = new Node<T>(value);

tail = tail->next;

}

++size;

}

template <typename T>

void LinkedList<T>::add(int index, const T &value) {

if(index == 0)

addFirst(value);

else if(index >= size)

addLast(value);

else {

Node<T> \*current = head;

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

current = current->next;

Node<T> \*temp = current->next;

current->next = new Node<T>(value);

current->next->next = temp;

++size;

}

}

template <typename T>

void LinkedList<T>::removeFirst() {

if(head == NULL)

throw runtime\_error("空链表!");

else {

Node<T> \*temp = head;

head = head->next;

if(head == NULL)

tail = NULL;

delete temp;

--size;

}

}

template <typename T>

void LinkedList<T>::removeLast() {

if(tail == NULL)

throw runtime\_error("空链表!");

else if(head == tail) {

Node<T> \*temp = head;

head = tail = NULL;

size = 0;

delete temp;

}

else {

Node<T> \*current = head;

while(current->next != tail)

current = current->next;

Node<T> \*temp = tail;

tail = current;

tail->next = NULL;

delete temp;

--size;

}

}

template <typename T>

void LinkedList<T>::removeAt(int index) {

if(index < 0 || index >= size)

throw runtime\_error("下标越界!");

else if(index == 0)

return removeFirst();

else if(index == size - 1)

return removeLast();

else {

Node<T> \*previous = head;

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

previous = previous->next;

Node<T> \*current = previous->next;

previous->next = current->next;

delete current;

--size;

}

}

template <typename T>

T LinkedList<T>::getFirst() const {

if(head == NULL)

throw runtime\_error("空链表!");

else

return head->value;

}

template <typename T>

T LinkedList<T>::getLast() const {

if(tail == NULL)

throw runtime\_error("空链表!");

else

return tail->value;

}

template <typename T>

void LinkedList<T>::clear() {

while(head != NULL) {

Node<T> \*temp = head;

head = head->next;

delete temp;

}

tail = NULL;

}

template <typename T>

bool LinkedList<T>::isEmpty() const {

return head == NULL;

}

template <typename T>

int LinkedList<T>::getSize() const {

return size;

}

template <typename T>

void LinkedList<T>::print() const {

Node<T> \*current = head;

while(current != NULL) {

cout << current->value << " ";

current = current->next;

}

cout << endl;

}

template <typename T>

class Stack {

public:

Stack();

void clear();

bool isEmpty() const;

void push(const T &value);

T pop();

T peek() const;

private:

LinkedList<T> data;

};

template <typename T>

Stack<T>::Stack() { }

template <typename T>

void Stack<T>::clear() {

data.clear();

}

template <typename T>

bool Stack<T>::isEmpty() const {

return data.isEmpty();

}

template <typename T>

void Stack<T>::push(const T &value) {

data.addLast(value);

}

template <typename T>

T Stack<T>::pop() {

T temp = data.getLast();

data.removeLast();

return temp;

}

template <typename T>

T Stack<T>::peek() const {

return data.getLast();

}

int main() {

Stack<int> intStack;

int num;

cin >> num;

while(num != -1) {

intStack.push(num);

cin >> num;

}

while(!intStack.isEmpty())

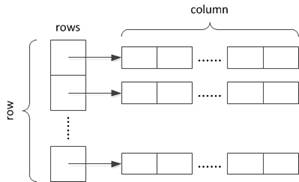
cout << intStack.pop() << " ";

cout << endl;

return 0;

}

1. 【描述】  
   矩阵是数学里的一种抽象对象，可以用C++提供的静态数组来表示矩阵，其大小在编译时就已经确定，在运行时无法修改，而且不检查下标是否越界。可以利用教材本章提供的向量类Vector，用向量的方式实现矩阵，用一个指针向量来表示矩阵，其中的每个指针又各指向一个向量，用它们来表示矩阵的行向量。矩阵的逻辑结构如下图所示。

****

组合向量类Vector，声明并实现一个Matrix类模板，表示矩阵。Matrix类模板包括：

* + Vector类型的私有数据成员rows，存放矩阵元素。
  + 构造函数，将矩阵的行、列大小设置为给定的参数。
  + 构造函数，将矩阵的行、列大小设置为给定的参数，给矩阵元素赋相同的初始值。
  + 重载下标运算符[]。
  + 访问器函数getRows，用于获取矩阵行数。
  + 访问器函数getColumns，用于获取矩阵列数。

Matrix类模板如下：  
template <typename T>  
class Matrix {  
public:  
    Matrix(int row, int column);  
    Matrix(int row, int column, const T &value);  
    Vector<T> &operator[](int index);  
    Vector<T> operator[](int index) const;  
    int getRows() const;  
    int getColumns() const;  
private:  
    Vector<Vector<T>\*> rows;        // 存放矩阵元素  
};

以普通函数的形式重载\*运算符函数，实现矩阵乘法。  
template <typename T>  
Matrix<T> operator\*(const Matrix<T> &lhs, const Matrix<T> &rhs);  
  
printMatrix函数输出矩阵的值。  
template <typename T>  
void printMatrix(const Matrix<T> &m);  
【输入】  
输入3×3矩阵a和矩阵b。  
【输出】  
矩阵a乘以矩阵b的结果。每个元素输出宽度为4。  
【输入示例】  
1 1 7  
7 5 6  
9 6 1  
6 1 6  
4 1 5  
1 5 1  
【输出示例】

17 37 18

68 42 73

79 20 85

【来源】  
《程序设计基础——以C++为例》第9章实验1。

（20分）

**我的答案：**

template <typename T>

class Matrix {

public:

Matrix(int row, int column); // 构造函数

Matrix(int row, int column, const T &value); // 构造函数

Vector<T> &operator[](int index); // 重载下标运算符

Vector<T> operator[](int index) const; // 重载下标运算符

int getRows() const;

int getColumns() const;

private:

Vector<Vector<T>\*> rows; // 存放矩阵元素

};

template <typename T>

Matrix<T>::Matrix(int row, int column):rows(row) {

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

rows[i] = new Vector<T>(column);

}

template <typename T>

Matrix<T>::Matrix(int row, int column, const T &value):rows(row) {

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

rows[i] = new Vector<T>(column, value);

}

template <typename T>

Vector<T> &Matrix<T>::operator[](int index) {

return \*rows[index];

}

template <typename T>

Vector<T> Matrix<T>::operator[](int index) const {

return \*rows[index];

}

template <typename T>

int Matrix<T>::getRows() const {

return rows.getSize();

}

template <typename T>

int Matrix<T>::getColumns() const {

return rows[0]->getSize();

}

template <typename T>

Matrix<T> operator\*(const Matrix<T> &lhs, const Matrix<T> &rhs) {

int lhsRow = lhs.getRows();

int lhsColumn = lhs.getColumns();

int rhsColumn = rhs.getColumns();

Matrix<T> result(lhsRow, rhsColumn, 0);

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

for(int j = 0; j < rhsColumn; ++j) {

for(int k = 0; k < lhsColumn; ++k)

result[i][j] += lhs[i][k] \* rhs[k][j];

}

}

return result;

}

template <typename T>

void printMatrix(const Matrix<T> &m) {

int row = m.getRows();

int column = m.getColumns();

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

for(int j = 0; j < column; ++j)

cout << setw(4) << m[i][j];

cout << endl;

}

}

|  |  |
| --- | --- |
| 题目得分 |  |

**参考答案：**

#include <iostream>

#include <iomanip>

#include <stdexcept>

using namespace std;

template <typename T>

class Vector {

public:

Vector(int size); // 构造函数

Vector(int size, const T &value); // 构造函数

Vector(const Vector<T> &v); // 拷贝构造函数

virtual ~Vector(); // 析构函数

const Vector<T> &operator=(const Vector<T> &right); // 重载赋值运算符

T &operator[](int index); // 重载下标运算符

T operator[](int index) const; // 重载下标运算符

int getSize() const;

void resize(int size);

private:

T \*pVector; // 指针，指向存放数组元素的动态分配内存空间

int size; // 数组长度

};

template <typename T>

Vector<T>::Vector(int size) {

if(size > 0)

this->size = size;

else

throw invalid\_argument("数组长度必须是正整数!");

pVector = new T[size];

}

template <typename T>

Vector<T>::Vector(int size, const T &value) {

if(size > 0)

this->size = size;

else

throw invalid\_argument("数组长度必须是正整数!");

pVector = new T[size];

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

pVector[i] = value;

}

template <typename T>

Vector<T>::Vector(const Vector<T> &v) {

size = v.size;

pVector = new T[size];

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

pVector[i] = v.pVector[i];

}

template <typename T>

Vector<T>::~Vector() {

delete[] pVector;

}

template <typename T>

const Vector<T> &Vector<T>::operator=(const Vector<T> &right) {

if(this != &right) {

if(size != right.size) {

delete[] pVector;

size = right.size;

pVector = new T[size];

}

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

pVector[i] = right.pVector[i];

}

return \*this;

}

template <typename T>

T &Vector<T>::operator[](int index) {

if(index < 0 || index > size - 1)

throw out\_of\_range("数组下标超出允许范围!");

return pVector[index];

}

template <typename T>

T Vector<T>::operator[](int index) const {

if(index < 0 || index > size - 1)

throw out\_of\_range("数组下标超出允许范围!");

return pVector[index];

}

template <typename T>

int Vector<T>::getSize() const {

return size;

}

template <typename T>

void Vector<T>::resize(int size) {

if(size > 0) {

if(this->size != size) {

T \*old = pVector;

pVector = new T[size];

int newSize = (this->size > size) ? size : this->size;

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

pVector[i] = old[i];

this->size = size;

delete[] old;

}

}

else

throw invalid\_argument("数组长度必须是正整数!");

}

template <typename T>

class Matrix {

public:

Matrix(int row, int column); // 构造函数

Matrix(int row, int column, const T &value); // 构造函数

Vector<T> &operator[](int index); // 重载下标运算符

Vector<T> operator[](int index) const; // 重载下标运算符

int getRows() const;

int getColumns() const;

private:

Vector<Vector<T>\*> rows; // 存放矩阵元素

};

template <typename T>

Matrix<T>::Matrix(int row, int column):rows(row) {

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

rows[i] = new Vector<T>(column);

}

template <typename T>

Matrix<T>::Matrix(int row, int column, const T &value):rows(row) {

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

rows[i] = new Vector<T>(column, value);

}

template <typename T>

Vector<T> &Matrix<T>::operator[](int index) {

return \*rows[index];

}

template <typename T>

Vector<T> Matrix<T>::operator[](int index) const {

return \*rows[index];

}

template <typename T>

int Matrix<T>::getRows() const {

return rows.getSize();

}

template <typename T>

int Matrix<T>::getColumns() const {

return rows[0]->getSize();

}

template <typename T>

Matrix<T> operator\*(const Matrix<T> &lhs, const Matrix<T> &rhs) {

int lhsRow = lhs.getRows();

int lhsColumn = lhs.getColumns();

int rhsColumn = rhs.getColumns();

Matrix<T> result(lhsRow, rhsColumn, 0);

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

for(int j = 0; j < rhsColumn; ++j) {

for(int k = 0; k < lhsColumn; ++k)

result[i][j] += lhs[i][k] \* rhs[k][j];

}

}

return result;

}

template <typename T>

void printMatrix(const Matrix<T> &m) {

int row = m.getRows();

int column = m.getColumns();

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

for(int j = 0; j < column; ++j)

cout << setw(4) << m[i][j];

cout << endl;

}

}

int main() {

const int ROW\_SIZE = 3;

const int COLUMN\_SIZE = 3;

Matrix<int> a(ROW\_SIZE, COLUMN\_SIZE);

Matrix<int> b(ROW\_SIZE, COLUMN\_SIZE);

for(int i = 0; i < ROW\_SIZE; ++i) {

for(int j = 0; j < COLUMN\_SIZE; ++j) {

cin >> a[i][j];

}

}

for(int i = 0; i < ROW\_SIZE; ++i) {

for(int j = 0; j < COLUMN\_SIZE; ++j) {

cin >> b[i][j];

}

}

printMatrix(a \* b);

return 0;

}

1. **【**描述**】**给定一个链表，链表的每个节点只含有一个int型元素和Node\*指针，该链表共有十个节点，输出链表元素值为奇数的项。部分代码已经写好，请补全output函数即可。  
   **【**输入**】**输入10个整数。  
   **【**输出**】**输出奇数值，以空格隔开。  
   **【**输入示例**】**1 3 4 5 6 7 8 10 11 15  
   **【**输出示例**】**1 3 5 7 11 15

（20分）

**我的答案：**

Node \*p = head;

while(p != NULL) {

if(p->value % 2 != 0)

cout << p->value << " ";

p = p->next;

}

|  |  |
| --- | --- |
| 题目得分 |  |

**参考答案：**

#include <iostream>

using namespace std;

class Node {

public:

int value;

Node \*next;

Node() { }

Node(int x) : value(x), next(NULL) { }

};

class SingleLinkedList {

public:

SingleLinkedList() {

head = NULL;

}

void init() { // 链表初始化

int val;

cin >> val;

head = new Node(val);

Node \*tmpNode = head;

for(int i = 0; i < 9; i++) { // 链表共有10个节点，head后还有9个

cin >> val;

Node \*pNode = new Node(val);

tmpNode->next = pNode;

tmpNode = pNode;

}

tmpNode->next = NULL;

}

void output() { // 输出链表

Node \*p = head;

while(p != NULL) {

if(p->value % 2 != 0)

cout << p->value << " ";

p = p->next;

}

}

private:

Node \*head;

};

int main() {

SingleLinkedList list;

list.init();

list.output();

return 0;

}

**《C++程序设计实践》实验十**

班级：

学号： 2019212212113

姓名： 覃旭日

得分：  0

**一、程序设计题（共10小题，100分）**

|  |  |
| --- | --- |
| 题型得分 | 0 |

1. 【描述】  
   以二进制方式打开图片文件并读取该文件中的第 13（从1开始计数，后同）,49, 80 个字节处的值，求这3个二进制数按位异或的结果（16进制表示）。  
   可以鼠标右键另存为下载图片文件：



【输入】  
图片文件image.jpg。（该文件已经存在，无需自己创建）  
【输出】  
第 13（从1开始计数，后同）,49, 80个字节处3个二进制数按位异或的结果（16进制表示）。  
【输入示例】  
图片文件image.jpg。  
【输出示例】  
a9  
【提示】  
输出示例只是格式说明，并非正确答案。

（10分）

**我的答案：**

#include "iostream"

#include "fstream"

using namespace std;

int main(){

ifstream fin("image.jpg",ios::binary);

char byte1,byte2,byte3;

fin.seekg(12,ios::beg);

fin.read((char \*) &byte1,1);

fin.seekg(48,ios::beg);

fin.read((char\*)&byte2,1);

fin.seekg(79,ios::beg);

fin.read((char\*)&byte3,1);

cout<<hex<<((byte1^byte2^byte3)&0x000000ff)<<endl;

return 0;

}

|  |  |
| --- | --- |
| 题目得分 |  |

**参考答案：**

#include <iostream>

#include <fstream>

using namespace std;

int main() {

ifstream fin("image.jpg", ios::binary);

char byte1, byte2, byte3;

fin.seekg(12, ios::beg);

fin.read((char \*)&byte1, 1);

fin.seekg(48, ios::beg);

fin.read((char \*)&byte2, 1);

fin.seekg(79, ios::beg);

fin.read((char \*)&byte3, 1);

cout << hex << ((byte1 ^ byte2 ^ byte3) & 0x000000ff) << endl;

return 0;

}

1. 【描述】  
   需要计算一些学生的加权平均分。给定输入文件gpa.dat，文件中每个学生的信息都由两行内容组成。第一行是学生姓名，第二行是他几门课的成绩。下面是某个输入文件gpa.dat的内容：  
   Zhang San  
   3 2.8 4 3.9 3 3.1  
   Li Si  
   3 3.9 3 4.0 4 3.9  
   Wang Wu  
   2 4.0 3 3.6 4 3.8 1 2.8  
   Liu Liu  
   3 3.0 4 2.9 3 3.2 2 2.5  
   例如，张三（Zhang San）同学：第一门课，学分3，成绩2.8；第二门课，学分4，成绩3.9；第三门课，学分3，成绩3.1。  
   总平均分等于学分乘以成绩，加权平均分等于总平均分除以总学分数。加权平均分最低0.0，最高4.0。  
   【输入】  
   文件gpa.dat。（该文件已经存在，无需自己创建）  
   注意：本地调试程序时，则要求自己预先建立好gpa.dat文件。在Windows下，可以使用记事本。  
   【输出】  
   在屏幕上显示每个学生的加权平均分以及加权平均分的最大和最小值。  
   【输入示例】  
   文件gpa.dat，注意：这里只给出了其中部分内容。  
   Zhang San  
   3 2.8 4 3.9 3 3.1  
   Li Si  
   3 3.9 3 4.0 4 3.9  
   Wang Wu  
   2 4.0 3 3.6 4 3.8 1 2.8  
   Liu Liu  
   3 3.0 4 2.9 3 3.2 2 2.5  
   【输出示例】  
   GPA for Zhang San = 3.33  
   GPA for Li Si = 3.93  
   GPA for Wang Wu = 3.68  
   GPA for Liu Liu = 2.93  
   max GPA = 3.93  
   min GPA = 2.93

（10分）

**我的答案：**

#include "iostream"

#include "fstream"

#include "string"

#include "sstream"

#include "cstdlib"

#include "iomanip"

using namespace std;

double computeAverage(stringstream & str){

double AverageTotal=0,weightingTotal=0;

double average,weight;

while (str>>average>>weight){

weightingTotal+=average;

AverageTotal+=average\*weight;

}

if(weightingTotal==0){

return 0;

}

else{

return AverageTotal/weightingTotal;

}

}

void compute(ifstream &file){

double max=0;

double min=4;

while (!file.eof()){

string name;

string grades;

getline(file,name);

getline(file,grades);

stringstream str(grades);

double gpa=computeAverage(str);

cout<<"GPA for "<<name<<" = "<<fixed

<<setprecision(2)<<gpa<<endl;

if(gpa>max) max=gpa;

if(gpa<min) min=gpa;

}

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

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

}

int main(){

ifstream file;

file.open("gpa.dat");

compute(file);

return 0;

}

|  |  |
| --- | --- |
| 题目得分 |  |

**参考答案：**

#include <iostream>

#include <iomanip>

#include <fstream>

#include <string>

#include <sstream>

#include <cstdlib>

using namespace std;

void process(ifstream &inFile);

double processGrades(stringstream &myStream);

int main() {

ifstream inFile;

inFile.open("gpa.dat");

if (!inFile.is\_open())

exit(EXIT\_FAILURE);

process(inFile);

return 0;

}

void process(ifstream &inFile) {

double max = 0.0;

double min = 4.0;

while (!inFile.eof()) {

string name, grades;

getline(inFile, name);

getline(inFile, grades);

stringstream myStream(grades);

double gpa = processGrades(myStream);

cout << "GPA for " << name << " = "

<< fixed << setprecision(2) << gpa << endl;

if (gpa > max)

max = gpa;

if (gpa < min)

min = gpa;

}

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

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

}

double processGrades(stringstream &myStream) {

double totalQualityPoints = 0.0; // 总平均分

double totalUnits = 0.0; // 总学分数

double units, grade;

while (myStream >> units >> grade) {

totalUnits += units;

totalQualityPoints += units \* grade;

}

if (totalUnits == 0.0)

return 0.0;

else

return totalQualityPoints / totalUnits;

}

1. 【描述】  
   输入10个整数存入文本文件example.txt中，文件每行存放5个整数，每行整数之间用一个空格间隔。行末不能有多余的空格。  
   【输入】  
   输入10个整数。  
   【输出】  
   生成文件example.txt，里面存放输入的10个整数。  
   不需要在屏幕上显示整数。  
   【输入示例】  
   1 2 3 4 5 6 7 8 9 10  
   【输出示例】  
   生成文件example.txt，其中内容：  
   1 2 3 4  
   6 7 8 9 10  
   【来源】  
   《程序设计基础——以C++为例》第8章实验1。

（10分）

**我的答案：**

#include "iostream"

#include "fstream"

#include "string"

#include "sstream"

#include "cstdlib"

#include "iomanip"

using namespace std;

int main(){

ofstream file;

file.open("example.txt");

int count=1;

int n;

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

cin>>n;

file<<n;

if(count++ %5==0){

file<<endl;

}else{

file<<" ";

}

}

file.close();

return 0;

}

|  |  |
| --- | --- |
| 题目得分 |  |

**参考答案：**

#include <iostream>

#include <fstream>

#include <cstdlib>

using namespace std;

int main() {

ofstream outFile;

outFile.open("example.txt");

if(!outFile.is\_open()) {

exit(EXIT\_FAILURE);

}

int count = 1;

int x;

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

cin >> x;

outFile << x;

if(count++ % 5 == 0)

outFile << endl;

else

outFile << " ";

}

outFile.close();

return 0;

}

1. 【描述】  
   声明并实现一个Student类，表示学生信息。Student类包括：  
   int类型的私有数据成员num，表示学号。  
   string类型的私有数据成员name，表示姓名。  
   int类型的私有数据成员score，表示成绩  
   char类型的私有数据成员grade，表示等级。  
   无参（默认）构造函数。  
   有参构造函数，将学号、姓名、成绩和等级设置为给定的参数。  
   访问器函数getNum、getName、getScore、getGrade，分别用于访问学号、姓名、成绩和等级。  
   重载流提取运算符>>和流插入运算符<<。输入输出一个Student对象  
   输入若干个学生的信息（学号、姓名和成绩），学号为0时，输入结束，根据成绩计算出对应等级。假设90分以上的成绩属于A级；80～89分、70～79分、60～69分的成绩分别属于B、C、D级；60分以下属于E级。创建Student对象，将它们写入文本文件student.txt中。  
   【输入】  
   输入若干个学生的信息。  
   每行一个学生信息，学号、姓名和成绩之间以空格间隔。  
   学号为0时，输入结束。  
   【输出】  
   文件student.txt。  
   不需要在屏幕上显示信息。  
   【输入示例】  
   100101 ZhangSan 78  
   100102 LiSi 67  
   100103 WangWu 83  
   100104 LiuLiu 45  
   100105 QianQi 93  
   0  
   【输出示例】  
   生成文件student.txt，其中内容：  
   100101 ZhangSan 78 C  
   100102 LiSi 67 D  
   100103 WangWu 83 B  
   100104 LiuLiu 45 E  
   100105 QianQi 93 A  
   【来源】  
   《程序设计基础——以C++为例》第8章实验3。（10分）

**我的答案：**

#include "iostream"

#include "fstream"

#include "string"

#include "sstream"

#include "cstdlib"

#include "iomanip"

using namespace std;

class Student{

public:

Student(){}

Student(int num,string name,int score,char grade){

this->name=name;

this->num=num;

this->score=score;

this->grade=grade;

}

int getNum() const{

return num;

}

string getName()const{

return name;

}

int getScore()const{

return score;

}

char getGrader()const{

return grade;

}

friend ostream &operator<<(ostream &out,const Student &student);

friend istream &operator>>(istream &in,Student &student);

private:

int num;

string name;

int score;

char grade;

};

ostream &operator<<(ostream &out,const Student &student){

out<<student.num<<" "<<student.name<<" "

<<student.score<<" "<<student.grade<<endl;

}

istream &operator>>(istream &in,Student &student){

in>>student.num>>student.name>>student.score>>student.grade;

return in;

}

int main(){

ofstream file;

file.open("student.txt");

int num;

string name;

int score;

char grade;

cin>>num;

while (num){

cin>>name>>score;

switch (score / 10) {

case 10:

case 9:

grade='A';

break;

case 8:

grade='B';

break;

case 7:

grade='C';

break;

case 6:

grade='D';

break;

default:

grade='E';

break;

}

Student student(num,name,score,grade);

file<<student;

cin>>num;

}

return 0;

}

|  |  |
| --- | --- |
| 题目得分 |  |

**参考答案：**

#include <iostream>

#include <fstream>

#include <string>

#include <cstdlib>

using namespace std;

class Student {

public:

Student() { }

Student(int num, string name, int score, char grade) {

this->num = num;

this->name = name;

this->score = score;

this->grade = grade;

}

int getNum() const {

return num;

}

string getName() const {

return name;

}

int getScore() const {

return score;

}

char getGrade() const {

return grade;

}

friend ostream &operator<<(ostream &out, const Student &student);

friend istream &operator>>(istream &in, Student &student);

private:

int num; // 学号

string name;// 姓名

int score; // 成绩

char grade; // 等级

};

ostream &operator<<(ostream &out, const Student &student) {

out << student.num << " " << student.name << " "

<< student.score << " " << student.grade << endl;

return out;

}

istream &operator>>(istream &in, Student &student) {

in >> student.num >> student.name >> student.score >> student.grade;

return in;

}

int main() {

ofstream outFile;

outFile.open("student.txt");

if(!outFile.is\_open()) {

exit(EXIT\_FAILURE);

}

int num;

string name;

int score;

char grade;

cin >> num;

while(num != 0) {

cin >> name >> score;

switch(score / 10) {

case 10:

case 9: grade = 'A';

break;

case 8: grade = 'B';

break;

case 7: grade = 'C';

break;

case 6: grade = 'D';

break;

default: grade = 'E';

break;

}

Student student(num, name, score, grade);

outFile << student;

cin >> num;

}

outFile.close();

return 0;

}

1. 【描述】  
   给定文件hours.txt，其中包含每个员工工作时间的记录。每一行表示一周的工作时间。每周有7天，所以每行最多有7个数字。规定每周从周一开始，文件中的每一行都是从周一的工作小时数开始，后面是周二，等等，周日的数字放在这一行的最后。每行中的数字可以少于7个，因为员工并不是每天都要工作。下面是文件hours.txt的内容：  
   8 8 8 8 8  
   8 4 8 4 8 4 4  
   8 4 8 4 8  
   3 0 0 8 6 4 4  
   8 8  
   0 0 8 8  
   8 8 4 8 4  
   编写一个程序从输入文件中读取数据，计算并报告每行和每列的总和。每行的总和表示该员工每周工作的小时数。每列的总和表示员工周一、周二等每天工作的累计小时数。最后输出总的小时数。针对上述文件hours.txt的输出结果见【输出示例】。  
   【输入】  
   文件hours.txt。（该文件已经存在，无需自己创建）  
   注意：本地调试程序时，则要求自己预先建立好hours.txt文件。在Windows下，可以使用记事本。  
   【输出】  
   员工每周工作的小时数。  
   员工周一、周二等每天工作的累计小时数。  
   最后输出总的小时数。  
   【输入示例】  
   文件hours.txt。  
   【输出示例】  
   Total hours = 40  
   Total hours = 40  
   Total hours = 32  
   Total hours = 25  
   Total hours = 16  
   Total hours = 16  
   Total hours = 32  
     
   Mon hours = 43  
   Tue hours = 32  
   Wed hours = 36  
   Thu hours = 40  
   Fri hours = 34  
   Sat hours = 8  
   Sun hours = 8  
   Total hours = 201  
   【提示】  
   文件hours.txt中数据组数是不定的。

（10分）

**我的答案：**

#include "iostream"

#include "fstream"

#include "string"

#include "sstream"

#include "cstdlib"

#include "iomanip"

using namespace std;

void compute(ifstream &file);

int change(int numbers[], const string &line);

int main(){

ifstream file;

file.open("hours.txt");

compute(file);

return 0;

}

void compute(ifstream &file){

int total[7]={0};

int numbers[7]={0};

int size;

while (!file.eof()){

string line;

getline(file,line);

size=change(numbers,line);

int s=0;

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

s+=numbers[i];

}

cout<<"Total hours = "<<s<<endl;

for(int i=0;i<7;i++){

total[i]+=numbers[i];

}

}

cout<<endl;

string weekDays[]={"Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"};

int sum=0;

for(int i=0;i<7;i++){

cout<<weekDays[i]<<" hours = "<<total[i]<<endl;

sum+=total[i];

}

cout<<"Total hours = "<<sum<<endl;

}

int change(int numbers[], const string &line){

int i = 0,j,val;

stringstream str(line);

while (str>>val){

numbers[i]=val;

i++;

}

for(j=i;j<7;j++){

numbers[j]=0;

}

return i;

}

|  |  |
| --- | --- |
| 题目得分 |  |

**参考答案：**

#include <iostream>

#include <fstream>

#include <string>

#include <sstream>

#include <cstdlib>

using namespace std;

const int DAYS = 7;

void process(ifstream &inFile);

// 将字符串line分离为数字存放在数组numbers中，并返回numbers中实际元素个数

int transferFrom(int numbers[], const string &line);

// 返回数组numbers的和（某员工每周工作时间的小时数）

int sum(int numbers[], int size);

// 统计员工每周每天工作累计小时数

void addTo(int total[], int numbers[]);

// 输出员工每周每天工作累计小时数和总的小时数

void print(int total[]);

int main() {

ifstream inFile;

inFile.open("hours.txt");

if(!inFile.is\_open())

exit(EXIT\_FAILURE);

process(inFile);

return 0;

}

void process(ifstream &inFile) {

int total[DAYS] = {0}; // 员工每周每天工作累计小时数

int numbers[DAYS] = {0}; // 某员工每周工作时间

int size;

while (!inFile.eof()) {

string line;

// 从文件中读取一行字符串

getline(inFile, line);

// 将字符串分离为数字

size = transferFrom(numbers, line);

cout << "Total hours = " << sum(numbers, size) << endl;

addTo(total, numbers);

}

cout << endl;

print(total);

}

int transferFrom(int numbers[], const string &line) {

int i = 0, j, value;

stringstream myStream(line);

while(myStream >> value) {

numbers[i] = value;

++i;

}

// 每行数据个数不同，Number数组后面没有数据的元素值设置为0

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

numbers[j] = 0;

return i;

}

int sum(int numbers[], int size) {

int s = 0;

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

s += numbers[i];

return s;

}

void addTo(int total[], int numbers[]) {

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

total[i] += numbers[i];

}

void print(int total[]) {

string dayNames[] = {"Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"};

int s = 0;

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

cout << dayNames[i] << " hours = " << total[i] << endl;

s += total[i];

}

cout << "Total hours = " << s << endl;

}

1. 【描述】  
   将一个明文文件plaintext.txt中的内容，按照一定的方法，对每个字符加密后存放到另一个密文文件ciphertext.txt中。  
   【输入】  
   文件plaintext.txt。（该文件已经存在，无需自己创建）  
   注意：本地调试程序时，则要求自己预先建立好plaintext.txt文件。在Windows下，可以使用记事本。  
   【输出】  
   生成文件ciphertext.txt，里面存放加密后的信息。  
   不需要在屏幕上显示信息。  
   【输入示例】  
   文件plaintext.txt，假定其中内容如下：  
   Welcome to C++!  
   【输出示例】  
   生成文件ciphertext.txt，加密后的内容如下：  
   Ygneqog"vq"E--#  
   【提示】  
   这里采用一种简单的加密方法，将每个字符的编码加2。  
   【来源】  
   《程序设计基础——以C++为例》第8章实验2。（10分）

**我的答案：**

#include <iostream>

#include <fstream>

#include <string>

#include <cstdlib>

using namespace std;

int main() {

const int key = 2;

ifstream inFile;

inFile.open("plaintext.txt");

ofstream outFile;

outFile.open("ciphertext.txt");

char c;

inFile.get(c);

while(!inFile.eof()) {

c += key;

outFile.put(c);

inFile.get(c);

}

inFile.close();

outFile.close();

return 0;

}

|  |  |
| --- | --- |
| 题目得分 |  |

**参考答案：**

#include <iostream>

#include <fstream>

#include <string>

#include <cstdlib>

using namespace std;

int main() {

const int CODING = 2;

ifstream inFile;

inFile.open("plaintext.txt");

if(!inFile.is\_open()) {

exit(EXIT\_FAILURE);

}

ofstream outFile;

outFile.open("ciphertext.txt");

if(!outFile.is\_open()) {

exit(EXIT\_FAILURE);

}

char ch;

inFile.get(ch);

while(!inFile.eof()) {

ch += CODING;

outFile.put(ch);

inFile.get(ch);

}

inFile.close();

outFile.close();

return 0;

}

1. **【**描述**】**处理日志文件，日志文件的储存格式为“年/月/日 时:分:秒 用户名 操作”。  
   日志文件有多条记录：  
   2015/4/218:00:33 37c3b6b58c6ac3 LOGIN  
   2015/4/218:15:35 11734e186f24fe4c LOGIN  
   2015/4/218:34:57 9f3cf331d19a9f LOGIN  
   2015/4/219:00:29 389bcca2159f5de7 LOGIN  
   2015/4/219:08:29 c3bde693fdb3c3d LOGIN  
   ……  
   可以下载日志文件：  
   [鼠标右键另存为](http://dodo.hznu.edu.cn/Upload/lab/3084b8184a6f9802/face/3706f94486f02c35/log.txt)【输入】  
   日志文件log.txt。（该文件已经存在，无需自己创建）  
   【输出】  
   生成文件times.txt，里面存放活跃用户的在线时长。  
   不需要在屏幕上显示信息。  
   【输入示例】  
   无  
   【输出示例】  
   文件times.txt，注意：这里只给出了其中部分内容。  
   37c3b6b58c6ac 31887052  
   11734e186f24fe4c 2088931  
   9f3cf331d19a9f 2147511  
   389bcca2159f5de 71398643  
   c3bde693fdb3c3d 1297166  
   ……  
   【提示】  
   活跃用户指的是在日志文件中有过操作的用户，记得把重复出现的用户去掉。  
   在线时长单位为秒。为简单起见，日志文件中日期都为同一年的。

（10分）

**我的答案：**

#include <iostream>

#include <fstream>

#include <string>

#include <cstdlib>

using namespace std;

const int N=1000;

struct Time{

int year,month,day,hour,minute,second;

};

int computeTime(Time ,Time);

int main() {

ifstream file("log.txt");

int userCount=0;

string idArray[N];

bool online[N];

Time lastOnline[N];

int seconds[N];

while (!file.eof()){

Time t;

string id,operation;

char tmp;

file>>t.year>>tmp>>t.month>>tmp>>t.day;

file>>t.hour>>tmp>>t.minute>>tmp>>t.second;

file>>id;

file>>operation;

int found=-1;

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

if(id==idArray[i]){

found=i;

break;

}

}

if(found==-1){

idArray[userCount]=id;

if(operation=="LOGIN"){

online[userCount]= true;

lastOnline[userCount]=t;

}else online[userCount]= false;

seconds[userCount]=0;

userCount++;

}else{

if(operation=="LOGIN"){

if(!online[found]){

online[found]= true;

lastOnline[found]=t;

}

}else{

if(online[found]){

online[found]= false;

seconds[found]+=computeTime(lastOnline[found],t);

}

}

}

}

file.close();

ofstream outFile("times.txt");

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

outFile<<idArray[i]<<' '<<seconds[i]<<endl;

}

outFile.close();

return 0;

}

int computeTime(Time t1, Time t2) {

int days[12] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

int day\_count = t2.day - t1.day;

for(int i = t1.month; i < t2.month; ++i)

day\_count += days[i - 1];

int result = day\_count \* 60 \* 60 \* 24;

result += (t2.hour - t1.hour) \* 60 \* 60;

result += (t2.minute - t1.minute) \* 60;

result += (t2.second - t1.second);

return result;

}

|  |  |
| --- | --- |
| 题目得分 |  |

**参考答案：**

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

/\* 时间类型的结构定义 \*/

struct Time\_t {

int year, month, day;

int hour, minute, second;

};

/\* 计算时长的函数声明 \*/

int timeDifference(Time\_t, Time\_t);

int main() {

ifstream fin("log.txt");

int user\_count = 0; /\* 用户数，初始值为0 \*/

string ids[600]; /\* 记录所有的编号 \*/

bool online[600]; /\* 在线状态 \*/

Time\_t last\_on[600]; /\* 上次登录时间 \*/

int secs[600]; /\* 累计在线秒数 \*/

while(!fin.eof()) {

Time\_t t;

string id, operation;

char tmp;

fin >> t.year >> tmp >> t.month >> tmp >> t.day;

fin >> t.hour >> tmp >> t.minute >> tmp >> t.second;

fin >> id;

fin >> operation;

/\* 线性查找 \*/

int found = -1;

for(int i = 0; i < user\_count; ++i)

if(id == ids[i]) {

found = i;

break;

}

/\* 如果没找到，说明是新用户，记录 \*/

if(found == -1) {

ids[user\_count] = id;

/\* 记录新用户的登陆状态 \*/

if(operation == "LOGIN") {

online[user\_count] = true;

last\_on[user\_count] = t;

}

else

online[user\_count] = false;

secs[user\_count] = 0;

++user\_count;

}

/\* 否则，修改登录状态，计算登录时长 \*/

else {

if(operation == "LOGIN") {

if(!online[found]) {

online[found] = true;

last\_on[found] = t;

}

}

else {

if(online[found]) {

online[found] = false;

/\* 调用函数，计算两个时刻之差 \*/

secs[found] += timeDifference(last\_on[found], t);

}

}

}

}

fin.close();

/\* 输出每个用户的在线时长 \*/

ofstream fout("times.txt");

for(int i = 0; i < user\_count; ++i)

fout << ids[i] << " " << secs[i] << endl;

fout.close();

return 0;

}

/\* 计算时间差，日期在同一年 \*/

int timeDifference(Time\_t t1, Time\_t t2) {

int days[12] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

int day\_count = t2.day - t1.day;

for(int i = t1.month; i < t2.month; ++i)

day\_count += days[i - 1];

int result = day\_count \* 60 \* 60 \* 24;

result += (t2.hour - t1.hour) \* 60 \* 60;

result += (t2.minute - t1.minute) \* 60;

result += (t2.second - t1.second);

return result;

}

1. 【描述】  
   声明并实现了一个Rectangle类，表示矩形。Rectangle类包括：  
   double类型的私有数据成员width和height，表示矩形的宽和高。  
   带默认参数的构造函数，将矩形的宽和高设置为给定的参数。宽和高的默认参数值为1。  
   更改器函数setWidth和setHeight，分别用于修改矩形的宽和高。  
   访问器函数getWidth和getHeight，分别用于访问矩形的宽和高。  
   成员函数computeArea，返回矩形的面积。  
   成员函数computePerimeter，返回矩形的周长。  
   创建5个Rectangle对象（每个Rectangle对象的宽和高分别为1、2、3、4、5）并将它们写入二进制文件object.dat中。修改第3个对象的宽为10、高为3.5，再把修改后的该对象写回文件原位置。  
   【输入】  
   没有输入。  
   【输出】  
   生成文件object.dat  
   不需要在屏幕上显示信息。  
   【来源】  
   《程序设计基础——以C++为例》第8章实验4。（10分）

**我的答案：**

#include <iostream>

#include <fstream>

#include <cstdlib>

using namespace std;

class Rectangle {

public:

Rectangle(double width = 1, double height = 1) {

this->width = width;

this->height = height;

}

void setWidth(double width) {

this->width = width;

}

void setHeight(double height) {

this->height = height;

}

double getWidth() const {

return width;

}

double getHeight() const {

return height;

}

double computeArea() const {

return width \* height;

}

double computePerimeter() const {

return 2 \* (width + height);

}

private:

double width;

double height;

};

int main() {

fstream ioFile;

ioFile.open("object.dat", ios::in | ios::out | ios::trunc | ios::binary);

if(!ioFile.is\_open()) {

exit(EXIT\_FAILURE);

}

Rectangle r1;

Rectangle r2(2, 2);

Rectangle r3(3, 3);

Rectangle r4(4, 4);

Rectangle r5(5, 5);

ioFile.write(reinterpret\_cast<char \*>(&r1), sizeof(Rectangle));

ioFile.write(reinterpret\_cast<char \*>(&r2), sizeof(Rectangle));

ioFile.write(reinterpret\_cast<char \*>(&r3), sizeof(Rectangle));

ioFile.write(reinterpret\_cast<char \*>(&r4), sizeof(Rectangle));

ioFile.write(reinterpret\_cast<char \*>(&r5), sizeof(Rectangle));

Rectangle test;

test.setWidth(10);

test.setHeight(3.5);

ioFile.seekp(2 \* sizeof(Rectangle), ios::beg);

ioFile.write(reinterpret\_cast<char \*>(&test), sizeof(Rectangle));

ioFile.close();

return 0;

}

|  |  |
| --- | --- |
| 题目得分 |  |

**参考答案：**

#include <iostream>

#include <fstream>

#include <cstdlib>

using namespace std;

class Rectangle {

public:

Rectangle(double width = 1, double height = 1) {

this->width = width;

this->height = height;

}

void setWidth(double width) {

this->width = width;

}

void setHeight(double height) {

this->height = height;

}

double getWidth() const {

return width;

}

double getHeight() const {

return height;

}

double computeArea() const {

return width \* height;

}

double computePerimeter() const {

return 2 \* (width + height);

}

private:

double width;

double height;

};

int main() {

fstream ioFile;

ioFile.open("object.dat", ios::in | ios::out | ios::trunc | ios::binary);

if(!ioFile.is\_open()) {

exit(EXIT\_FAILURE);

}

Rectangle r1;

Rectangle r2(2, 2);

Rectangle r3(3, 3);

Rectangle r4(4, 4);

Rectangle r5(5, 5);

ioFile.write(reinterpret\_cast<char \*>(&r1), sizeof(Rectangle));

ioFile.write(reinterpret\_cast<char \*>(&r2), sizeof(Rectangle));

ioFile.write(reinterpret\_cast<char \*>(&r3), sizeof(Rectangle));

ioFile.write(reinterpret\_cast<char \*>(&r4), sizeof(Rectangle));

ioFile.write(reinterpret\_cast<char \*>(&r5), sizeof(Rectangle));

Rectangle ra;

ra.setWidth(10);

ra.setHeight(3.5);

ioFile.seekp(2 \* sizeof(Rectangle), ios::beg);

ioFile.write(reinterpret\_cast<char \*>(&ra), sizeof(Rectangle));

ioFile.close();

return 0;

}

1. 【描述】  
   输入一篇英文文章，求出现次数前三名的字母，不区分大小写（即出现A也计入出现a的次数中），按照次数由大到小输出，如果出现次数一样，按字典顺序输出。其它字符不予考虑。  
   附带说明：英文中，每个字母出现的频率是不一样的，在破译英文密码时，尤其是凯撒密码，统计每个字母出现的频率有着重要意义。  
   可以下载英文文章：  
   [鼠标右键另存为](http://dodo.hznu.edu.cn/Upload/lab/3084b8184a6f9802/face/a1a1583a5b0522e0/news.txt)【输入】  
   英文文章news.txt。（文件有多行，该文件已经存在，无需自己创建）  
   【输出】  
   以如下格式：  
   字母:出现次数  
   显示出现次数前三名的字母和出现次数。  
   【输入示例】  
   英文文章news.txt。  
   【输出示例】  
   a:xxx  
   b:xx  
   c:xx  
   【提示】  
   输出示例只是格式说明，并非正确答案。

（10分）

**我的答案：**

#include <iostream>

#include <fstream>

#include <cstdlib>

#include "map"

using namespace std;

int main() {

int sum[26]={0};

ifstream file("news.txt");

string line;

while (getline(file,line)){

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

if(isalpha(line[i])){

if(isupper(line[i])) line[i]=tolower(line[i]);

sum[line[i]-'a']++;

}

}

}

map<int,char,greater<int>>m;

for(int i=0;i<26;i++){

m.insert(make\_pair(sum[i],static\_cast<char>('a'+i)));

}

map<int,char>::iterator p=m.begin();

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

cout<<p->second<<":"<<p->first<<endl;

p++;

}

return 0;

}

|  |  |
| --- | --- |
| 题目得分 |  |

**参考答案：**

#include <iostream>

#include <fstream>

#include <string>

#include <cctype>

#include <map>

#include <functional>

using namespace std;

int main() {

int counts[26] = {0};

ifstream fin("news.txt");

string line;

while(getline(fin, line)) {

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

if(isalpha(line[i])) {

if(isupper(line[i]))

line[i] = tolower(line[i]);

++counts[line[i] - 'a'];

}

}

}

map<int, char, greater<int> > m;

for(int i = 0; i < 26; ++i)

m.insert(make\_pair(counts[i], static\_cast<char>('a' + i)));

map<int, char>::iterator p = m.begin();

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

cout << p->second << ":" << p->first << endl;

++p;

}

return 0;

}

1. **【**描述**】**处理日志文件，日志文件的储存格式为“年/月/日 时:分:秒 用户名 操作”。  
   日志文件有多条记录：  
   2015/4/218:00:33 37c3b6b58c6ac3 LOGIN  
   2015/4/218:15:35 11734e186f24fe4c LOGIN  
   2015/4/218:34:57 9f3cf331d19a9f LOGIN  
   2015/4/219:00:29 389bcca2159f5de7 LOGIN  
   2015/4/219:08:29 c3bde693fdb3c3d LOGIN  
   ……  
   可以下载日志文件：  
   [鼠标右键另存为](http://dodo.hznu.edu.cn/Upload/lab/3084b8184a6f9802/face/6946dcda8d1cdb3/log.txt)【输入】  
   日志文件log.txt。（该文件已经存在，无需自己创建）  
   【输出】  
   日志文件中活跃用户的数量。  
   【输入示例】  
   无  
   【输出示例】  
   123  
   【提示】  
   活跃用户指的是在日志文件中有过操作的用户，记得把重复出现的用户去掉。  
   输出示例只是格式说明，并非正确答案。

（10分）

**我的答案：**

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

int main() {

ifstream fin("log.txt");

int user\_count = 0;

string idArray[600];

while (!fin.eof()) {

int year, month, day, hour, minute, second;

char tmp;

string id, operation;

fin >> year >> tmp >> month >> tmp >> day;

fin >> hour >> tmp >> minute >> tmp >> second;

fin >> id;

fin >> operation;

int found = -1;

for(int i = 0; i < user\_count; i++) {

if (id == idArray[i]) {

found = i;

break;

}

}

if (found == -1)

idArray[user\_count++] = id;

}

fin.close();

cout << user\_count << endl;

return 0;

}

|  |  |
| --- | --- |
| 题目得分 |  |

**参考答案：**

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

int main() {

ifstream fin("log.txt");

int user\_count = 0;

string ids[600];

while (!fin.eof()) {

int year, month, day, hour, minute, second;

char tmp;

string id, operation;

fin >> year >> tmp >> month >> tmp >> day;

fin >> hour >> tmp >> minute >> tmp >> second;

fin >> id;

fin >> operation;

int found = -1;

for(int i = 0; i < user\_count; i++) {

if (id == ids[i]) {

found = i;

break;

}

}

if (found == -1)

ids[user\_count++] = id;

}

fin.close();

cout << user\_count << endl;

return 0;

}

**《C++程序设计实践》实验九**

**一、程序设计题（共10小题，100分）**

|  |  |
| --- | --- |
| 题型得分 | 100 |

1. 【描述】  
   ①声明并实现一个名为Vehicle的基类，表示汽车。Vehicle类包括：  
   int类型的私有数据成员numberOfDoors，表示车门数量。  
   int类型的私有数据成员numberOfCylinders，表示气缸数量。  
   string类型的私有数据成员color，表示汽车颜色。  
   double类型的私有数据成员fuelLevel，表示油位，即燃油数量。  
   int类型的私有数据成员transmissionType，表示变速箱类型，即0表示手动、1表示自动。  
   string类型的私有数据成员className，表示汽车类别。  
   有参构造函数，将车门数量、气缸数量、汽车颜色、燃油数量、变速箱类型设置为给定的参数。  
   访问器函数getNumberOfDoors、getNumberOfCylinders、getColor、getFuelLevel、getTransmissionType、getClassNme，分别用于访问车门数量、气缸数量、汽车颜色、燃油数量、变速箱类型、汽车类别。  
   更改器函数setColor、setFuelLevel、setClassNme，分别用于更改汽车颜色、燃油数量、汽车类别。  
   重载流插入运算符<<，输出相关信息。  
   ②从Vehicle类派生出Taxi类，表示出租车。Taxi类有数据成员customers（是否载客，bool类型）以及有参构造函数，将车门数量、气缸数量、汽车颜色、燃油数量、变速箱类型、是否载客设置为给定的参数；访问器/更改器函数hasCustomers和setCustomers；重载流插入运算符<<，输出相关信息。  
   ③从Vehicle类派生出Truck类，表示卡车。Truck类有数据成员cargo（是否载货，bool类型）以及有参构造函数，将车门数量、气缸数量、汽车颜色、燃油数量、变速箱类型、是否载货设置为给定的参数；访问器/更改器函数hasCargo和setCargo；重载流插入运算符<<，输出相关信息。  
   【输入】  
   输入车门数量、气缸数量，颜色、燃油数量、变速箱类型。  
   【输出】  
   见【输出示例】  
     
   【输入示例】  
   2 6 red 50 1  
   4 6 yellow 60 0  
   2 16 black 100 0  
     
   【输出示例】  
   Vehicle  
   Number of doors:2  
   Number of cylinders:6  
   Transmission type:Automatic  
   Color:red  
   Fuel level:50  
   Taxi  
   Number of doors:4  
   Number of cylinders:6  
   Transmission type:Manual  
   Color:yellow  
   Fuel level:60  
   Has no passengers  
   Truck  
   Number of doors:2  
   Number of cylinders:16  
   Transmission type:Manual  
   Color:black  
   Fuel level:100  
   Is carrying cargo

【来源】  
《程序设计基础——以C++为例》第7章实验3。（10分）

**我的答案：**

class Vehicle {

public:

Vehicle(int doors, int cylinders, string color, double fuel, int transmission);

int getNumberOfDoors() const;

int getNumberOfCylinders() const;

string getColor() const;

double getFuelLevel() const;

int getTransmissionType() const;

string getClassName() const;

void setColor(string color);

void setFuelLevel(double fuel);

void setClassName(string className);

private:

int numberOfDoors; // 车门数量

int numberOfCylinders;// 气缸数量

string color; // 汽车颜色

double fuelLevel; // 油位

int transmissionType; // 变速箱类型

string className; // 汽车类别

friend ostream &operator<<(ostream &out, const Vehicle &v);

};

Vehicle::Vehicle(int doors, int cylinders, string color, double fuel, int transmission) {

numberOfDoors = doors;

numberOfCylinders = cylinders;

this->color = color;

fuelLevel = fuel;

transmissionType = transmission;

className = "Vehicle";

}

int Vehicle::getNumberOfDoors() const {

return numberOfDoors;

}

int Vehicle::getNumberOfCylinders() const {

return numberOfCylinders;

}

string Vehicle::getColor() const {

return color;

}

double Vehicle::getFuelLevel() const {

return fuelLevel;

}

int Vehicle::getTransmissionType() const {

return transmissionType;

}

string Vehicle::getClassName() const {

return className;

}

void Vehicle::setColor(string color) {

this->color = color;

}

void Vehicle::setFuelLevel(double fuel) {

fuelLevel = fuel;

}

void Vehicle::setClassName(string className) {

this->className = className;

}

ostream &operator<<(ostream &out, const Vehicle &v) {

string msg;

out << v.className << endl

<< "Number of doors:" << v.numberOfDoors << endl

<< "Number of cylinders:" << v.numberOfCylinders << endl;

if(v.transmissionType == 0)

msg = "Manual";

else

msg = "Automatic";

out << "Transmission type:" << msg << endl

<< "Color:" << v.color << endl

<< "Fuel level:" << v.fuelLevel << endl;

return out;

}

class Taxi:public Vehicle {

public:

Taxi(int doors, int cylinders, string color, double fuel, int transmission, bool customers);

void setCustomers(bool customers);

bool hasCustomers() const;

private:

bool customers;

friend ostream &operator<<(ostream &out, const Taxi &t);

};

Taxi::Taxi(int doors, int cylinders, string color, double fuel, int transmission, bool customers)

:Vehicle(doors, cylinders, color, fuel, transmission) {

this->customers = customers;

setClassName("Taxi");

}

void Taxi::setCustomers(bool customers) {

this->customers = customers;

}

bool Taxi::hasCustomers() const {

return customers;

}

ostream &operator<<(ostream &out, const Taxi &t) {

string msg;

out << t.getClassName()<< endl

<< "Number of doors:" << t.getNumberOfDoors() << endl

<< "Number of cylinders:" << t.getNumberOfCylinders() << endl;

if(t.getTransmissionType() == 0)

msg = "Manual";

else

msg = "Automatic";

out << "Transmission type:" << msg << endl

<< "Color:" << t.getColor() << endl

<< "Fuel level:" << t.getFuelLevel() << endl;

if(t.hasCustomers())

msg = "Has passengers";

else

msg = "Has no passengers";

out << msg << endl;

return out;

}

class Truck:public Vehicle {

public:

Truck(int doors, int cylinders, string color, double fuel, int transmission, bool cargo);

void setCargo(bool cargo);

bool hasCargo() const;

private:

bool cargo;

friend ostream &operator<<(ostream &out, const Truck &t);

};

Truck::Truck(int doors, int cylinders, string color, double fuel, int transmission, bool cargo)

:Vehicle(doors, cylinders, color, fuel, transmission) {

this->cargo = cargo;

setClassName("Truck");

}

void Truck::setCargo(bool cargo) {

this->cargo = cargo;

}

bool Truck::hasCargo() const {

return cargo;

}

ostream &operator<<(ostream &out, const Truck &t) {

string msg;

out << t.getClassName()<< endl

<< "Number of doors:" << t.getNumberOfDoors() << endl

<< "Number of cylinders:" << t.getNumberOfCylinders() << endl;

if(t.getTransmissionType() == 0)

msg = "Manual";

else

msg = "Automatic";

out << "Transmission type:" << msg << endl

<< "Color:" << t.getColor() << endl

<< "Fuel level:" << t.getFuelLevel() << endl;

if(t.hasCargo())

msg = "Is carrying cargo";

else

msg = "Is not carrying cargo";

out << msg << endl;

return out;

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

**参考答案：**

#include <iostream>

#include <string>

using namespace std;

class Vehicle {

public:

Vehicle(int doors, int cylinders, string color, double fuel, int transmission);

int getNumberOfDoors() const;

int getNumberOfCylinders() const;

string getColor() const;

double getFuelLevel() const;

int getTransmissionType() const;

string getClassName() const;

void setColor(string color);

void setFuelLevel(double fuel);

void setClassName(string className);

private:

int numberOfDoors; // 车门数量

int numberOfCylinders;// 气缸数量

string color; // 汽车颜色

double fuelLevel; // 油位

int transmissionType; // 变速箱类型

string className; // 汽车类别

friend ostream &operator<<(ostream &out, const Vehicle &v);

};

Vehicle::Vehicle(int doors, int cylinders, string color, double fuel, int transmission) {

numberOfDoors = doors;

numberOfCylinders = cylinders;

this->color = color;

fuelLevel = fuel;

transmissionType = transmission;

className = "Vehicle";

}

int Vehicle::getNumberOfDoors() const {

return numberOfDoors;

}

int Vehicle::getNumberOfCylinders() const {

return numberOfCylinders;

}

string Vehicle::getColor() const {

return color;

}

double Vehicle::getFuelLevel() const {

return fuelLevel;

}

int Vehicle::getTransmissionType() const {

return transmissionType;

}

string Vehicle::getClassName() const {

return className;

}

void Vehicle::setColor(string color) {

this->color = color;

}

void Vehicle::setFuelLevel(double fuel) {

fuelLevel = fuel;

}

void Vehicle::setClassName(string className) {

this->className = className;

}

ostream &operator<<(ostream &out, const Vehicle &v) {

string msg;

out << v.className << endl

<< "Number of doors:" << v.numberOfDoors << endl

<< "Number of cylinders:" << v.numberOfCylinders << endl;

if(v.transmissionType == 0)

msg = "Manual";

else

msg = "Automatic";

out << "Transmission type:" << msg << endl

<< "Color:" << v.color << endl

<< "Fuel level:" << v.fuelLevel << endl;

return out;

}

class Taxi:public Vehicle {

public:

Taxi(int doors, int cylinders, string color, double fuel, int transmission, bool customers);

void setCustomers(bool customers);

bool hasCustomers() const;

private:

bool customers;

friend ostream &operator<<(ostream &out, const Taxi &t);

};

Taxi::Taxi(int doors, int cylinders, string color, double fuel, int transmission, bool customers)

:Vehicle(doors, cylinders, color, fuel, transmission) {

this->customers = customers;

setClassName("Taxi");

}

void Taxi::setCustomers(bool customers) {

this->customers = customers;

}

bool Taxi::hasCustomers() const {

return customers;

}

ostream &operator<<(ostream &out, const Taxi &t) {

string msg;

out << t.getClassName()<< endl

<< "Number of doors:" << t.getNumberOfDoors() << endl

<< "Number of cylinders:" << t.getNumberOfCylinders() << endl;

if(t.getTransmissionType() == 0)

msg = "Manual";

else

msg = "Automatic";

out << "Transmission type:" << msg << endl

<< "Color:" << t.getColor() << endl

<< "Fuel level:" << t.getFuelLevel() << endl;

if(t.hasCustomers())

msg = "Has passengers";

else

msg = "Has no passengers";

out << msg << endl;

return out;

}

class Truck:public Vehicle {

public:

Truck(int doors, int cylinders, string color, double fuel, int transmission, bool cargo);

void setCargo(bool cargo);

bool hasCargo() const;

private:

bool cargo;

friend ostream &operator<<(ostream &out, const Truck &t);

};

Truck::Truck(int doors, int cylinders, string color, double fuel, int transmission, bool cargo)

:Vehicle(doors, cylinders, color, fuel, transmission) {

this->cargo = cargo;

setClassName("Truck");

}

void Truck::setCargo(bool cargo) {

this->cargo = cargo;

}

bool Truck::hasCargo() const {

return cargo;

}

ostream &operator<<(ostream &out, const Truck &t) {

string msg;

out << t.getClassName()<< endl

<< "Number of doors:" << t.getNumberOfDoors() << endl

<< "Number of cylinders:" << t.getNumberOfCylinders() << endl;

if(t.getTransmissionType() == 0)

msg = "Manual";

else

msg = "Automatic";

out << "Transmission type:" << msg << endl

<< "Color:" << t.getColor() << endl

<< "Fuel level:" << t.getFuelLevel() << endl;

if(t.hasCargo())

msg = "Is carrying cargo";

else

msg = "Is not carrying cargo";

out << msg << endl;

return out;

}

int main() {

int doors, cylinders, transmission;

string color;

double fuel;

cin >> doors >> cylinders >> color >> fuel >> transmission;

Vehicle vehicle(doors, cylinders, color, fuel, transmission);

cout << vehicle;

cin >> doors >> cylinders >> color >> fuel >> transmission;

Taxi taxi(doors, cylinders, color, fuel, transmission, false);

cout << taxi;

cin >> doors >> cylinders >> color >> fuel >> transmission;

Truck truck(doors, cylinders, color, fuel, transmission, true);

cout << truck;

return 0;

}

1. 【描述】  
   ①Employee类是抽象类。Employee类的派生类有Boss类、CommissionWorker类、PieceWorker类和HourlyWorker类。  
   ②Employee类包括私有数据成员name（姓名，string类型）；有参构造函数，将name设置为给定的参数；访问器函数getName；还有纯虚函数show和earnings。show和earning函数将在每个派生类中实现，因为每个派生类显示的信息不同、计算工资的方法不同。  
   ③Boss类有固定的周工资且不计工作时间。Boss类包括私有数据成员weeklySalary（周工资，double类型）；有参构造函数，将name、weeklySalary设置为给定的参数；更改器函数setWeeklySalary；show函数和earnings函数。  
   ④CommissionWorker类有工资加上销售提成。CommissionWorker类包括私有数据成员salary（工资，double类型）、commission（佣金，double类型）和quantity（销售数量，int类型）；有参构造函数，将name、salary、commission、quantity设置为给定的参数；更改器函数setSalary、setCommission和setQuantity；show函数和earnings函数。  
   ⑤PieceWorker类的工资根据其生产的产品数量而定。PieceWorker类包括私有数据成员wagePerPiece（每件产品工资，double类型）、quantity（生产数量，int类型）；有参构造函数，将name、wagePerPiece、quantity设置为给定的参数；更改器函数setWage、setQuantity；show函数和earnings函数。  
   ⑥HourlyWorker类的工资根据小时计算并有加班工资。HourlyWorker类包括私有数据成员wage（小时工资，double类型）、hours（工作时数，double类型）；有参构造函数，将name、wage、hours设置为给定的参数；更改器函数setWage、setHours；show函数和earnings函数。  
   动态绑定show和earnings函数显示和计算各类人员的信息和工资。  
   【输入】  
   输入Boss的姓名和周工资。  
   输入CommissonWorker的姓名、工资、佣金和销售数量。  
   输入PieceWorker的姓名、每件产品工资和生产数量。  
   输入HourlyWorker的姓名、小时工资、工作时数。  
   【输出】  
   见【输出示例】  
   【输入示例】  
   ZhangSan 800.0  
   LiSi 400.0 3.0 150  
   WangWu 2.5 200  
   LiuLiu 13.75 40  
   【输出示例】  
   Boss: ZhangSan  
   Earned: $800  
   Commission Worker: LiSi  
   Earned: $850  
   Piece Worker: WangWu  
   Earned: $500  
   Hourly Worker: LiuLiu  
   Earned: $550  
   【来源】  
   《程序设计基础——以C++为例》第7章实验7。

（10分）

**我的答案：**

class Employee {

public:

Employee(string name) {

this->name = name;

}

string getName() {

return name;

}

virtual void show() = 0;

virtual double earnings() = 0;

private:

string name;

};

class Boss:public Employee {

public:

Boss(string name, double weeklySalary):Employee(name) {

this->weeklySalary = weeklySalary;

}

void setWeeklySalary(double weeklySalary) {

this->weeklySalary = weeklySalary;

}

void show() {

cout << "Boss: " << getName() << endl;

}

double earnings() {

return weeklySalary;

}

private:

double weeklySalary;

};

class CommissionWorker:public Employee {

public:

CommissionWorker(string name, double salary, double commission, int quantity):Employee(name) {

this->salary = salary;

this->commission = commission;

this->quantity = quantity;

}

void setSalary(double salary) {

this->salary = salary;

}

void setCommission(double commission) {

this->commission = commission;

}

void setQuantity(int quantity) {

this->quantity = quantity;

}

void show() {

cout << "Commission Worker: " << getName() << endl;

}

double earnings() {

return salary + commission \* quantity;

}

private:

double salary;

double commission;

int quantity;

};

class PieceWorker:public Employee {

public:

PieceWorker(string name, double wagePerPiece, int quantity):Employee(name) {

this->wagePerPiece = wagePerPiece;

this->quantity = quantity;

}

void setWagePerPiece(double wagePerPiece) {

this->wagePerPiece = wagePerPiece;

}

void setQuantity(int quantity) {

this->quantity = quantity;

}

void show() {

cout << "Piece Worker: " << getName() << endl;

}

double earnings() {

return wagePerPiece \* quantity;

}

private:

double wagePerPiece;

int quantity;

};

class HourlyWorker:public Employee {

public:

HourlyWorker(string name, double wage, double hours):Employee(name) {

this->wage = wage;

this->hours = hours;

}

void setWage(double wage) {

this->wage = wage;

}

void setHours(double hours) {

this->hours = hours;

}

void show() {

cout << "Hourly Worker: " << getName() << endl;

}

double earnings() {

return wage \* hours;

}

private:

double wage;

double hours;

};

|  |  |
| --- | --- |
| 题目得分 | 10 |

**参考答案：**

#include <iostream>

#include <string>

using namespace std;

class Employee {

public:

Employee(string name) {

this->name = name;

}

string getName() {

return name;

}

virtual void show() = 0;

virtual double earnings() = 0;

private:

string name;

};

class Boss:public Employee {

public:

Boss(string name, double weeklySalary):Employee(name) {

this->weeklySalary = weeklySalary;

}

void setWeeklySalary(double weeklySalary) {

this->weeklySalary = weeklySalary;

}

void show() {

cout << "Boss: " << getName() << endl;

}

double earnings() {

return weeklySalary;

}

private:

double weeklySalary; // 周工资

};

class CommissionWorker:public Employee {

public:

CommissionWorker(string name, double salary, double commission, int quantity):Employee(name) {

this->salary = salary;

this->commission = commission;

this->quantity = quantity;

}

void setSalary(double salary) {

this->salary = salary;

}

void setCommission(double commission) {

this->commission = commission;

}

void setQuantity(int quantity) {

this->quantity = quantity;

}

void show() {

cout << "Commission Worker: " << getName() << endl;

}

double earnings() {

return salary + commission \* quantity; // 工资+销售提成

}

private:

double salary; // 工资

double commission; // 佣金

int quantity; // 销售数量

};

class PieceWorker:public Employee {

public:

PieceWorker(string name, double wagePerPiece, int quantity):Employee(name) {

this->wagePerPiece = wagePerPiece;

this->quantity = quantity;

}

void setWagePerPiece(double wagePerPiece) {

this->wagePerPiece = wagePerPiece;

}

void setQuantity(int quantity) {

this->quantity = quantity;

}

void show() {

cout << "Piece Worker: " << getName() << endl;

}

double earnings() {

return wagePerPiece \* quantity;

}

private:

double wagePerPiece; // 每件产品工资

int quantity; // 生产数量

};

class HourlyWorker:public Employee {

public:

HourlyWorker(string name, double wage, double hours):Employee(name) {

this->wage = wage;

this->hours = hours;

}

void setWage(double wage) {

this->wage = wage;

}

void setHours(double hours) {

this->hours = hours;

}

void show() {

cout << "Hourly Worker: " << getName() << endl;

}

double earnings() {

return wage \* hours;

}

private:

double wage; // 小时工资

double hours; // 工作时数

};

int main() {

string name;

double weeklySalary, salary, commission, quantity, wagePerPiece, wage, hours;

cin >> name >> weeklySalary;

Boss b(name, weeklySalary);

cin >> name >> salary >> commission >> quantity;

CommissionWorker c(name, salary, commission, quantity);

cin >> name >> wagePerPiece >> quantity;

PieceWorker p(name, wagePerPiece, quantity);

cin >> name >> wage >> hours;

HourlyWorker h(name, wage, hours);

Employee \*ref; // 基类指针

ref = &b;

ref->show();

cout << "Earned: $" << ref->earnings() << endl;

ref = &c;

ref->show();

cout << "Earned: $" << ref->earnings() << endl;

ref = &p;

ref->show();

cout << "Earned: $" << ref->earnings() << endl;

ref = &h;

ref->show();

cout << "Earned: $" << ref->earnings() << endl;

return 0;

}

1. 【描述】  
   下面是不完整的类定义:  
   class A {  
   public:  
       virtual void print(){  
           cout << "print come form class A" << endl;  
       }  
   };  
   class B : public A {  
   private:  
       char \*buf;  
   public:  
       void print() {  
           cout << "print come from class B" << endl;  
       }     
   };  
   void fun(A \*a) {  
       delete a;  
   }  
   试完成其定义（需要增加必要的构造函数、析构函数）。

类A析构函数输出：A::~A() called

类B析构函数输出：B::~B() called

给定如下main函数：  
int main() {  
    A \*a = new B(10);  
    a->print();  
    fun(a);  
    B \*b = new B(20);  
    fun(b);  
    return 0;  
}  
【输入】  
没有输入。  
【输出】  
主函数的输出已经写好。（10分）

**我的答案：**

class A {

public:

virtual ~A() {

cout << "A::~A() called" << endl;

}

virtual void print(){

cout << "print come form class A" << endl;

}

};

class B : public A {

private:

char \*buf;

public:

B(int size) {

buf = new char[size];

}

~B() {

delete buf;

cout << "B::~B() called" << endl;

}

void print() {

cout << "print come from class B" << endl;

}

};

|  |  |
| --- | --- |
| 题目得分 | 10 |

**参考答案：**

#include <iostream>

using namespace std;

class A {

public:

virtual ~A() {

cout << "A::~A() called" << endl;

}

virtual void print(){

cout << "print come form class A" << endl;

}

};

class B : public A {

private:

char \*buf;

public:

B(int size) {

buf = new char[size];

}

~B() {

delete buf;

cout << "B::~B() called" << endl;

}

void print() {

cout << "print come from class B" << endl;

}

};

void fun(A \*a) {

delete a;

}

int main() {

A \*a = new B(10);//调用A的无参（默认）构造函数

a->print();

fun(a);

B \*b = new B(20);

fun(b);

return 0;

}

1. 【描述】  
   下面是类A的定义，需要补充或增加完整的无参构造函数、虚析构函数以及fun()虚函数。  
   class A {  
   public:  
      // ...  
      void g() {  
         fun();  
      }  
   };  
   构造函数输出：A constructor，并调用g()函数  
   析构函数输出：A destructor  
   fun()函数输出：Call class A's fun  
   下面是类B的定义，继承类A，需要补充或增加完整的无参构造函数、析构函数。  
   class B {  
   public:  
       // ...  
   };  
   无参构造函数输出：B constructor  
   析构函数输出：B destructor  
   下面是类C的定义，继承类B，需要补充或增加完整的无参构造函数、析构函数以及fun()函数。  
   class C {  
   public:  
       // ...  
   };  
   无参构造函数输出：C constructor  
   析构函数输出：C destructor  
   fun()函数输出：Call class C's fun  
   给定如下main函数：  
   int main() {   
       A \*a = new C;  
       a->g();  
       delete a;  
       return 0;  
   }  
   【输入】  
   没有输入。  
   【输出】  
   主函数的输出已经写好。  
   （10分）

**我的答案：**

class A {

public:

A() {

cout << "A constructor" << endl;

g();

}

virtual ~A() {

cout << "A destructor" << endl;

}

void g() {

fun();

}

virtual void fun() {

cout << "Call class A's fun" << endl;

}

};

class B : public A {

public:

B() {

cout << "B constructor" << endl;

}

~B() {

cout << "B destructor" << endl;

}

};

class C : public B {

public:

C() {

cout << "C constructor" << endl;

}

~C() {

cout << "C destructor" << endl;

}

void fun() {

cout << "Call class C's fun" << endl;

}

};

|  |  |
| --- | --- |
| 题目得分 | 10 |

**参考答案：**

#include <iostream>

using namespace std;

class A {

public:

A() {

cout << "A constructor" << endl;

g();

}

virtual ~A() {

cout << "A destructor" << endl;

}

void g() {

fun();

}

virtual void fun() {

cout << "Call class A's fun" << endl;

}

};

class B : public A {

public:

B() {

cout << "B constructor" << endl;

}

~B() {

cout << "B destructor" << endl;

}

};

class C : public B {

public:

C() {

cout << "C constructor" << endl;

}

~C() {

cout << "C destructor" << endl;

}

void fun() {

cout << "Call class C's fun" << endl;

}

};

int main() {

A \*a = new C;

a->g();

delete a;

return 0;

}

1. 【描述】  
   有5个类A、B、C、D、E。它们的形式均为：  
   class T {  
   public:  
       T() {  
           cout << "T()" << endl;  
       }  
       ~T() {  
           cout << "~T()" << endl;  
       }  
   };  
   这里T代表类名A、B、C、D、E。  
   给定如下main函数：  
   int main() {  
       E e;  
       return 0;  
   }  
   要求根据输出结果，声明并实现类A、B、C、D、E，确定类A、B、C、D、E之间的继承关系。  
   【输入】  
   没有输入。  
   【输出】  
   C()  
   B()  
   C()  
   A()  
   D()  
   E()  
   ~E()  
   ~D()  
   ~A()  
   ~C()  
   ~B()  
   ~C()

（10分）

**我的答案：**

class C {

public:

C() {

cout << "C()" << endl;

}

~C(){

cout << "~C()" << endl;

}

};

class A : public C {

public:

A() {

cout << "A()" << endl;

}

~A(){

cout << "~A()" << endl;

}

};

class B : public C {

public:

B() {

cout << "B()" << endl;

}

~B(){

cout << "~B()" << endl;

}

};

class D : public B, public A {

public:

D() {

cout << "D()" << endl;

}

~D(){

cout << "~D()" << endl;

}

};

class E : public D {

public:

E() {

cout << "E()" << endl;

}

~E() {

cout << "~E()" << endl;

}

};

|  |  |
| --- | --- |
| 题目得分 | 10 |

**参考答案：**

#include <iostream>

using namespace std;

class C {

public:

C() {

cout << "C()" << endl;

}

~C(){

cout << "~C()" << endl;

}

};

class A : public C {

public:

A() {

cout << "A()" << endl;

}

~A(){

cout << "~A()" << endl;

}

};

class B : public C {

public:

B() {

cout << "B()" << endl;

}

~B(){

cout << "~B()" << endl;

}

};

class D : public B, public A {

public:

D() {

cout << "D()" << endl;

}

~D(){

cout << "~D()" << endl;

}

};

class E : public D {

public:

E() {

cout << "E()" << endl;

}

~E() {

cout << "~E()" << endl;

}

};

int main() {

E e;

return 0;

}

1. 【描述】  
   将输入数据按特定要求原样输出。  
   【输入】  
   第一行是整数t，表明一共t组数据。  
   对每组数据：  
   第一行是整数n，表示下面一共有n行，0<n<100。  
   下面的每行，以一个字母开头，然后跟着一个整数，两者用空格分隔。字母只会是'A'或'B'。整数范围0到100。  
   【输出】  
   对每组输入数据，将其原样输出，每组数据的最后输出一行"\*\*\*\*"。  
   【输入示例】  
   2  
   4  
   A 3  
   B 4  
   A 5  
   B 6  
   3  
   A 4  
   A 3  
   A 2  
   【输出示例】  
   4  
   A 3  
   B 4  
   A 5  
   B 6  
   \*\*\*\*  
   3  
   A 4  
   A 3  
   A 2  
   \*\*\*\*（10分）

**我的答案：**

class A {

protected:

int x;

public:

virtual void Print() {

cout << "A " << x << endl;

}

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

};

class B:public A {

public:

virtual void Print() {

cout << "B " << x << endl;

}

B(int i):A(i) { }

};

void PrintInfo(A \*p) {

p->Print();

}

A \*a[100];

|  |  |
| --- | --- |
| 题目得分 | 10 |

**参考答案：**

#include <iostream>

using namespace std;

// 此处填写自己的代码

class A {

protected:

int x;

public:

virtual void Print() {

cout << "A " << x << endl;

}

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

};

class B:public A {

public:

virtual void Print() {

cout << "B " << x << endl;

}

B(int i):A(i) { }

};

void PrintInfo(A \*p) {

p->Print();

}

A \*a[100];

// 填写代码结束

int main() {

int t;

cin >> t;

while(t--) {

int n;

cin >> n;

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

char c;

int k;

cin >> c >> k;

if(c == 'A')

a[i] = new A(k);

else

a[i] = new B(k);

}

cout << n << endl;

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

PrintInfo(a[i]);

cout << "\*\*\*\*" << endl;

}

}

1. 【描述】  
   按要求计算数值。  
   【输入】  
   第一行是整数n，表示第二行有n个整数。  
   第二行：n个整数。  
   所有整数都在0和100之间。  
   【输出】  
   先输出：  
   1  
   100  
   101  
   101  
   对于输入中第二行的每个整数x，输出两行：  
   第一行：k=x；  
   第二行：x的平方。  
   【输入示例】  
   3  
   3 4 5  
   【输出示例】  
   1  
   100  
   101  
   101  
   k=3  
   9  
   k=4  
   16  
   k=5  
   25

（10分）

**我的答案：**

A(int i):n(i) { }

operator int() {

return n;

}

A &operator++() {

++n;

return \*this;

}

A operator++(int) {

A tmp(\*this);

++n;

return tmp;

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

**参考答案：**

#include <iostream>

#include <string>

using namespace std;

class A {

private:

int n;

public:

//此处填写自己的代码

A(int i):n(i) { }

operator int() {

return n;

}

A &operator++() {

++n;

return \*this;

}

A operator++(int) {

A tmp(\*this);

++n;

return tmp;

}

// 填写代码结束

};

class B {

private:

static int k;

static int Square(int n) {

cout << "k=" << k <<endl;

return n \* n;

}

friend int main();

};

int B::k;

int main() {

A a1(1), a2(2);

cout << a1++ << endl;

(++a1) = 100;

cout << a1 << endl;

cout << ++a1 << endl;

cout << a1 << endl;

int n;

cin >> n;

while(n --) {

cin >> B::k;

A a3(B::k);

cout << B::Square(a3) << endl;

}

return 0;

}

1. 【描述】  
   ①声明并实现一个名为Arc的类，在二维空间中以某一个点为中心，绘制圆弧。Arc类包含私有数据成员p（圆心，Point类型），radius（圆弧半径，double类型）；有参构造函数，将圆心、圆弧半径设置为给定的参数；成员函数draw，输出圆心和半径。  
   ②声明并实现一个名为Circle的类，在二维空间中以某一个点为中心，绘制圆。Circle类包含私有数据成员p（圆心，Point类型），radius（圆半径，double类型）；有参构造函数，将圆心、圆半径设置为给定的参数；成员函数draw，输出圆心和半径。  
   ③声明并实现一个名为Ellipse的类，在二维空间中以某一个点为中心，绘制椭圆。Ellipse类包含私有数据成员p（圆心，Point类型），xRadius、yRadius（椭圆轴，double类型）；有参构造函数，将圆心、椭圆轴设置为给定的参数；成员函数draw，输出圆心和轴。  
   ④声明并实现一个名为Rectangle的类，在二维空间中以某一个点为左上角，绘制矩形。Rectangle类包含私有数据成员p（左上角坐标，Point类型），width、height（矩形宽度和高度，double类型）；有参构造函数，将左上角坐标、矩形宽度和高度设置为给定的参数；成员函数draw，输出左上角坐标和宽度、高度。  
   ⑤声明并实现一个名为Mix的类，在二维空间中以某一个点为中心，绘制圆弧、圆、椭圆，以某一个点为左上角，绘制矩形。Mix类包含有参构造函数，将点坐标、圆弧半径、圆半径、椭圆轴、矩形宽度和高度设置为给定的参数；成员函数draw绘制圆弧、圆、椭圆和矩形，调用Arc类、Circle类、Ellipse类、Rectangle类的draw函数，输出相关信息。  
   Mix类是Arc类、Circle类、Ellipse类、Rectangle类的多继承派生类。  
   【输入】  
   没有输入。  
   【输出】  
   Drawing an arc: Center(320, 240), radius(100)  
   Drawing a circle: Center(320, 240), radius(70)  
   Drawing an ellipse: Center(320, 240), x-axis(100), y-axis(70)  
   Drawing a rectangle: Upper left corner coordinates(320, 240), width(100), height(70)  
   【提示】  
   需要自定义Point类。  
   给定如下main函数：  
   int main() {  
       Point p(320, 240);  
       Mix mix(p, 100, 70);  
       mix.draw();  
       return 0;  
   }  
   【来源】  
   《程序设计基础——以C++为例》第7章实验5。（10分）

**我的答案：**

class Point {

public:

Point(double x = 0, double y = 0) {

this->x = x;

this->y = y;

}

double getX() const {

return x;

}

double getY() const {

return y;

}

private:

double x;

double y;

};

class Arc {

public:

Arc(Point p, double radius):p(p), radius(radius) { }

void draw() {

cout << "Drawing an arc: Center(" << p.getX() << ", " << p.getY()

<< "), radius(" << radius << ")" << endl;

}

private:

Point p;

double radius;

};

class Circle {

public:

Circle(Point p, double radius):p(p), radius(radius) { }

void draw() {

cout << "Drawing a circle: Center(" << p.getX() << ", " << p.getY()

<< "), radius(" << radius << ")" << endl;

}

private:

Point p;

double radius;

};

class Ellipse {

public:

Ellipse(Point p, double xRadius, double yRadius):p(p), xRadius(xRadius), yRadius(yRadius) { }

void draw() {

cout << "Drawing an ellipse: Center(" << p.getX() << ", " << p.getY()

<< "), x-axis(" << xRadius << "), y-axis(" << yRadius << ")" << endl;

}

private:

Point p;

double xRadius;

double yRadius;

};

class Rectangle {

public:

Rectangle(Point p, double width, double height):p(p), width(width), height(height) { }

void draw() {

cout << "Drawing a rectangle: Upper left corner coordinates(" << p.getX() << ", " << p.getY()

<< "), Width(" << width << "), Height(" << height << ")" << endl;

}

private:

Point p;

double width;

double height;

};

class Mix:public Arc, public Circle, public Ellipse, public Rectangle {

public:

Mix(Point p, double radius1, double radius2)

:Arc(p, radius1), Circle(p, radius2),

Ellipse(p, radius1, radius2), Rectangle(p, radius1, radius2) { }

void draw() {

Arc::draw();

Circle::draw();

Ellipse::draw();

Rectangle::draw();

}

};

|  |  |
| --- | --- |
| 题目得分 | 10 |

**参考答案：**

#include <iostream>

using namespace std;

class Point {

public:

Point(double x = 0, double y = 0) {

this->x = x;

this->y = y;

}

double getX() const {

return x;

}

double getY() const {

return y;

}

private:

double x;

double y;

};

class Arc {

public:

Arc(Point p, double radius):p(p), radius(radius) { }

void draw() {

cout << "Drawing an arc: Center(" << p.getX() << ", " << p.getY()

<< "), radius(" << radius << ")" << endl;

}

private:

Point p;

double radius;

};

class Circle {

public:

Circle(Point p, double radius):p(p), radius(radius) { }

void draw() {

cout << "Drawing a circle: Center(" << p.getX() << ", " << p.getY()

<< "), radius(" << radius << ")" << endl;

}

private:

Point p;

double radius;

};

class Ellipse {

public:

Ellipse(Point p, double xRadius, double yRadius):p(p), xRadius(xRadius), yRadius(yRadius) { }

void draw() {

cout << "Drawing an ellipse: Center(" << p.getX() << ", " << p.getY()

<< "), x-axis(" << xRadius << "), y-axis(" << yRadius << ")" << endl;

}

private:

Point p;

double xRadius;

double yRadius;

};

class Rectangle {

public:

Rectangle(Point p, double width, double height):p(p), width(width), height(height) { }

void draw() {

cout << "Drawing a rectangle: Upper left corner coordinates(" << p.getX() << ", " << p.getY()

<< "), Width(" << width << "), Height(" << height << ")" << endl;

}

private:

Point p;

double width;

double height;

};

class Mix:public Arc, public Circle, public Ellipse, public Rectangle {

public:

Mix(Point p, double radius1, double radius2)

:Arc(p, radius1), Circle(p, radius2),

Ellipse(p, radius1, radius2), Rectangle(p, radius1, radius2) { }

void draw() {

Arc::draw();

Circle::draw();

Ellipse::draw();

Rectangle::draw();

}

};

int main() {

Point p(320, 240);

Mix mix(p, 100, 70);

mix.draw();

return 0;

}

1. 【描述】  
   ①声明并实现一个名为Person的基类，Person类有保护数据成员name（姓名，string类型）、sex（性别，char类型，'M'表示男，'F'表示女）。以及有参构造函数，将姓名、性别设置为给定的参数；成员函数print，输出姓名和性别。  
   ②从Person类派生出Student类，Student类有私有数据成员status（状态，枚举类型），表示年级（FRESHMAN、SOPHOMORE、JUNIOR、SENIOR），表示大一、大二、大三、大四学生。以及有参构造函数，将姓名、性别、状态设置为给定的参数；成员函数print，print函数输出姓名、性别和状态。  
   ③定义MyDate类，它包含私有数据成员year、month和day以及带默认参数的有参构造函数，年、月、日的默认参数值分别为1900、1、1；成员函数print，输出年、月、日。  
   ④从Person类派生出Employee类，Employee类有保护数据成员salary（薪水，int类型）、dateHired（雇佣日期），dataHired的类型是MyDate。以及有参构造函数，将姓名、性别、薪水和雇佣日期设置为给定的参数；成员函数print，输出姓名、性别、薪水和雇佣日期。  
   ⑤从Employee类派生出Faculty类，Faculty类有私有数据成员rank（级别，枚举类型），有（PROFESSOR、ASSOCIATE\_PROFESSOR、LECTURER），表示教授、副教授、讲师。以及有参构造函数，将姓名、性别、薪水、雇佣日期和级别设置为给定的参数；成员函数print，输出姓名、性别、薪水、雇佣日期和级别。  
   ⑥从Employee类派生出Staff类，Staff类有私有数据成员headship（职务，枚举类型），有（PRESIDENT、DEAN、DEPARTMENT\_CHAIRMAN），表示校长、院长、系主任。以及有参构造函数，将姓名、性别、薪水、雇佣日期和职务设置为给定的参数；成员函数print，输出姓名、性别、薪水、雇佣日期和职务。  
   【输入】  
   没有输入。  
   【输出】  
   Name:ZhangSan, Sex:M  
   Name:LiSi, Sex:F  
   Status:Freshman  
   Name:WangWu, Sex:M  
   Salary:5000, Hire date:2012-3-1  
   Name:LiuLiu, Sex:M  
   Salary:10000, Hire date:2012-3-1  
   Rank:Professor  
   Name:QianQi, Sex:M  
   Salary:8000, Hire date:2012-3-1  
   Headship:Department chairman  
   【来源】  
   《程序设计基础——以C++为例》第7章实验4。

（10分）

**我的答案：**

class Person {

public:

Person(string name, char sex) {

this->name = name;

this->sex = sex;

}

void print() {

cout << "Name:" << name << ", Sex:" << sex << endl;

}

protected:

string name;

char sex;

};

enum Status {FRESHMAN, SOPHOMORE, JUNIOR, SENIOR};

class Student:public Person {

public:

Student(string name, char sex, Status status):Person(name, sex) {

this->status = status;

}

void print() {

string msg;

Person::print();

switch(status) {

case FRESHMAN:

msg = "Freshman";

break;

case SOPHOMORE:

msg = "Sophomore";

break;

case JUNIOR:

msg = "Junior";

break;

case SENIOR:

msg = "Senior";

break;

}

cout << "Status:" << msg << endl;

}

private:

Status status;

};

class MyDate {

public:

MyDate(int year = 1900, int month = 1, int day = 1) {

this->year = year;

this->month = month;

this->day = day;

}

void print() {

cout << year << "-" << month << "-" << day << endl;

}

private:

int year;

int month;

int day;

};

class Employee:public Person {

public:

Employee(string name, char sex, int salary, MyDate &dateHired)

:Person(name, sex) {

this->salary = salary;

this->dateHired = dateHired;

}

void print() {

Person::print();

cout << "Salary:" << salary << ", Hire date:";

dateHired.print();

}

protected:

int salary;

MyDate dateHired;

};

enum Rank {PROFESSOR, ASSOCIATE\_PROFESSOR, LECTURER};

class Faculty:public Employee {

public:

Faculty(string name, char sex, int salary, MyDate &dateHired, Rank rank)

:Employee(name, sex, salary, dateHired) {

this->rank = rank;

}

void print() {

string msg;

Employee::print();

switch(rank) {

case PROFESSOR:

msg = "Professor";

break;

case ASSOCIATE\_PROFESSOR:

msg = "Associate professor";

break;

case LECTURER:

msg = "Lecturer";

break;

}

cout << "Rank:" << msg << endl;

}

private:

Rank rank;

};

enum Headship {PRESIDENT, DEAN, DEPARTMENT\_CHAIRMAN};

class Staff:public Employee {

public:

Staff(string name, char sex, int salary, MyDate &dateHired, Headship headship)

:Employee(name, sex, salary, dateHired) {

this->headship = headship;

}

void print() {

string msg;

Employee::print();

switch(headship) {

case PRESIDENT:

msg = "President";

break;

case DEAN:

msg = "Dean";

break;

case DEPARTMENT\_CHAIRMAN:

msg = "Department chairman";

break;

}

cout << "Headship:" << msg << endl;

}

private:

Headship headship;

};

|  |  |
| --- | --- |
| 题目得分 | 10 |

**参考答案：**

#include <iostream>

#include <string>

using namespace std;

class Person {

public:

Person(string name, char sex) {

this->name = name;

this->sex = sex;

}

void print() {

cout << "Name:" << name << ", Sex:" << sex << endl;

}

protected:

string name;

char sex;

};

enum Status {FRESHMAN, SOPHOMORE, JUNIOR, SENIOR};

class Student:public Person {

public:

Student(string name, char sex, Status status):Person(name, sex) {

this->status = status;

}

void print() {

string msg;

Person::print();

switch(status) {

case FRESHMAN:

msg = "Freshman";

break;

case SOPHOMORE:

msg = "Sophomore";

break;

case JUNIOR:

msg = "Junior";

break;

case SENIOR:

msg = "Senior";

break;

}

cout << "Status:" << msg << endl;

}

private:

Status status;

};

class MyDate {

public:

MyDate(int year = 1900, int month = 1, int day = 1) {

this->year = year;

this->month = month;

this->day = day;

}

void print() {

cout << year << "-" << month << "-" << day << endl;

}

private:

int year;

int month;

int day;

};

class Employee:public Person {

public:

Employee(string name, char sex, int salary, MyDate &dateHired)

:Person(name, sex) {

this->salary = salary;

this->dateHired = dateHired;

}

void print() {

Person::print();

cout << "Salary:" << salary << ", Hire date:";

dateHired.print();

}

protected:

int salary;

MyDate dateHired;

};

enum Rank {PROFESSOR, ASSOCIATE\_PROFESSOR, LECTURER};

class Faculty:public Employee {

public:

Faculty(string name, char sex, int salary, MyDate &dateHired, Rank rank)

:Employee(name, sex, salary, dateHired) {

this->rank = rank;

}

void print() {

string msg;

Employee::print();

switch(rank) {

case PROFESSOR:

msg = "Professor";

break;

case ASSOCIATE\_PROFESSOR:

msg = "Associate professor";

break;

case LECTURER:

msg = "Lecturer";

break;

}

cout << "Rank:" << msg << endl;

}

private:

Rank rank;

};

enum Headship {PRESIDENT, DEAN, DEPARTMENT\_CHAIRMAN};

class Staff:public Employee {

public:

Staff(string name, char sex, int salary, MyDate &dateHired, Headship headship)

:Employee(name, sex, salary, dateHired) {

this->headship = headship;

}

void print() {

string msg;

Employee::print();

switch(headship) {

case PRESIDENT:

msg = "President";

break;

case DEAN:

msg = "Dean";

break;

case DEPARTMENT\_CHAIRMAN:

msg = "Department chairman";

break;

}

cout << "Headship:" << msg << endl;

}

private:

Headship headship;

};

int main() {

Person person("ZhangSan", 'M');

Student student("LiSi", 'F', FRESHMAN);

MyDate date(2012, 3, 1);

Employee employee("WangWu", 'M', 5000, date);

Faculty faculty("LiuLiu", 'M', 10000, date, PROFESSOR);

Staff staff("QianQi", 'M', 8000, date, DEPARTMENT\_CHAIRMAN);

person.print();

student.print();

employee.print();

faculty.print();

staff.print();

return 0;

}

1. 【描述】  
   ①Shape类是抽象类，包括了纯虚函数getArea（求面积）、getPerimeter（求周长）、show（输出对象信息）以及成员函数getClassName（返回类名“Shape”）。  
   ②Circle类继承了Shape类，包括了double类型的私有数据成员radius，表示圆半径；带默认参数的有参构造函数，radius的默认参数值为1；访问器函数getRadius和更改器函数setRadius；重定义了getArea（求圆面积）、getPerimeter（求圆周长）、show（输出半径）、getClassName（返回类名“Circle”）函数。  
   ③Rectangle继承了Shape类，包括了double类型的私有数据成员width、height，分别表示矩形的宽度和高度；带默认参数的有参构造函数，width和height的默认参数值分别为1、1；访问器函数getWidth、getHeight和更改器函数setWidth、setHeight；重定义了getArea（求矩形面积）、getPerimeter（求矩形周长）、show（输出宽度和高度）、getClassName（返回类名“Rectangle”）函数。  
   ④Triangle类继承了Shape类，包括了double类型的私有数据成员side1、side2和side3，表示三角形三条边；有参构造函数，将三角形三条边设置为给定的参数；重定义了getArea（求三角形面积）、getPerimeter（求三角形周长）、show（输出三条边）、getClassName（返回类名“Triangle”）函数。  
   假设PI为3.14159。  
   【输入】  
   输入圆的半径、矩形的宽度和高度以及三角形的三条边。  
   【输出】  
   见【输出示例】  
   【输入示例】  
   3.5  
   5.8 11.8  
   1 1.5 1  
   【输出示例】
2. Circle:
3. Radius:3.5
4. Area:38.4845, Perimeter:21.9911
5. Rectangle:
6. Width:5.8, Height:11.8
7. Area:68.44, Perimeter:35.2
8. Triangle:
9. Side:1, 1.5, 1
10. Area:0.496078, Perimeter:3.5

【来源】  
《程序设计基础——以C++为例》第7章实验6。

（10分）

**我的答案：**

class Shape {

public:

virtual double getArea() = 0;

virtual double getPerimeter() = 0;

virtual void show() = 0;

string getClassName() {

return "Shape";

};

};

class Circle : public Shape {

public:

Circle(double radius = 1) {

this->radius = radius;

}

double getRadius() {

return radius;

}

void setRadius(double radius) {

this->radius = radius;

}

double getArea() {

return PI \* radius \* radius;

}

double getPerimeter() {

return 2 \* PI \* radius;

}

void show() {

cout << "Radius:" << radius << endl;

}

string getClassName() {

return "Circle";

}

private:

double radius;

};

class Rectangle : public Shape {

public:

Rectangle(double width = 1, double height = 1) {

this->width = width;

this->height = height;

}

double getWidth() {

return width;

}

double getHeight() {

return height;

}

void setWidth(double width) {

this->width = width;

}

void setHeight(double height) {

this->height = height;

}

double getArea() {

return width \* height;

}

double getPerimeter() {

return 2 \* (width + height);

}

void show() {

cout << "Width:" << width << ", Height:" << height << endl;

}

string getClassName() {

return "Rectangle";

}

private:

double width, height;

};

class Triangle : public Shape {

public:

Triangle(double side1, double side2, double side3) {

this->side1 = side1;

this->side2 = side2;

this->side3 = side3;

}

double getArea() {

double s = getPerimeter() / 2;

return sqrt(s \* (s - side1) \* (s - side2) \* (s - side3));

}

double getPerimeter() {

return side1 + side2 + side3;

}

void show() {

cout << "Side:" << side1 << ", " << side2 << ", " << side3 << endl;

}

string getClassName() {

return "Triangle";

}

private:

double side1, side2, side3;

};

|  |  |
| --- | --- |
| 题目得分 | 10 |

**参考答案：**

#include <iostream>

#include <string>

#include <cmath>

using namespace std;

const double PI = 3.14159;

class Shape {

public:

virtual double getArea() = 0;

virtual double getPerimeter() = 0;

virtual void show() = 0;

string getClassName() {

return "Shape";

};

};

class Circle : public Shape {

public:

Circle(double radius = 1) {

this->radius = radius;

}

double getRadius() {

return radius;

}

void setRadius(double radius) {

this->radius = radius;

}

double getArea() {

return PI \* radius \* radius;

}

double getPerimeter() {

return 2 \* PI \* radius;

}

void show() {

cout << "Radius:" << radius << endl;

}

string getClassName() {

return "Circle";

}

private:

double radius;

};

class Rectangle : public Shape {

public:

Rectangle(double width = 1, double height = 1) {

this->width = width;

this->height = height;

}

double getWidth() {

return width;

}

double getHeight() {

return height;

}

void setWidth(double width) {

this->width = width;

}

void setHeight(double height) {

this->height = height;

}

double getArea() {

return width \* height;

}

double getPerimeter() {

return 2 \* (width + height);

}

void show() {

cout << "Width:" << width << ", Height:" << height << endl;

}

string getClassName() {

return "Rectangle";

}

private:

double width, height;

};

class Triangle : public Shape {

public:

Triangle(double side1, double side2, double side3) {

this->side1 = side1;

this->side2 = side2;

this->side3 = side3;

}

double getArea() {

double s = getPerimeter() / 2;

return sqrt(s \* (s - side1) \* (s - side2) \* (s - side3));

}

double getPerimeter() {

return side1 + side2 + side3;

}

void show() {

cout << "Side:" << side1 << ", " << side2 << ", " << side3 << endl;

}

string getClassName() {

return "Triangle";

}

private:

double side1, side2, side3;

};

int main() {

double radius, width, height, side1, side2, side3;

cin >> radius;

Circle circle(radius);

cin >> width >> height;

Rectangle rectangle(width, height);

cin >> side1 >> side2 >> side3;

Triangle triangle(side1, side2, side3);

cout << circle.getClassName() << ":" << endl;

circle.show();

cout << "Area:" << circle.getArea();

cout << ", Perimeter:" << circle.getPerimeter() << endl;

cout << rectangle.getClassName() << ":" << endl;

rectangle.show();

cout << "Area:" << rectangle.getArea();

cout << ", Perimeter:" << rectangle.getPerimeter() << endl;

cout << triangle.getClassName() << ":" << endl;

triangle.show();

cout << "Area:" << triangle.getArea();

cout << ", Perimeter:" << triangle.getPerimeter() << endl;

return 0;

}

程序设计题，第09套，第01题 总分 10 得分 10

详细：

正确 用例1运行：通过

10 10

程序设计题，第09套，第02题 总分 10 得分 10

详细：

正确 用例1运行：通过

10 10

程序设计题，第09套，第03题 总分 10 得分 10

详细：

正确 用例1运行：通过

10 10

程序设计题，第09套，第04题 总分 10 得分 10

详细：

正确 用例1运行：通过

10 10

程序设计题，第09套，第05题 总分 10 得分 10

详细：

正确 用例1运行：通过

用例2运行：通过

10 10

程序设计题，第09套，第06题 总分 10 得分 10

详细：

正确 用例1运行：通过

用例2运行：通过

10 10

程序设计题，第09套，第07题 总分 10 得分 10

详细：

正确 用例1运行：通过

10 10

程序设计题，第09套，第08题 总分 10 得分 10

详细：

正确 用例1运行：通过

10 10

程序设计题，第09套，第09题 总分 10 得分 10

详细：

正确 用例1运行：通过

10 10

程序设计题，第09套，第10题 总分 10 得分 10

详细：

正确 用例1运行：通过

用例2运行：通过

10 10

**《C++程序设计实践》实验8**

**一、程序设计题（共10小题，100分）**

|  |  |
| --- | --- |
| 题型得分 | 100 |

1. 【描述】  
   声明并实现一个类模板Pair，表示数对。类模板Pair包括：  
   私有数据成员first和second，first的类型为类参数T1，second的类型为类参数T2。  
   有参构造函数，将first和second设置为给定的参数。  
   访问器函数，分别返回first和second的值。  
   成员函数print，输出一个数对。、  
   成员函数swap，交换两个数对的值。  
   重载关系运算符<、<=、>、>=、==、!=，判断两个数对之间的关系。如果满足关系，返回true，否则返回false。  
   【输入】  
   输入二行。第一行和第二行都是一个整数、一个浮点数，表示一个数对，整数和浮点数之间以空格间隔。  
   【输出】  
   见【输出示例】  
   【输入示例】  
   1 2.5  
   3 4.5  
   【输出示例】  
   p1:[1,2.5]  
   p2:[3,4.5]  
   p1:[3,4.5]  
   p2:[1,2.5]  
   p1 > p2  
   【提示】  
   约定数对p1小于p2，等价于p1.first小于p2.first或者p1.first大于等于p2.first且p1.second小于p2.second。  
   【来源】  
   《程序设计基础——以C++为例》第6章实验7。

（10分）

**我的答案：**

template<typename T1, typename T2>

class Pair {

private:

T1 first;

T2 second;

public:

Pair(Pair<T1, T2>& pair) {

this->first = pair.getFirst();

this->second = pair.getSecond();

}

Pair(T1 first, T2 second) {

this->first = first;

this->second = second;

}

void setFirst(T1 first) {

this->first = first;

}

T1 getFirst() {

return this->first;

}

void setSecond(T2 second) {

this->second = second;

}

T2 getSecond() {

return this->second;

}

void print() const{

cout << "[" << this->first << "," << this->second << "]" << endl;

}

void swap(Pair<T1, T2>& pair) {

T1 firstTemp = this->first;

T2 secondTemp = this->second;

this->first = pair.getFirst();

this->second = pair.getSecond();

pair.setFirst(firstTemp);

pair.setSecond(secondTemp);

}

bool operator<(Pair<T1, T2>& pair) {

return this->first < pair.getFirst() || this->second < pair.getSecond();

}

bool operator==(Pair<T1, T2>& pair) {

return this->first == pair.getFirst() && this->second == pair.getSecond();

}

};

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   以成员函数的形式重载关系运算符（<、<=、==、!=、>、>=），使用半径比较Circle对象大小。  
   【输入】  
   输入一行，两个圆半径，圆半径以空格间隔。  
   【输出】  
   分行输出圆半径对应的圆对象的比较结果。  
   【输入示例】  
   5 6  
   【输出示例】  
   true  
   true  
   false  
   true  
   false  
   false  
   【提示】  
   Circle类可以参阅《程序设计基础——以C++为例》第5章【例5.1】。  
   【来源】  
   《程序设计基础——以C++为例》第6章实验3。  
   （10分）

**我的答案：**

class Circle {

private:

double radius;

public:

Circle(double radius) {

this->radius = radius;

}

bool operator<(const Circle& circle) {

return this->radius < circle.radius;

}

bool operator<=(const Circle& circle) {

return this->radius <= circle.radius;

}

bool operator==(const Circle& circle) {

return this->radius == circle.radius;

}

bool operator!=(const Circle& circle) {

return this->radius != circle.radius;

}

bool operator>(const Circle& circle) {

return this->radius > circle.radius;

}

bool operator>=(const Circle& circle) {

return this->radius >= circle.radius;

}

};

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   声明并实现一个带下标越界检查的TwoArray类，表示二维数组（矩阵），存放整数。TwoArray类包括：  
   int类型的指针pArray，指向存放数组元素的动态分配内存空间。  
   int类型的私有数据成员rowSize，存放数组行长度。  
   int类型的私有数据成员columnSize，存放数组列长度。  
   有参构造函数，将数组行、列长度设置为给定的参数，并动态分配对应的内存空间。  
   拷贝构造函数。  
   析构函数。  
   重载赋值运算符。  
   重载函数调用运算符。  
   两个访问器函数getRowSize和getColumnSize，用于获取数组行、列长度。  
   函数调用运算符()必须以成员函数形式进行重载。假设matrix是TwoArray类的对象，重载函数调用运算符的意图是使用matrix(i, j)这样的形式来访问二维数组元素，而不是原来的matrix[i][j]的形式。  
   重载函数调用运算符的语法如下：  
      函数类型operator()(参数表);  
   和重载下标运算符一样，在twoArray类中，重载函数调用运算符有两种形式：  
          int &operator()(int row, int column);  
   以及：  
          int operator()(int row, int column)const;  
   定义普通函数：void printTwoArray(const TwoArray &array)  
   按行输出二维数组，每个元素宽度为3。  
   【输入】  
   输入一个3×4矩阵和一个4×4矩阵。  
   【输出】  
   见【输出示例】  
   【输入示例】  
   0 1 2 3  
   2 3 4 5  
   4 5 6 7  
   0 1 2 3  
   3 4 5 6  
   6 7 8 9  
   9 10 11 12  
   【输出示例】  
   array1:  
     0  1  2  3  
     2  3  4  5  
     4  5  6  7  
   array2:  
     0  1  2  3  
     3  4  5  6  
     6  7  8  9  
     9 10 11 12  
   array3:  
     0  1  2  3  
     2  3  4  5  
     4  5  6  7  
   Row index out of range!  
   array1:  
     0  1  2  3  
     3  4  5  6  
     6  7  8  9  
     9 10 11 12  
   【来源】  
   《程序设计基础——以C++为例》第6章实验4。

（10分）

**我的答案：**

class TwoArray {

private:

int rowSize;

int columnSize;

int (\*pArray)[4];

public:

TwoArray(int rowSize, int columnSize) {

this->rowSize = rowSize;

this->columnSize = columnSize;

pArray = new int[rowSize][4];

}

TwoArray(TwoArray& array) {

\*this = array;

}

~TwoArray() {

delete []pArray;

}

int getRowSize() {

return this->rowSize;

}

int getColumnSize() {

return this->columnSize;

}

int &operator()(int row, int column) const{

if (row >= this->rowSize) {

throw out\_of\_range("Row index out of range!");

} else if (column >= this->columnSize) {

throw out\_of\_range("Column index out of range!");

}

return this->pArray[row][column];

}

};

void printTwoArray(TwoArray& array) {

for (int i = 0; i < array.getRowSize(); i++) {

for (int j = 0; j < array.getColumnSize(); j++) {

cout << setw(3) << array(i, j);

}

cout << endl;

}

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   我们都知道，对于一个表达式可以这么写：1+2-3+4  
   那我们是否也可以这样想，在对象的层次上能不能也有这样类似的级联（链式）调用来完成这样的功能呢？  
   例如上边的表达式可以变成如下格式：  
   Number op(1);  
   op.add(2).sub(3).add(4)  
   op.print();         //在一行上输出最终结果：4  
   注意：本题所涉及的类型均为int型，无需考虑其他类型。  
   现在你被安排完成这样一个级联调用的功能，给定类Number，请你完成其中的add方法和sub方法，这两个方法分别接受一个int型的参数，此外Number还有一个有参构造函数，接受的参数也是int型，最后你需要实现一个print的方法，用于打印结果。  
   【输入】  
   输入在一行中给出三个整数a、b、c，以空格间隔。  
   【输出】  
   输出a-b+c的结果。  
   【输入示例】  
   1 3 5  
   【输出示例】  
   3  
   （10分）

**我的答案：**

class Number {

private:

int result;

public:

Number(int value = 0) {

this->result = value;

}

Number& add(int value) {

this->result += value;

return \*this;

}

Number& sub(int value) {

this->result -= value;

return \*this;

}

void print() const {

cout << this->result << endl;

}

};

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   声明并实现一个Line类，表示线段。Line类包括：  
   Point类的私有对象数据成员start和end，表示线段的两个端点。  
   有参构造函数，将线段端点设置为给定的参数。  
   成员函数slope，计算线段的斜率。  
   【输入】  
   输入4个数，分别表示点1坐标(x1, y1)和点2坐标(x2, y2)。  
   【输出】  
   输出直线的斜率。  
   【输入示例】  
   10 20 30 70  
   【输出示例】  
   2.5  
   【提示】  
   Point类可以参阅《程序基础基础——以C++为例》第5章实验1。  
   【来源】  
   《程序设计基础——以C为例》第6章实验1。

（10分）

**我的答案：**

class Point {

private:

double x;

double y;

public:

Point(int x = 0, int y = 0) {

this->x = x;

this->y = y;

}

double getX() {

return this->x;

}

double getY() {

return this->y;

}

};

class Line {

private:

Point start;

Point end;

public:

Line(Point start, Point end) {

this->start = start;

this->end = end;

}

double slope() {

return (this->end.getY() - this->start.getY()) / (this->end.getX() - this->start.getX());

}

};

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   声明和实现一个向量类MyVector，包括一个点的坐标位置x、y和z，实现其构造函数和三个友元函数，完成两个向量的加法、减法、点乘与叉乘运算。  
   【输入】  
   输入一个点的坐标位置x、y和z。  
   【输出】  
   见【输出示例】  
   【输入示例】  
   3 4 5  
   【输出示例】  
   (1,0,0)  
   (-1,-3,0)  
   (15,-10,-1)  
   18（10分）

**我的答案：**

MyVector::MyVector(int x, int y, int z) {

this->x = x;

this->y = y;

this->z = z;

}

MyVector::MyVector(const MyVector& vector) {

\*this = vector;

}

void MyVector::display() {

cout << "(" << this->x << "," << this->y << "," << this->z << ")" << endl;

}

MyVector add(MyVector &v1, MyVector &v2) {

MyVector v;

v.x = v1.x + v2.x;

v.y = v1.y + v2.y;

v.z = v1.z + v2.z;

return v;

}

MyVector sub(MyVector &v1, MyVector &v2) {

MyVector v;

v.x = v1.x - v2.x;

v.y = v1.y - v2.y;

v.z = v1.z - v2.z;

return v;

}

int dot(MyVector &v1, MyVector &v2) {

return v1.x \* v2.x + v1.y \* v2.y + v1.z \* v2.z;

}

MyVector cross(MyVector &v1, MyVector &v2) {

MyVector v;

v.x = v1.y \* v2.z - v2.y \* v1.z;

v.y = v2.x \* v1.z - v1.x \* v2.z;

v.z = v1.x \* v2.y - v2.x \* v1.y;

return v;

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   请根据main函数中对该类的操作，补充类实现部分完成代码。  
   该类有个私有静态变量count记录该类的所有对象数，主函数将会在不同语句之后输出对象数，只有正确地实现该类，保证count正确记录该类的对象数，才能输出正确的结果。  
   【输入】  
   没有输入。  
   【输出】  
   主函数的输出已经写好。

（10分）

**我的答案：**

Student() {

count++;

}

~Student() {

count++;

}

Student(int id) {

this->id = id;

count++;

}

Student(Student& student) {

this->id = student.id;

}

friend void PrintCount();

static void InitCount() {

count = 0;

}

friend void Print(Student);

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   声明并实现一个Rational类，表示有理数。有理数的＋、－、×和÷运算要求通过重载运算符来实现。Rational类包括：  
   两个int类型的私有数据成员numerator、denominator，表示分子、分母。  
   一个私有成员函数gcd，用于求分子、分母的最大公约数。  
   一个私有成员函数compare，用于判断两个有理数之间的大小关系。一个有理数与另一个有理数相比是大于、等于还是小于，分别返回1、0和-1。  
   一个带默认参数值的构造函数，将分子、分母设置为给定的参数。分子、分母的默认参数值分别为0、1。  
   两个访问器函数getNumerator、getDenominator，分别用于获取分子、分母。  
   重载算术运算符+、-、\*、/，实现有理数算术运算。  
   重载自增运算符++和自减运算符--。要同时考虑前置和后置。  
   重载复合算术赋值运算符+=、-=、\*=、/=。  
   重载关系运算符<、<=、>、>=、==、!=，判断两个有理数之间的关系。如果满足关系，返回true，否则返回false。  
   使用类型转换函数，将有理数转换为浮点数。  
   重载流提取运算符>>和流插入运算符<<。输入输出一个有理数。  
   【输入】  
   输入两行。第一行为第一个有理数的分子、分母，以空格间隔。第二行为第二个有理数的分子、分母，以空格间隔。  
   【输出】  
   见【输出示例】。  
   【输入示例】  
   7 3  
   1 3  
   【输出示例】  
   a=7/3  
   b=1/3  
   7/3+1/3=8/3  
   7/3-1/3=2  
   7/3\*1/3=7/9  
   7/3/1/3=7  
   7/3<1/3=false  
   7/3>1/3=true  
   7/3==1/3=false  
   7/3!=1/3=true  
   c=1/2  
   2.5+1/2=3  
   3/2  
   3/2  
   3/2  
   5/2  
   17/6  
   【来源】  
   《程序设计基础——以C++为例》第6章实验6。

（10分）

**我的答案：**

class Rational {

private:

int numberator;

int denominator;

int gcd(int, int);

// int compare()

void simplify(Rational&);

public:

Rational(int = 0, int = 1);

int getNumberator();

int getDenominator();

Rational operator+(Rational);

friend Rational operator+(double, Rational);

Rational operator-(Rational);

Rational operator\*(Rational);

Rational operator/(Rational);

bool operator<(Rational);

bool operator>(Rational);

bool operator==(Rational);

bool operator!=(Rational);

Rational operator++(int);

Rational operator++();

Rational operator+=(Rational);

};

int Rational::gcd(int a, int b) {

return b == 0 ? a : gcd(b, a%b);

}

void Rational::simplify(Rational& rational) {

int number = gcd(rational.numberator, rational.denominator);

if (number < 0) {

number = -number;

}

rational.numberator /= number;

rational.denominator /= number;

}

Rational::Rational(int numberator, int denominator) {

if (denominator < 0) {

numberator = -numberator;

denominator = -denominator;

}

this->numberator = numberator;

this->denominator = denominator;

}

int Rational::getNumberator(){

this->simplify(\*this);

return this->numberator;

}

int Rational::getDenominator(){

this->simplify(\*this);

return this->denominator;

}

Rational Rational::operator+(Rational rational) {

Rational temp;

temp.numberator = this->numberator\*rational.denominator + rational.numberator\*this->denominator;

temp.denominator = this->denominator\*rational.denominator;

return temp;

}

Rational operator+(double value, Rational rational) {

return Rational(5, 2) + rational;

}

Rational Rational::operator-(Rational rational) {

Rational temp;

temp.numberator = this->numberator\*rational.denominator - rational.numberator\*this->denominator;

temp.denominator = this->denominator\*rational.denominator;

return temp;

}

Rational Rational::operator\*(Rational rational) {

Rational temp;

temp.numberator = this->numberator\*rational.numberator;

temp.denominator = this->denominator\*rational.denominator;

return temp;

}

Rational Rational::operator/(Rational rational) {

Rational temp;

temp.numberator = this->numberator\*rational.denominator;

temp.denominator = this->denominator\*rational.numberator;

return temp;

}

bool Rational::operator<(Rational rational) {

return this->numberator\*rational.denominator < rational.numberator\*this->denominator;

}

bool Rational::operator>(Rational rational) {

return this->numberator\*rational.denominator > rational.numberator\*this->denominator;

}

bool Rational::operator==(Rational rational) {

return this->numberator\*rational.denominator == rational.numberator\*this->denominator;

}

bool Rational::operator!=(Rational rational) {

return this->numberator\*rational.denominator != rational.numberator\*this->denominator;

}

Rational Rational::operator++(int) {

Rational temp = \*this;

if (this->numberator < 0) {

this->numberator -= this->denominator;

} else {

this->numberator += this->denominator;

}

return temp;

}

Rational Rational::operator++() {

if (this->numberator < 0) {

this->numberator -= this->denominator;

} else {

this->numberator += this->denominator;

}

return \*this;

}

Rational Rational::operator+=(Rational rational) {

\*this = \*this + rational;

return \*this;

}

istream& operator>>(istream& in, Rational& rational) {

int numberator;

int denominator;

in >> numberator >> denominator;

rational = Rational(numberator, denominator);

return in;

}

ostream& operator<<(ostream& out, Rational rational) {

if (rational.getDenominator() == 1) {

return out << rational.getNumberator();

}

out << rational.getNumberator() << "/" << rational.getDenominator();

return out;

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   第6题中的TwoArray类，表示二维数组（矩阵），只能存放整数。修改TwoArray类，使其成为一个类模板。并新增重载关系运算符==和!=，判断两个矩阵是否相等。  
   【输入】  
   输入一个3×4矩阵和一个4×4矩阵。  
   【输出】  
   见【输出示例】  
   【输入示例】  
   0 1 2 3  
   2 3 4 5  
   4 5 6 7  
   0 1 2 3  
   3 4 5 6  
   6 7 8 9  
   9 10 11 12  
   【输出示例】  
   array1:  
     0  1  2  3  
     2  3  4  5  
     4  5  6  7  
   array2:  
     0  1  2  3  
     3  4  5  6  
     6  7  8  9  
     9 10 11 12  
   array3:  
   Row index out ofrange!  
   array1:  
     0  1  2  3  
     3  4  5  6  
     6  7  8  9  
     9 10 11 12  
   array1 == array2  
   【提示】  
   设置输出宽度为3。  
   【来源】  
   《程序设计基础——以C++为例》第6章实验8

（10分）

**我的答案：**

template<typename T>

class TwoArray {

private:

int rowSize;

int columnSize;

T (\*pArray)[4];

public:

TwoArray(int rowSize, int columnSize) {

this->rowSize = rowSize;

this->columnSize = columnSize;

pArray = new T[rowSize][4];

}

TwoArray(TwoArray& array) {

\*this = array;

}

~TwoArray() {

delete []pArray;

}

int getRowSize() {

return this->rowSize;

}

int getColumnSize() {

return this->columnSize;

}

int &operator()(int row, int column) const{

if (row >= this->rowSize) {

throw out\_of\_range("Row index out of range!");

} else if (column >= this->columnSize) {

throw out\_of\_range("Column index out of range!");

}

return this->pArray[row][column];

}

bool operator==(TwoArray array) {

if (this->rowSize != array.getRowSize() || this->columnSize != array.getColumnSize()) {

return false;

}

for (int i = 0; i < array.getRowSize(); i++) {

for (int j = 0; j < array.getColumnSize(); j++) {

if (this->pArray[i][j] != array(i, j)) {

return false;

}

}

}

return true;

}

};

template<typename T>

void printTwoArray(TwoArray<T>& array) {

for (int i = 0; i < array.getRowSize(); i++) {

for (int j = 0; j < array.getColumnSize(); j++) {

cout << setw(3) << array(i, j);

}

cout << endl;

}

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   声明并实现一个Time类，表示时间。Time类包括：  
   int类型的私有数据成员hour、minute、second，表示时、分、秒。  
   带默认参数的构造函数，将时、分、秒设置为给定的参数。时、分、秒的默认参数值为0。需要检查时、分、秒的有效性。如果超出允许范围，则抛出invalid\_argument("Invalid argument!")异常。  
   访问器函数getHour、getMinute和getSecond，分别用于访问时、分、秒。  
   成员函数setTime，用于同时修改时、分、秒。需要检查时、分、秒的有效性。如果超出允许范围，则抛出invalid\_argument("Invalid argument!")异常。  
   重载自增运算符++，将时间递增1秒。要考虑增加1秒后，时间增加到下一分钟、下一小时、下一天的情况。要考虑前置++和后置++。  
   重载关系运算符==，判断两个时间是否相等。  
   使用类型转换函数，将时间转换为秒数。  
   重载流插入运算符<<，以24小时制格式输出时间，格式为时:分:秒。  
   【输入】  
   输入一行，有3个数据，分别表示时、分、秒，以空格间隔。  
   【输出】  
   见【输出示例】  
   【输入示例】  
   23 59 57  
   【输出示例】  
   t1:23:59:57  
   t2:00:00:00  
   t1:23:59:58  
   t2:23:59:57  
   t1 != t2  
   86398  
   【来源】  
   《程序设计基础——以C++为例》第6章实验5。

（10分）

**我的答案：**

class Time {

private:

int hour;

int minute;

int second;

public:

Time(int hour = 0, int minute = 0, int second = 0) {

this->hour = hour;

this->minute = minute;

this->second = second;

if (hour > 60 || hour < 0 || minute < 0 || minute > 60 || second < 0 || second > 60) {

throw invalid\_argument("Invalid argument!");

}

}

int getHour() const {

return this->hour;

}

int getMinute() const {

return this->minute;

}

int getSecond() const {

return this->second;

}

int getTotalTime() {

int totalTime = 0;

totalTime += this->hour \* 3600;

totalTime += this->minute \* 60;

totalTime += this->second;

return totalTime;

}

Time operator++(int) {

Time time = \*this;

int totalTime = this->getTotalTime() + 1;

this->hour = totalTime / 3600 % 24;

this->minute = totalTime / 60 % 60;

this->second = totalTime % 60;

return time;

}

bool operator==(Time& time) {

return this->getTotalTime() == time.getTotalTime();

}

operator int() {

return this->getTotalTime();

}

};

ostream& operator << (ostream&, Time& time) {

return cout << setw(2) << setfill('0') << time.getHour() << ":" <<

setw(2) << setfill('0') << time.getMinute() << ":" <<

setw(2) << setfill('0') << time.getSecond() << endl;

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

**《C++程序设计实践》实验7**

**一、程序设计题（共10小题，100分）**

|  |  |
| --- | --- |
| 题型得分 | 100 |

1. 【描述】  
   声明并实现一个Rectangle类，表示矩形。Rectangle类包括：  
   double类型的私有数据成员width和height，表示矩形的宽和高。  
   带默认参数的构造函数，将矩形的宽和高设置为给定的参数。宽和高的默认参数值为1。  
   更改器函数setWidth和setHeight，分别用于修改矩形的宽和高。  
   访问器函数getWidth和getHeight，分别用于访问矩形的宽和高。  
   成员函数computeArea，返回矩形的面积。  
   成员函数computePerimeter，返回矩形的周长。  
   【输入】  
   5 40  
   10 3.5  
   【输出】  
   200 90  
   35 27  
   【来源】  
   《程序设计基础——以C++为例》第5章实验2。

（10分）

**我的答案：**

class Rectangle {

private:

double width;

double height;

public:

Rectangle() {

width = 1;

height = 1;

}

Rectangle(double w, double h) {

width = w;

height = h;

}

void setWidth(double w) {

width = w;

}

void setHeight(double h) {

height = h;

}

double getWidth() {

return width;

}

double getHeight() {

return height;

}

double computeArea() {

return width\*height;

}

double computePerimeter() {

return width\*2+height\*2;

}

};

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   利用上题的Rational类，求如下级数的和：  
   ****  
   【输入】  
   输入一个正整数n（n≤16）。  
   【输出】  
   以有理数的形式输出级数前n项的和。  
   【输入示例】  
   1  
   【输出示例】  
   1/2﻿﻿（10分）

**我的答案：**

int main() {

int n;

cin >> n;

Rational rational;

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

rational = rational.add(Rational(i, i+1));

}

rational.print();

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   自定义异常类TriangleException。该类有一个string类型的私有数据成员message，用来存放异常信息；一个无参（默认）构造函数（默认值为“Exception”）和一个有参构造函数，用来设置异常信息；成员函数what，用来显示异常信息。  
   声明并实现一个Triangle类，表示三角形。Triangle类包括：  
   double类型的私有数据成员side1、side2、side3，表示三角形三条边。  
   私有成员函数isValid，判断三条边能否构成三角形。如果能构成三角形，返回true，否则返回false。  
   私有成员函数check，判断边是否大于0。如果边大于0，返回true，否则返回false。  
   带默认参数的构造函数，将三角形三条边设置为给定的参数。三条边的默认参数值为1。需要调用check函数检查边是否大于0，如果边小于等于0，则抛出TriangleException异常；以及调用isValid函数判断更改边后能否构成三角形，如果不能构成三角形，则抛出TriangleException异常。  
   更改器函数setSide1、setSide2和setSide3，分别用于修改三角形三条边。和构造函数一样，每个更改器函数需要调用check函数和isValid函数检查边的有效性。  
   访问器函数getSide1、getSide2和getSide3，分别用于访问三角形三条边。  
   成员函数computeArea，返回三角形面积。  
   成员函数computePerimeter，返回三角形周长。  
   【输入】  
   输入三角形的三条边，边长以空格间隔。  
   【输出】  
   若三条边有负数，输出“Negative side”。  
   若三条边能构成三角形，输出三角形面积和周长，以空格间隔。  
   若不能构成三角形，输出“Don't make a triangle”。  
   【来源】  
   《程序设计基础——以C++为例》第5章实验8。

（10分）

**我的答案：**

class TriangleException {

private:

string message;

public:

TriangleException(string message = "Exception") {

this->message = message;

}

string what() {

return this->message;

}

};

class Triangle {

private:

double side1;

double side2;

double side3;

bool isValid() {

return side1+side2>side3 && side1+side3>side2 && side2+side3>side1;

}

bool check() {

return side1>0 && side2>0 && side3>0;

}

public:

Triangle(double side1 = 1, double side2 = 1, double side3 = 1) {

this->side1 = side1;

this->side2 = side2;

this->side3 = side3;

if (!check()) {

throw TriangleException("Negative side");

}

if (!isValid()) {

throw TriangleException("Don't make a triangle");

}

}

double computeArea() {

double p = (side1+side2+side3)/2;

return sqrt(p\*(p-side1)\*(p-side2)\*(p-side3));

}

double computePerimeter() {

return side1+side2+side3;

}

};

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   声明并实现一个Cylinder类，表示圆柱体。Cylinder类包括：  
   double类型的私有数据成员radius和height，分别表示圆柱体底半径和高。  
   带默认参数的构造函数，将圆柱体底半径和高设置为给定的参数。半径和高的默认参数值为1。  
   访问器函数，分别用于访问圆柱体底半径和高。  
   成员函数computeVolume，返回圆柱体体积。  
   成员函数computeSurfaceArea，返回圆柱体表面积。  
   假设PI为3.14159。  
   【输入】  
   输入圆柱体的底半径和高。  
   【输出】  
   输出圆柱体的体积和表面积。  
   【输入示例】  
   4 8  
   【输出示例】  
   402.124  
   301.593  
   【来源】  
   《程序设计基础——以C++为例》第5章实验3。

（10分）

**我的答案：**

class Cylinder {

private:

double radius;

double height;

public:

Cylinder() {

radius = 1;

height = 1;

}

Cylinder(double r, double h) {

radius = r;

height = h;

}

double computeVolume() {

return PI\*radius\*radius\*height;

}

double computeSurfaceArea() {

return PI\*radius\*radius\*2 + height\*2\*PI\*radius;

}

};

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   声明并实现一个Account类，表示银行账户。Account类包括：  
   string类型的私有数据成员id，表示账号；string类型的私有数据成员name，表示客户名；double类型的私有数据成员balance，表示账户余额；double类型的私有数据成员annualInterestRate，表示年利率。  
   有参构造函数，将账号、客户名、账户余额、年利率设置为给定的参数。  
   更改器函数setId、setName、setBalance和setAnnualInterestRate，分别用于修改账号、客户名、账户余额、年利率。  
   访问器函数getId、getName、getBalance和getAnnualInterestRate，分别用于访问账号、客户名、账户余额、年利率。  
   成员函数withdraw，从账户中取款。  
   成员函数deposit，往账户中存款。  
   成员函数computeMonthlyInterestRate，返回月利率。  
   成员函数print，输出账户信息。  
   【输入】  
   输入账号、客户名、账户余额和年利率。  
   【输出】  
   输出账号、客户名、账户余额和月利率。  
   【输入示例】  
   112233 ZhangSan 20000 4.5  
   【输出示例】  
   112233  
   ZhangSan  
   20500  
   0.375%  
   【提示】  
   假设年利率为4.5%，则以4.5作为输入值。  
   月利率=年利率/12  
   【来源】  
   《程序设计基础——以C++为例》第5章实验4。

（10分）

**我的答案：**

class Account {

private:

string id;

string name;

double balance;

double annualInterestRate;

public:

Account(string accountId, string accountName, double accountBalance, double accountAnnualInterestRate) {

id = accountId;

name = accountName;

balance = accountBalance;

annualInterestRate = accountAnnualInterestRate;

}

void withdraw(double amount) {

balance -= amount;

}

void deposit(double amount) {

balance += amount;

}

void print() {

cout << id << endl;

cout << name << endl;

cout << balance << endl;

cout << annualInterestRate/12 << "%" << endl;

}

};

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   自定义异常类NegativeNumberException，表示对负数执行操作时出现的异常，如计算负数的平方根。该类有一个string类型的私有数据成员message，用来存放异常信息；一个无参（默认）构造函数和一个有参构造函数，用来设置异常信息；成员函数what，用来显示异常信息。在main函数中，让用户输入某个数，并调用squareRoot函数，计算该数的平方根。如果输入的是负数，squareRoot函数将抛出NegativeNumberException异常，否则返回该数的平方根。  
   【输入】  
   输入一个数。  
   【输出】  
   输出该数的平方根或者输出错误信息“Invalid argument!”。  
   【输入示例】  
   -8  
   【输出示例】  
   Invalid argument!  
   【来源】  
   《程序设计基础——以C++为例》第5章实验7。

（10分）

**我的答案：**

class NegativeNumberException {

private:

string message;

public:

NegativeNumberException() {

message = "Invalid argument!";

}

NegativeNumberException(string str) {

message = str;

}

string what() {

return message;

}

};

double squareRoot(double value) {

if (value < 0) {

throw NegativeNumberException();

}

return sqrt(value);

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   声明并实现一个Point类，表示直角坐标系中的一个点。Point类包括：  
   double类型的私有数据成员x和y，表示坐标。  
   无参（默认）构造函数，将坐标设置为原点。  
   有参构造函数，将坐标设置为给定的参数。  
   访问器函数getX和getY，分别用于访问点的x坐标和y坐标。  
   【输入】  
   0,0 4,5  
   【输出】  
   (0,0)  
   (4,5)  
   【来源】  
   《程序设计基础——以C++为例》第5章实验1。

（10分）

**我的答案：**

class Point {

private:

double x;

double y;

public:

Point() {

x = 0;

y = 0;

}

Point(double x1, double y1) {

x = x1;

y = y1;

}

double getX() { return x; }

double getY() { return y; }

};

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   有理数是由分子和分母组成的a/b形式的数，a是分子，b是分母。例如，1/3、3/4和10/4等都是有理数。有理数不能以0为分母，但可以以0为分子。整数a等价于有理数a/1。  
   有理数用于包含分数的精确运算。例如，1/3=0.333333……，这个数是不能用浮点数精确表示，为了得到精确的结果，必须使用有理数。  
   一个有理数可能有很多与其值相等的其他有理数，例如，1/3=2/6=3/9=4/12。为简单起见，用1/3表示所有值等于1/3的有理数。因此，需要对有理数进行优化，使分子和分母之间没有公约数（1除外）。求分子和分母绝对值的最大公约数，然后将分子和分母都除以此最大公约数，得到有理数的优化表示形式。  
   声明并实现一个Rational类，表示有理数。Rational类包括：  
   int类型的私有数据成员numerator、denominator，表示分子、分母。  
   私有成员函数gcd，用于求分子、分母的最大公约数。  
   带默认参数的构造函数，将分子、分母设置为给定的参数。分子、分母的默认参数值分别为0、1。  
   访问器函数getNumerator、getDenominator，分别用于访问分子、分母。  
   成员函数add、subtract、multiply、divide，实现有理数的＋、－、×和÷运算。  
   成员函数equals，判断一个有理数与另一个有理数是否相等。如果相等，返回true，否则返回false。  
   成员函数doubleValue，将有理数转换为浮点数。  
   成员函数print，输出一个有理数。若分母为0，则输出“Inf”。  
   【输入】  
   4/2 2/3  
   【输出】  
   2  
   2/3  
   8/3  
   4/3  
   4/3  
   3  
   a!=b  
   0.666667  
   【来源】  
   《程序设计基础——以C++为例》第5章实验9。

（10分）

**我的答案：**

class Rational {

private:

int numberator;

int denominator;

int gcd(int a, int b) {

return b == 0 ? a : gcd(b, a%b);

}

Rational simplify(Rational &rational) {

int number = gcd(rational.numberator, rational.denominator);

if (number < 0) {

number = -number;

}

rational.numberator /= number;

rational.denominator /= number;

return rational;

}

public:

Rational(int numberator = 0, int denominator = 1) {

this->numberator = numberator;

this->denominator = denominator;

\*this = simplify(\*this);

}

Rational add(Rational &rational) {

Rational temp;

temp.numberator = numberator\*rational.denominator + rational.numberator\*denominator;

temp.denominator = denominator\*rational.denominator;

return temp;

}

Rational subtract(Rational &rational) {

Rational temp;

temp.numberator = numberator\*rational.denominator - rational.numberator\*denominator;

temp.denominator = denominator\*rational.denominator;

return temp;

}

Rational multiply(Rational &rational) {

Rational temp;

temp.numberator = numberator\*rational.numberator;

temp.denominator = denominator\*rational.denominator;

return temp;

}

Rational divide(Rational &rational) {

Rational temp;

temp.numberator = numberator\*rational.denominator;

temp.denominator = denominator\*rational.numberator;

return temp;

}

bool equals(Rational &rational) {

return numberator == rational.numberator && denominator == rational.denominator;

}

double doubleValue() {

double result = double(numberator)/double(denominator);

return result;

}

void print() {

if (denominator == 0) {

cout << "Inf" << endl;

return;

}

\*this = simplify(\*this);

if (denominator == 1) {

cout << numberator << endl;

} else {

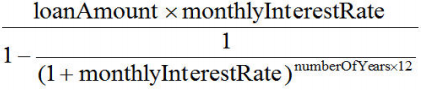
cout << numberator << "/" << denominator << endl;

}

}

};

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   声明并实现一个Loan类，表示贷款。Loan类包括：  
   double类型的私有数据成员loanAmount，表示贷款额。double类型的私有数据成员annualInterestRate，表示贷款年利率。int类型的私有数据成员numberOfYears，表示贷款年限。  
   有参构造函数，将贷款额、贷款年利率、贷款年限设置为给定的参数。  
   更改器函数setLoanAmount、setAnnualInterestRate和setNumberOfYears，分别用于修改贷款额、贷款年利率、贷款年限。  
   访问器函数getLoanAmount、getAnnualInterestRate和getNumberOfYears，分别用于访问贷款额、贷款年利率、贷款年限。  
   成员函数getMonthlyPayment，返回月还款额。  
   成员函数getTotalPayment，返回总还款额。  
   【输入】  
   输入贷款额、贷款年利率、贷款年限。  
   【输出】  
   月还款额和总还款额。  
   【输入示例】  
   60000  
   6.25  
   15  
   【输出示例】  
   514.454  
   92601.7  
   【提示】  
   假设年利率为6.25%，则以6.25作为输入值。  
   计算月还款额的公式如下：  
   ****  
   【来源】  
   《程序设计基础——以C++为例》第5章实验5。

（10分）

**我的答案：**

class Loan {

private:

double loanAmount;

double annualInterestRate;

int numberOfYears;

public:

Loan(double amount, double loanAnnualInterestRate, int loanNumberOfYears) {

loanAmount = amount;

annualInterestRate = loanAnnualInterestRate/100;

numberOfYears = loanNumberOfYears;

}

double getMonthlyPayment() {

double monthlyInterestRate = annualInterestRate/12;

return (loanAmount\*monthlyInterestRate)/(1-(1/pow(1+monthlyInterestRate, numberOfYears\*12)));

}

double getTotalPayment() {

return getMonthlyPayment()\*numberOfYears\*12;

}

};

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   声明并实现一个Time类，表示时间。Time类包括：  
   int类型的私有数据成员hour、minute、second，表示时、分、秒。  
   带默认参数的构造函数，将时、分、秒设置为给定的参数。时、分、秒的默认参数值为0。需要检查时、分、秒的有效性。若超出允许范围，则抛出invalid\_argument标准异常。  
   更改器函数setHour、setMinute和setSecond，分别用于修改时、分、秒。每个更改器函数都要检查对应的时、分、秒的有效性。若超出允许范围，则抛出invalid\_argument标准异常。  
   访问器函数getHour、getMinute和getSecond，分别用于访问时、分、秒。  
   成员函数setTime，用于同时修改时、分、秒。需要检查时、分、秒的有效性。若超出允许范围，则抛出invalid\_argument标准异常。  
   成员函数printTime24，以24小时制格式输出时间，格式为时:分:秒。  
   成员函数printTime12，以12小时制格式输出时间，格式为上午或下午时:分:秒。  
   成员函数tick，将时间递增1秒。要考虑增加1秒后，时间增加到下一分钟、下一小时、下一天的情况。  
   【输入】  
   没有输入。  
   【输出】  
   00:00:00  
   AM12:00:00  
   02:00:00  
   AM2:00:00  
   21:34:00  
   PM9:34:00  
   12:25:42  
   PM12:25:42  
   Invalid argument!  
   00:00:00  
   AM12:00:00  
   【来源】  
   《程序设计基础——以C++为例》第5章实验6。

（10分）

**我的答案：**

class Time {

private:

int hour;

int minute;

int second;

public:

Time(int h = 0, int m = 0, int s = 0) {

if (h<0 || h>23 || m<0 || m>59 || s<0 || s>59) {

throw invalid\_argument("Invalid argument!");

}

hour = h;

minute = m;

second = s;

}

void printTime24() {

cout << setw(2) << setfill('0') << hour << ":";

cout << setw(2) << setfill('0') << minute << ":";

cout << setw(2) << setfill('0') << second << endl;

}

void printTime12() {

cout << (hour<= 12 ? "AM" : "PM");

cout << (hour==0 ? hour+12 : hour<=12 ? hour : hour-12) << ":";

cout << setw(2) << setfill('0') << minute << ":";

cout << setw(2) << setfill('0') << second << endl;

}

void tick() {

int totalSeconds = hour\*3600 + minute\*60 + second + 1;

hour = totalSeconds/3600 % 24;

minute = totalSeconds/60 % 60;

second = totalSeconds%60;

}

};

|  |  |
| --- | --- |
| 题目得分 | 10 |

**《C++程序设计实践》实验6**

**一、程序设计题（共10小题，100分）**

|  |  |
| --- | --- |
| 题型得分 | 100 |

1. 【描述】  
   定义和调用函数：bool isAnagram(string str1, string str2)，检查两个单词是否是字母易位词，如果是，返回true；否则返回false。两个单词如果包含相同的字母，次序不同，则称为字母易位词（anagram）。例如，“silent”和“listen”是字母易位词。  
   【输入】  
   输入有两行，分别对应两个单词。  
   【输出】  
   若两个单词是字母易位词，输出true，否则输出false。  
   【输入示例】  
   silent  
   listen  
   【输出示例】  
   true  
   【来源】  
   《程序设计基础——以C++为例》第4章实验题11。

（10分）

**我的答案：**

bool isAnagram(string str1, string str2) {

sort(str1.begin(), str1.end());

sort(str2.begin(), str2.end());

return str1 == str2;

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   本题要求编写程序，计算N个有理数的平均值。要求使用结构表示有理数。  
   【输入】  
   输入第1行给出正整数N（<=100）；第2行中按照“a1/b1 a2/b2 ……”的格式给出N个分数形式的有理数，其中分子和分母全是整型范围内的整数；如果是负数，则负号一定出现在最前面。  
   【输出】  
   在一行中按照“a/b”的格式输出N个有理数的平均值。注意必须是该有理数的最简分数形式，若分母为1，则只输出分子。  
   【输入示例1】  
   4  
   1/2 1/6 3/6 -5/10  
   【输出示例1】  
   1/6  
   【输入示例2】  
   2  
   4/3 2/3  
   【输出示例2】  
   1

（10分）

**我的答案：**

#include <cstdio>

#include <cmath>

int main() {

int n;

Rational a[N];

scanf("%d", &n);

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

scanf("%d/%d", &a[i].n, &a[i].d);

}

// 分子总数

int totalN = a[0].n;

// 分母总数

int totalD = a[0].d;

if (totalD == 0) {

return 0;

}

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

totalN = totalN\*a[i].d + a[i].n\*totalD;

totalD \*= a[i].d;

}

// 平均值, 就是分母乘上n

totalD \*= n;

// 算出分子分母的最大公约数

int number = gcd(totalN, totalD);

totalN /= number;

totalD /= number;

if (totalD == 1) {

cout << totalN;

} else {

cout << totalN << "/" << totalD << endl;

}

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   输入5个字符串，输出其中最大的字符串（按照字典顺序）。  
   【输入】  
   输入5个字符串。  
   【输出】  
   输出5个字符串中最大的字符串。  
   【输入示例】  
   red  
   blue  
   yellow  
   green  
   purple  
   【输出示例】  
   yellow  
   【来源】  
   《程序设计基础——以C++为例》第4章实验10。

（10分）

**我的答案：**

#include <iostream>

#include <cstring>

#include <string>

#include <cstdio>

using namespace std;

int main() {

string a[5];

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

cin >> a[i];

}

string maxString = a[0];

for (string str : a) {

if (str > maxString) {

maxString = str;

}

}

cout << maxString;

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   判断输入的一个字符串是否为回文串，若是输出“Yes”，否则输出“No”。回文串是指正读和反读都一样的字符串，如level。  
   【输入】  
   输入一个字符串。  
   【输出】  
   输出“Yes”或“No”。  
   【输入示例】  
   abcddcba  
   【输出示例】  
   Yes  
   【来源】  
   《程序设计基础——以C++为例》第4章实验8。

（10分）

**我的答案：**

#include <iostream>

#include <string>

using namespace std;

int main() {

string str1;

cin >> str1;

string str2(str1.rbegin(), str1.rend());

cout << (str1 == str2 ? "Yes" : "No");

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   通讯录中的一条记录包含下述基本信息：朋友的姓名、出生日期、性别、固定电话号码、移动电话号码。本题要求编写程序，录入N条记录，并且根据要求显示任意某条记录。  
   【输入】  
   输入在第1行给出正整数N（<=10）；随后N行，每行按照格式“姓名 生日 性别 固话 手机”给出一条记录。其中“姓名”是不超过10个字符、不包含空格的非空字符串；生日按“yyyy/mm/dd”的格式给出年月日；性别用“M”表示“男”、“F”表示“女”；“固话”和“手机”均为不超过15位的连续数字，前面有可能出现“+”。  
   在通讯录记录输入完成后，最后一行给出正整数K，并且随后给出K个整数，表示要查询的记录编号（从0到N-1顺序编号）。数字间以空格分隔。  
   【输出】  
   对每一条要查询的记录编号，在一行中按照“姓名 固话 手机 性别 生日”的格式输出该记录。若要查询的记录不存在，则输出“Not Found”。  
   【输入示例】  
   3  
   Chris 1984/03/10 F +86181779452 13707010007  
   LaoLao 1967/11/30 F 057187951100 +8618618623333  
   QiaoLin 1980/01/01 M 84172333 10086  
   2 1 7  
   【输出示例】  
   LaoLao 057187951100 +8618618623333 F 1967/11/30  
   Not Found（10分）

**我的答案：**

#include <cstdio>

int main() {

int n;

Person a[N];

cin >> n;

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

cin >> a[i].name >> a[i].birthday >> a[i].sex >> a[i].fixed >> a[i].mobile;

}

int k, j;

cin >> k;

while (k-- > 0) {

cin >> j;

if (j < n) {

cout << a[j].name << " " << a[j].fixed << " " << a[j].mobile << " " << a[j].sex << " " << a[j].birthday << endl;

} else {

cout << "Not Found" << endl;

}

}

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   本题要求编写程序，比较两个有理数的大小。要求用结构表示有理数。  
   【输入】  
   输入在一行中按照“a1/b1 a2/b2”的格式给出两个分数形式的有理数，其中分子和分母全是整型范围内的正整数。  
   【输出】  
   在一行中按照“a1/b1 关系符 a2/b2”的格式输出两个有理数的关系。其中“>”表示“大于”，“<”表示“小于”，“=”表示“等于”。  
   【输入示例1】  
   1/2 3/4  
   【输出示例1】  
   1/2 < 3/4  
   【输入示例2】  
   6/8 3/4  
   【输出示例2】  
   6/8 = 3/4

（10分）

**我的答案：**

#include <cstdio>

int gcd(int a, int b) {

return b == 0 ? a : gcd(b, a%b);

}

int main() {

Rational a[2];

scanf("%d/%d %d/%d", &a[0].n, &a[0].d, &a[1].n, &a[1].d);

int a1, b1, a2, b2;

a1 = a[0].n \* a[1].d;

b1 = a[0].d \* a[1].d;

a2 = a[1].n \* a[0].d;

b2 = a[1].d \* a[0].d;

int sum1 = gcd(a1, b1);

a1 /= sum1;

int sum2 = gcd(a2, b2);

a2 /= sum2;

if (a1 > a2) {

cout << a[0].n << "/" << a[0].d << " > " << a[1].n << "/" << a[1].d;

} else if (a1 == a2) {

cout << a[0].n << "/" << a[0].d << " = " << a[1].n << "/" << a[1].d;

} else {

cout << a[0].n << "/" << a[0].d << " < " << a[1].n << "/" << a[1].d;

}

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   本题要求编写程序，计算两个有理数的和。要求使用结构表示有理数。  
   【输入】  
   输入在一行中按照“a1/b1 a2/b2”的格式给出两个分数形式的有理数，其中分子和分母全是整型范围内的正整数。   
   【输出】  
   在一行中按照“a/b”的格式输出两个有理数的和。注意必须是该有理数的最简分数形式，若分母为1，则只输出分子。  
   【输入示例1】  
   1/3 1/6  
   【输出示例1】  
   1/2  
   【输入示例2】  
   4/3 2/3  
   【输出示例2】  
   2

（10分）

**我的答案：**

#include <cstdio>

int main() {

Rational a[2];

scanf("%d/%d %d/%d", &a[0].n, &a[0].d, &a[1].n, &a[1].d);

int c, b;

c = a[0].n \* a[1].d + a[1].n \* a[0].d;

b = a[0].d \* a[1].d;

int sum1 = gcd(c, b);

c /= sum1;

b /= sum1;

if (b == 1) {

cout << c;

} else {

cout << c << "/" << b;

}

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   输入一个字符串，求出其中最长的英文单词的长度，并输出。单词之间只能用空格间隔。  
   【输入】  
   输入一个字符串。  
   【输出】  
   输出字符串中最长的英文单词的长度。  
   【输入示例】  
   Nice to meet you  
   【输出示例】  
   4  
   【C++代码】  
   《程序设计基础——以C++为例》第4章实验题12。

（10分）

**我的答案：**

#include <iostream>

#include <cstdio>

#include <cstring>

#include <string>

using namespace std;

int main() {

char ch;

string str;

int maxLength = -1;

while(cin >> str) {

int length = str.length();

if (length > maxLength) {

maxLength = length;

}

ch=getchar();

if (ch == '\n') break;

}

cout << maxLength;

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   输入一个字符串，统计并输出该字符串中26个英文字母（不区分大小写）出现的次数。  
   【输入】  
   输入一个字符串。  
   【输出】  
   分行输出26个英文字母（不区分大小写）出现的次数。  
   【输入示例】  
   I am a student.  
   【输出示例】  
   a:2  
   d:1  
   e:1  
   i:1  
   m:1  
   n:1  
   s:1  
   t:2  
   u:1  
   【来源】  
   《程序设计基础——以C++为例》第4章实验9。

（10分）

**我的答案：**

#include <iostream>

#include <string>

using namespace std;

int main() {

int a[1003]={0};

string str;

getline(cin, str);

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

if (str[i] >= 'A' && str[i] <= 'Z') {

str[i] += 32;

}

++a[str[i]];

}

for (int i = 97; i < 122; i++) {

if (a[i] > 0) {

cout << char(i) << ":" << a[i] << endl;

}

}

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   输入20个数，每个数都在1～10之间，求1～10中的众数（众数就是出现次数最多的数，如果存在一样多次数的众数，则输出值较小的）。  
   【输入】  
   输入20个1～10之间的数。  
   【输出】  
   输出20个1～10之间的数的众数。  
   【输入示例】  
   5 1 5 10 3 5 3 4 8 6 8 3 6 5 10 7 10 2 6 2  
   【输出示例】  
   5

（10分）

**我的答案：**

int comp(const void \*p, const void \*q) {

return ((struct Data\*)q)->count - ((struct Data\*)p)->count;

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

**《C++程序设计实践》实验5**

**一、程序设计题（共10小题，100分）**

|  |  |
| --- | --- |
| 题型得分 | 100 |

1. 【描述】  
   编写程序，创建一个m×n（2≤m、n≤10）的矩阵，输入矩阵的值，找出该矩阵的鞍点，鞍点是指本行最大、本列最小的元素，可能没有鞍点，也可能有多个鞍点。简单起见，只考虑一个鞍点和没有鞍点的情况。测试数据保证矩阵中任意两个数互不相等。  
   【输入】  
   第一行输入矩阵的行列数m和n，以空格间隔。  
   接着按矩阵的行列数输入矩阵的值。  
   【输出】  
   按“行 列 鞍点值”的格式输出鞍点，若不存在鞍点，则输出“No saddle point”。  
   【输入示例】  
   3 4  
   41 89 31 39  
   96 94 15 20  
   40 96 86 11   
   【输出示例】  
   0 1 89  
   【来源】  
   《程序设计基础——以C++为例》第4章实验6。

（10分）

**我的答案：**

#include <iostream>

#include <cstring>

#include <string>

#include <cstdio>

#include <cmath>

using namespace std;

int main() {

int m, n;

int i, j;

int a[102][102];

scanf("%d %d", &m, &n);

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

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

scanf("%d", &a[i][j]);

}

}

int rowMax;

int index[2];

int columnMin[103];

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

columnMin[i] = a[0][i];

for (j = 0; j < m; j++) {

columnMin[i] = min(columnMin[i], a[j][i]);

}

}

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

rowMax = a[i][0];

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

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

rowMax = a[i][j];

index[0] = i;

index[1] = j;

}

}

if (rowMax == columnMin[index[1]]) {

cout << index[0] << " " << index[1] << " " << rowMax;

return 0;

}

}

cout << "No saddle point" << endl;

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   输入10个整数，存放在一维数组中，找出值最大和最小的元素，输出最大值、最小值及它们所在的元素下标。  
   【输入】  
   在一行中输入10个整数，整数以空格间隔。  
   【输出】  
   第一行输出最大值及其所在的元素下标，最大值和下标以空格间隔。  
   第二行输出最小值及其所在的元素下标，最小值和下标以空格间隔。  
   【输入示例】  
   1 3 57 9 6 0 8 2 4  
   【输出示例】  
   9 4  
   0 6  
   【来源】  
   《程序设计基础——以C++为例》第4章实验1。

（10分）

**我的答案：**

#include <iostream>

#include <cstring>

#include <string>

#include <cstdio>

#include <cmath>

using namespace std;

int main() {

int i;

int a[10];

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

scanf("%d", &a[i]);

}

int maxValue = a[0];

int minValue = a[0];

int index1 = 0;

int index2 = 0;

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

if (maxValue < a[i]) {

maxValue = a[i];

index1 = i;

}

if (minValue > a[i]) {

minValue = a[i];

index2 = i;

}

}

cout << maxValue << " " << index1 << endl;

cout << minValue << " " << index2 << endl;

return 0;

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   输入一个正整数n（2≤n≤10）和n×n矩阵a中的元素，如果a是上三角矩阵，输出“Yes”，否则输出“No”。  
   【输入】  
   第一行为正整数n，表示矩阵大小。  
   接着n行，每一行n个整数，整数以空格间隔。  
   【输出】  
   输出“Yes”或“No”。  
   【输入示例】  
   3  
   3 4 5  
   1 2 3  
   1 3 4  
   【输出示例】  
   No  
   【提示】  
   用二维数组表示n×n矩阵时（i表示行下标，j表示列下标），则：  
   主对角线i==j，副对角线i + j == n – 1。  
   上三角矩阵i<=j。  
   下三角矩阵i>=j。  
   【来源】  
   《程序设计基础——以C++为例》第4章实验5。

（10分）

**我的答案：**

#include <iostream>

#include <iomanip>

#include <cstdio>

using namespace std;

int main() {

int n;

int i , j;

int a[103][103];

scanf("%d", &n);

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

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

scanf("%d", &a[i][j]);

}

}

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

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

if (a[i][j]) {

cout << "No" << endl;

return 0;

}

}

}

cout << "Yes" << endl;

return 0;

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   编写程序，以左下三角的形式输出前n行杨辉三角形。  
   【输入】  
   输入在一行中给出n（1≤n≤10）。  
   【输出】  
   以左下三角的格式输出前n行杨辉三角形。每个数字占固定4位。  
   【输入示例】  
   5  
   【输出示例】  
      1  
      1   1  
      1   2   1  
      1   3   3   1  
      1   4   6   4   1  
   【来源】  
   《程序设计基础——以C++为例》第4章实验7。

（10分）

**我的答案：**

#include <iostream>

#include <iomanip>

#include <cstdio>

using namespace std;

int main() {

int i, j, n;

int a[102][102];

scanf("%d", &n);

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

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

if (i == j || j == 0) {

a[i][j] = 1;

cout << setw(4) << a[i][j];

} else {

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

cout << setw(4) << a[i-1][j] + a[i-1][j-1];

}

}

cout << endl;

}

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   编写程序，定义和调用函数：void mulMatrix(int \*\*a, int \*\*b, int \*\*c, int cRow, int cCol, intaCol)，实现矩阵a和矩阵b相乘，结果存放在矩阵c中。  
   【输入】  
   第一行输入矩阵a的行数和列数，以空格间隔。  
   第二行输入矩阵b的列数（矩阵b的函数等于矩阵a的列数）。  
   接着按矩阵a的行列数输入矩阵a的值。  
   然后按矩阵b的行列数输入矩阵b的值。  
   【输出】  
   按矩阵c的行列数输出矩阵a和矩阵b相乘的结果矩阵c。元素间以空格分隔，但行末不得有多余空格。  
   【输入示例】  
   3 2  
   3  
   5 7  
   8 3  
   7 4  
   12 3 6  
   4 2 7  
   【输出示例】  
   88 29 79  
   108 30 69  
   100 29 70  
   【来源】  
   《程序设计基础——以C++为例》第4章实验13。

（10分）

**我的答案：**

void mulMatrix(int \*\*a, int \*\*b, int \*\*c, int cRow, int cCol, int aCol) {

int i, j, k;

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

for (j = 0; j < cCol; j++) {

int result = 0;

for (k = 0; k < aCol; k++) {

result += a[i][k] \* b[k][j];

}

c[i][j] = result;

}

}

}

void printMatrix(int \*\*array, int arrayRow, int arrayCol) {

int i, j;

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

for (j = 0; j < arrayCol; j++) {

cout << (j == 0 ? "" : " ");

cout << array[i][j];

}

cout << endl;

}

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   输入一指定金额（以元为单位），然后输出支付该金额的各种面额的人民币数量，显示100元，50元，20元，10元，5元，1元各多少张，要求尽量使用大面额的。  
   【输入】  
   输入一个小于1000的正整数。  
   【输出】  
   分行输出，每行显示一个整数，从上到下分别表示100元，50元，20元，10元，5元，1元人民币的张数。  
   【输入示例】  
   735  
   【输出示例】  
   7  
   0  
   1  
   1  
   1  
   0  
   【来源】  
   《程序设计基础——以C++为例》第4章实验2。

（10分）

**我的答案：**

#include <iostream>

#include <cstring>

#include <string>

#include <cstdio>

#include <cmath>

#include <cstdlib>

#include <algorithm>

using namespace std;

int main() {

int n;

cin >> n;

cout << n / 100 << endl;

cout << n % 100 / 50 << endl;

cout << n % 50 / 20 << endl;

cout << n % 20 / 10 << endl;

cout << n % 10 / 5 << endl;

cout << n % 5 << endl;

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   输入n（1≤n≤100）个正整数（无序的），找出第k（k≤n）大的数。注意，第k大的数意味着从大到小排在第k位置的数。  
   【输入】  
   n  
   k  
   a1 a2 a3 a4...an  
   【输出】  
   b  
   【输入示例】  
   5  
   2  
   32 3 12 5 89  
   【输出示例】  
   32

（10分）

**我的答案：**

#include <iostream>

#include <cstring>

#include <string>

#include <cstdio>

#include <cmath>

#include <cstdlib>

#include <algorithm>

using namespace std;

int main() {

int i, n, k;

int a[102];

scanf("%d %d", &n, &k);

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

scanf("%d", &a[i]);

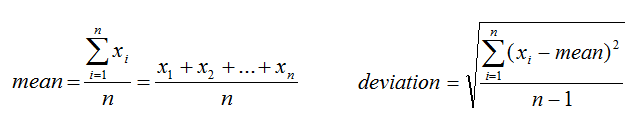
}

sort(a, a+n, greater<int>());

cout << a[k-1];

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   编写程序，输入10个数，计算这10个数的均值和标准偏差。用下面的公式计算均值mean和标准偏差deviation：  
   ****  
   【输入】  
   在第一行中给出10个数，数间以空格间隔。  
   【输出】  
   第一行为均值。  
   第二行为标准偏差。  
   【输入示例】  
   583 566 58 632 244 485 600 432 88 562  
   【输出示例】  
   425  
   216.476  
   【来源】  
   《程序设计基础——以C++为例》第4章实验4。﻿﻿

（10分）

**我的答案：**

double mean(const double x[], int arraySize) {

double sum = 0;

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

sum += x[i];

}

return sum / arraySize;

}

double deviation(const double x[], int arraySize) {

double result = 0;

double average = mean(x, arraySize);

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

result += pow(x[i] - average, 2);

}

return sqrt(result / (arraySize - 1));

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   中位数定义：一组数据按从小到大的顺序依次排列，处在中间位置的一个数或最中间两个数据的平均值（如果这组数的个数为奇数，则中位数为位于中间位置的那个数；如果这组数的个数为偶数，则中位数是位于中间位置的两个数的平均值）。  
   给出一组无序整数，求出中位数，如果求最中间两个数的平均数，向下取整即可（不需要使用浮点数）。  
   【输入】  
   包含多组测试数据，每一组测试数据的第一行为n，代表该组测试数据包含的数据个数，1≤n≤10000。  
   接着n行为n个数据。n为0时结束输入。  
   【输出】  
   输出中位数，每一组测试数据输出一行。  
   【输入示例】  
   4  
   10  
   30  
   20  
   40  
   3  
   40  
   30  
   50  
   4  
   1  
   2  
   3  
   4  
   0  
   【输出示例】  
   25  
   40  
   2

（10分）

**我的答案：**

#include <iostream>

#include <cstring>

#include <string>

#include <cstdio>

#include <cmath>

#include <cstdlib>

#include <algorithm>

using namespace std;

int main() {

int i, n;

int a[10004];

while (~scanf("%d", &n), n) {

memset(a, 0, sizeof(a));

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

scanf("%d", &a[i]);

}

sort(a, a+n);

if (n%2 == 0) {

cout << (a[n/2] + a[n/2-1])/2 << endl;

} else {

cout << a[n/2] << endl;

}

}

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   给定一组整数，要求利用数组把这组数保存起来，实现对数组中的数循环移动。假定共有n个整数，则要使前面各数顺序向后移m个位置，并使最后m个数变为最前面的m个数。  
   要求只用一个数组的方式实现，一定要保证在输出结果时，输出的顺序和数组中数的顺序是一致的。  
   【输入】  
   第一行包含一个正整数n和一个正整数m，n和m以空格间隔。  
   第二行包含n个正整数，整数以空格间隔。  
   【输出】  
   依次输出经过循环移动后数组中元素值，元素值以空格间隔。  
   【输入示例】  
   11 4  
   15 3 76 67 84 87 13 67 45 34 45  
   【输出示例】  
   67 45 34 45 15 3 76 67 84 87 13

（10分）

**我的答案：**

#include <iostream>

#include <cstring>

#include <string>

#include <cstdio>

#include <cmath>

#include <algorithm>

using namespace std;

int main() {

int i, j;

int n, m;

int a[1003];

memset(a, 0, sizeof(a));

scanf("%d %d", &n, &m);

m = m % n;

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

scanf("%d", &a[i]);

}

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

int temp = a[n-1];

for (j = n - 1; j > 0; j--) {

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

}

a[0] = temp;

}

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

cout << (i != 0 ? " " : "");

cout << a[i];

}

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

**《C++程序设计实践》实验4**

**一、程序设计题（共10小题，100分）**

|  |  |
| --- | --- |
| 题型得分 | 100 |

1. 【描述】  
   输入一个数，定义和调用函数：double squareRoot(double x)，求x的平方根，如果x是负数，函数抛出一个异常，否则返回x的平方根。要求使用异常机制来处理错误。  
   【输入】  
   输入一个数。  
   【输出】  
   如果该数是负数，则抛出异常，输出“Invalid”；否则输出该数的平方根。  
   【输入示例】  
   -6  
   【输出示例】  
   Invalid  
   【提示】  
   输出错误信息使用cout，不要使用cerr。【来源】  
   《程序设计基础——以C++为例》第3章实验12。（10分）

**我的答案：**

double squareRoot(double x) {

if (x < 0) throw 1;

return pow(x, 0.5);

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   定义和调用sum函数，返回若干个整数的和，体会函数默认参数的使用。  
   【输入】  
   没有输入。  
   【输出】  
   100  
   106  
   16  
   36  
   【提示】  
   根据sum函数调用时实参的使用和结果的输出，推测sum函数的声明和实现。（10分）

**我的答案：**

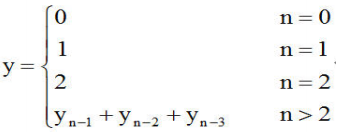
int sum(int a = 0, int b = 100, int c = 0) {

return a+b+c;

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   按如下公式：

****

求出数列的前n项（n≤20）并输出，要求每行输出5项。定义和调用函数：long sequence(int n)，计算数列第n项的值。  
【输入】  
输入一个正整数n。  
【输出】  
输出数列的前n项。每行输出5项。每项宽度为6。  
【输入示例】  
20  
【输出示例】

0 1 2 3 6

11 20 37 68 125

230 423 778 1431 2632

4841 8904 16377 30122 55403

【来源】  
《程序设计基础——以C++为例》第3章实验9。﻿﻿

（10分）

**我的答案：**

long sequence(int n) {

return n == 0 ? 0 : n == 1 ? 1 : n == 2 ? 2 : sequence(n-1) + sequence(n-2) + sequence(n-3);

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   定义和调用函数：int sum(int (\*fp)(int), int start, int end)和int f(int x)，f函数的功能是求x的平方；sum函数的功能是求f(start)+…+f(end)的值。  
   输入两个整数num1和num2，要求num1 <= num2；调用函数sum(f, num1, num2)，求出f(num1)+…+f(num2)的值。例如，假设num1为1，num2为5，则f(1)为1、f(2)为4、f(3)为9、f(4)为16、f(5)为25、f(1)+...+f(5)的值为55。  
   【输入】  
   输入两个整数num1和num2，两个正整数以空格间隔。  
   【输出】  
   输出f(num1)+…+f(num2)的值  
   【输入示例】  
   1 5  
   【输出示例】  
   55  
   【来源】  
   《程序设计基础——以C++为例》第3章实验10。（10分）

**我的答案：**

int f(int x) {

return x\*x;

}

int sum(int (\*fp)(int), int num1, int num2) {

int result = 0;

for (;num1 <= num2; num1++) {

result += fp(num1);

}

return result;

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   定义和调用如下计算面积的函数：  
   double computeArea(double radius = 1);  
   double computeArea(double width, doubleheight);  
   double computeArea(double side1, doubleside2, double side3);  
   分别计算圆、矩形和三角形的面积。如果函数返回的面积为0，表示输入数据不合法。π值为3.14159。  
   要求用函数重载实现。  
   内联函数isValid用于判断三角形合法性：  
   inline bool isValid(double side1, doubleside2, double side3);  
   【输入】  
   输入有三行：第一行圆半径；第二行矩形宽度和高度，两个实数以空格间隔；第三行三角形边长，三个实数以空格间隔。  
   【输出】  
   分行输出对应的圆、矩形和三角形面积。  
   【输入示例】  
   5  
   10 10  
   1 2 3  
   【输出示例】  
   78.5397  
   100  
   0  
   【来源】  
   《程序设计基础——以C++为例》第2章实验7。（10分）

**我的答案：**

const double PI = 3.14159;

inline bool isValid(double side1, double side2, double side3) {

return side1+side2 > side3 && side2+side3 > side1 && side1+side3 > side2;

}

double computeArea(double radius = 1) {

return PI\*radius\*radius;

}

double computeArea(double width, double height) {

return width\*height;

}

double computeArea(double side1, double side2, double side3) {

if (!isValid(side1, side2, side3)) {

return 0;

}

double p = (side1+side2+side3) / 2;

return sqrt(p\*(p-side1)\*(p-side2)\*(p-side3));

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   定义和调用三个swapValue函数，实现两个整数、两个浮点数和两个字符的交换。  
   要求用函数重载实现。函数参数可以是引用。  
   在main中输入交换前和输出交换后的值。  
   【输入】  
   输入的第一行是一个整数n（1≤n≤100），表示测试数据的总数。  
   接下来的n行，每行为测试数据，包含交换前的两个整数，两个浮点数和两个字符。  
   【输出】  
   输出包括n行，每行为测试结果，包含交换后的两个整数、两个浮点数和两个字符。  
   【输入示例】  
   2  
   12345 67890 1.23 8.79 A F  
   213879 7892 12379.2 8081.23 a u  
   【输出示例】  
   67890 12345 8.79 1.23 F A  
   7892 213879 8081.23 12379.2 u a（10分）

**我的答案：**

template<typename T>

void swapValue(T &a, T &b) {

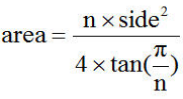
T temp = a;

a = b;

b = temp;

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   正多边形是一个有n条边的多边形，每条边的长度side相同，每个角的度数也相同。求正多边形面积的公式如下：  
   ****  
   要求定义和调用函数：double computeArea(int n,double side)，该函数返回正多边形面积。π值为3.14159。  
   【输入】  
   输入有两行：第一行一个正整数，表示正多边形的边数；第二行一个实数，表示正多边形的边长。  
   【输出】  
   输入对应的正多边形面积。  
   【输入示例】  
   5  
   3  
   【输出示例】  
   15.4843  
   【来源】  
   《程序设计基础——以C++为例》第2章实验1强化练习。

（10分）

**我的答案：**

double computeArea(int n, double side) {

const double PI = 3.14159;

return (n\*side\*side)/(4\*tan(PI/n));

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】如果一个素数可以写成****的形式，其中p是一个正整数，那么该素数就称为梅森素数。  
   例如，p为2时，****为3，3就是梅森素数。  
   注意：1不是素数。  
   要求定义和调用函数：int isPrime(int n)，如果n是素数，该函数返回1，否则返回0。  
   【输入】  
   输入一个正整数n（0≤n≤31）。  
   【输出】  
   输出p≤n的梅森素数的个数。  
   【输入示例】  
   3  
   【输出示例】  
   2

（10分）

**我的答案：**

bool isPrime(int n) {

if (n < 2) {

return false;

}

int i;

for (i = 2; i < sqrt(n); i++) {

if (n % i == 0) {

break;

}

}

return i > sqrt(n);

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   定义和调用函数模板：int compare(Tvalue1, T value2)，比较两个值的大小。如果是第一个值小于第二个值，函数返回-1；如果是第一个值等于第二个值，函数返回0；如果是第一个值大于第二个值，函数返回1。  
   【输入】  
   输入有三行：第一行两个整数，两个整数以空间间隔；第二行两个实数，两个实数以空格间隔；第三行两个字符，两个字符以空格间隔。  
   【输出】  
   分行输出对应的判断结果。  
   【输入示例】  
   5 2  
   3.8 5.4  
   a a  
   【输出示例】  
   1  
   -1  
   0  
   【来源】  
   《程序设计基础——以C++为例》第3章实验8。（10分）

**我的答案：**

template<typename T>

int compare(T value1, T value2) {

if (value1 < value2) {

return -1;

}

return value1 > value2;

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   输入一个正整数，判断它是否是回文数。要求定义和调用函数：bool isPalindrome(int n)，如果n是回文数，该函数返回true，否则返回false。回文数是指正读和反读都相同的数。  
   【输入】  
   输入一个正整数。  
   【输出】  
   如果该正整数是回文数，输出true，否则输出false。  
   【输入示例】  
   616  
   【输出示例】  
   true  
   【来源】  
   《程序设计基础——以C++为例》第3章实验3。（10分）

**我的答案：**

bool isPalindrome(int n) {

int m = n;

int result = 0;

for(int i = 0; m > 0; i++, m/=10) {

result = result\*10 + m%10;

}

return n == result;

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

**《C++程序设计实践》实验3**

**一、程序设计题（共10小题，100分）**

|  |  |
| --- | --- |
| 题型得分 | 100 |

1. 【描述】  
   编写程序，根据输入的字符以及棱形的边长，输出以该字符为填充字符的棱形。  
   【输入】  
   输入在一行中给出字符和边长。  
   【输出】  
   输出以该字符为填充字符，相应边长的棱形。  
   【输入示例】  
   A 5  
   【输出示例】
2. A
3. AAA
4. AAAAA
5. AAAAAAA
6. AAAAAAAAA
7. AAAAAAA
8. AAAAA
9. AAA
10. A

【来源】  
《程序设计基础——以C++为例》第2章实验19。（10分）

**我的答案：**

#include <iostream>

#include <cstdio>

#include <cmath>

using namespace std;

int main() {

int n;

char ch;

scanf("%c %d", &ch, &n);

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

for (int j = 1; j <= 2\*n-1; j++) {

if (j < n-i+1) {

cout << " ";

} else if (j < n+i) {

cout << ch;

}

}

cout << endl;

}

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

for (int j = 1; j <= 2\*n-1; j++) {

if (j < i+1) {

cout << " ";

} else if (j < 2\*n-i) {

cout << ch;

}

}

cout << endl;

}

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   计算数列1+(1+2)+(1+2+3)+(1+2+3+4)+…的前n项之和。  
   【输入】  
   输入一个正整数n。  
   【输出】  
   输出数列前n项的和。  
   【输入示例】  
   3  
   【输出示例】  
   10  
   【C++代码】  
   《程序设计基础——以C++》第2章实验13。

（10分）

**我的答案：**

#include <iostream>

#include <cstdio>

#include <cmath>

using namespace std;

int main() {

int n;

long long total = 0;

scanf("%d", &n);

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

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

total += j;

}

}

cout << total;

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   计算数列1+1/3+1/5+…的前n项之和。  
   【输入】  
   输入一个正整数n。  
   【输出】  
   输出数列前n项的和。  
   【输入示例】  
   10  
   【输出示例】  
   2.13326  
   【来源】  
   《程序设计基础——以C++为例》第2章实验10。

（10分）

**我的答案：**

#include <iostream>

#include <cstdio>

#include <cmath>

using namespace std;

int main() {

int n;

double a = 1;

double sum = 0;

scanf("%d", &n);

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

sum += 1/a;

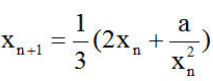
a += 2;

}

cout << sum;

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   编写程序，用迭代法求立方根  
   ****  
   求立方根的迭代公式为：  
   ****  
   当满足如下条件时：  
   ****  
   迭代停止。  
   【输入】  
   输入一个数。  
   【输出】  
   输出该数的立方根。  
   【输入示例】  
   27  
   【输出示例】  
   3  
   【来源】  
   《程序设计基础——以C++为例》第2章实验18。

（10分）

**我的答案：**

#include <iostream>

#include <cstdio>

#include <cmath>

using namespace std;

int main() {

double a1, a2, n;

double N = 1e-6;

scanf("%lf", &n);

if (n == 0) {

cout << "0" << endl;

return 0;

}

a1 = 1;

a2 = (2\*a1+n/(a1\*a1))/3;

while (fabs(a2 - a1) > N) {

a1 = a2;

a2 = (2\*a1+n/(a1\*a1))/3;

}

cout << a2;

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   给定一个十进制正整数，求其对应的二进制数中1的个数。  
   【输入】  
   第一个正整数表示有n（n > 0）个测试数据，其后n行是对应的测试数据，每行为一个正整数。  
   【输出】  
   分行输出n个正整数对应的二进制数中1的个数。  
   【输入示例】  
   4  
   2  
   100  
   1000  
   66  
   【输出示例】  
   1  
   3  
   6  
   2

（10分）

**我的答案：**

#include <iostream>

#include <cstdio>

#include <cmath>

using namespace std;

int main() {

int n, t;

scanf("%d", &t);

while (t--) {

int countOne = 0;

scanf("%d", &n);

while (n > 0) {

if (n&1) {

countOne++;

}

n /= 2;

}

cout << countOne << endl;

}

return 0;

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   某工地需要搬运砖块，已知男人一人搬3块，女人一人搬2块，小孩两人搬1块。用45人正好搬45块砖，问有多少种搬法？  
   【输入】  
   没有输入。  
   【输出】  
   输出搬砖的男人、女人和小孩数。  
   【输出示例】  
   A,B,C  
   A、B、C分别表示男人、女人、小孩数。  
   【来源】  
   《程序设计基础——以C++为例》第2章实验14。

（10分）

**我的答案：**

#include <iostream>

#include <cstdio>

#include <cmath>

using namespace std;

int main() {

int i, j, k;

for (i = 0; i <= 15; i++) {

for (j = 0; j <= 30; j++) {

for (k = 0; k <= 90; k+=2) {

if (i\*3+j\*2+k/2 == 45 && i+j+k == 45) {

printf("%d,%d,%d\n", i, j, k);

}

}

}

}

return 0;

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   输入一个正整数，从小到大输出该数所有的质因子。  
   质因数（质因数）是指能整除给定正整数的质数（素数）。  
   【输入】  
   输入一个正整数。  
   【输出】  
   分行从小到大输出该数所有的质因子。  
   【输入示例】  
   120  
   【输出示例】  
   2  
   2  
   2  
   3  
   5  
   【来源】  
   《程序设计基础——以C++为例》第2章实验17。

（10分）

**我的答案：**

#include <iostream>

#include <cstdio>

#include <cmath>

#include <cstring>

#include <string>

using namespace std;

bool prime[10004];

void isPrime() {

int i, j;

memset(prime, true, sizeof(prime));

prime[0] = prime[1] = false;

for (i = 2; i < sqrt(1e4); i++) {

if (prime[i]) {

for (j = i+i; j < 1e4; j+=i) {

prime[j] = false;

}

}

}

}

int main() {

int n;

scanf("%d", &n);

isPrime();

while (n != 1) {

for (int i = 2; i < 1e4; i++) {

if (prime[i] && n%i == 0) {

n /= i;

cout << i << endl;

break;

}

}

}

return 0;

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   在校园里，没有自行车，上课办事会很不方便。但实际上，并非去办任何事情都是骑车快，因为骑车总要找车、开锁、停车、锁车等，这要耽误一些时间。假设找到自行车，开锁并骑上自行车的时间为27秒；停车锁车的时间为23秒；步行每秒行走1.2米，骑车每秒行走3.0米。请判断走不同的距离去办事，骑车快还是走路快。  
   【输入】  
   第一个正整数表示有n（n > 0）个测试数据，其后n行是对应的测试数据，每行为一次办事要行走的距离，单位为米。  
   【输出】  
   对应每个办事要行走的距离，如果骑车快，输出一行“Bike”；如果走路快，输出一行“Walk”；如果一样快，输出一行“All”。  
   【输入示例】  
   4  
   50  
   90  
   120  
   180  
   【输出示例】  
   Walk  
   Walk  
   Bike  
   Bike

（10分）

**我的答案：**

#include <iostream>

#include <cstdio>

#include <cmath>

#include <cstring>

#include <string>

using namespace std;

int main() {

int n, m;

scanf("%d", &n);

while (n-- > 0) {

scanf("%d", &m);

if (m > 100) {

cout << "Bike" << endl;

} else if (m == 100) {

cout << "All" << endl;

} else {

cout << "Walk" << endl;

}

}

return 0;

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   输入若干个整数，如果输入0，输入即终止。判定读入的整数中有多少个正整数、多少个负整数，并计算这些整数的总和和平均值（0不计算在内）。平均值结果保留2位小数。  
   【输入】  
   输入若干个整数，如果输入0，输入即终止。  
   【输出】  
   分行输出这些整数中的正整数个数、负整数个数、总和、平均值（0不计算在内）。  
   若只输入0，则输出：No input。  
   【输入示例】  
   -1 -2 -3 -4 -5 6 7 8 9 0  
   【输出示例】  
   4  
   5  
   15  
   1.67  
   【来源】  
   《程序设计基础——以C++为例》第2章实验12。

（10分）

**我的答案：**

#include <iostream>

#include <cstdio>

#include <cmath>

#include <cstring>

#include <string>

using namespace std;

int main() {

int n;

int sum = 0;

int count1 = 0;

int count2 = 0;

while (~scanf("%d", &n) && n != 0) {

if (n > 0) {

count1++;

} else {

count2++;

}

sum += n;

}

if (count1 == 0 & count2 == 0) {

cout << "No input";

return 0;

}

cout << count1 << endl << count2 << endl << sum << endl;

printf("%.2f", sum\*1.0/(count1+count2));

return 0;

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   计算如下式子：  
   ****  
   的值，计算到最后一项的值小于给定的阈值时为止。  
   【输入】  
   输入在一行中给出小于1的阈值。  
   【输出】  
   在一行中输出满足阈值条件的式子值，结果保留6位小数。  
   【输入示例】  
   0.000001  
   【输出示例】  
   2.718282  
   【来源】  
   《程序设计基础——以C++为例》第2章实验11强化练习。﻿﻿

（10分）

**我的答案：**

#include <iostream>

#include <iomanip>

#include <cstdio>

#include <cmath>

#include <cstring>

#include <string>

using namespace std;

int main() {

double i, j, n;

double e = 1;

double m = 1;

scanf("%lf", &n);

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

for (m = 1, j = 1; j <= i; j++) {

m \*= j;

}

if (1/m < n) {

break;

}

e += 1/m;

}

cout << fixed << setprecision(6) << e << endl;

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

**《C++程序设计实践》实验2**

**一、程序设计题（共10小题，100分）**

|  |  |
| --- | --- |
| 题型得分 | 100 |

1. 【描述】  
   输入a、b和c，若它们能构成三角形，则输出三角形周长，否则输出“Invalid”。  
   【输入】  
   输入a、b和c。  
   【输出】  
   输出三角形周长或“Invalid”。  
   【输入示例1】  
   1 2 3  
   【输出示例1】  
   Invalid  
   【输入示例2】  
   3 3 3  
   【输出示例2】  
   9

（10分）

**我的答案：**

#include <iostream>

using namespace std;

int main() {

double a, b, c;

cin >> a >> b >> c;

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

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

} else {

cout << "Invalid" << endl;

}

return 0;

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   输入一个整数，检查它是否能同时被2和3整除，是否被2或3整除，是否被2或3整除且只被其一整除。  
   【输入】  
   输入一个整数。  
   【输出】  
   分行输出该整数是否能同时被2和3整除，是否被2或3整除，是否被2或3整除且只被其一整除。见输出示例。  
   【输入示例】  
   18  
   【输出示例】  
   18 divisible by 2 and 3? true  
   18 divisible by 2 or 3? true  
   18 divisible by 2 or 3, but not both? false  
   【来源】  
   《程序设计基础——以C++为例》第2章实验6。

（10分）

**我的答案：**

#include <iostream>

using namespace std;

int main() {

int a;

cin >> a;

cout << a << " divisible by 2 and 3? " << (!a%2 && !a%3 ? "true":"false") << endl;

cout << a << " divisible by 2 or 3? " << (!a%2 || !a%3 ? "true":"false") << endl;

cout << a << " divisible by 2 or 3, but not both? " << ((a%2 && !a%3) || (!a%2 && a%3) ? "true":"false") << endl;

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   求一元二次方程  
   ****  
   的根，系数a、b、c为浮点数。  
   【输入】  
   输入a、b和c。  
   【输出】  
   若无穷解，则输出：Infinitely solution。  
   若无解，则输出：No solution。  
   若是一个实根，则输出格式为：x=…，数字、符号之间没有空格，结果保留2位小数。  
   若两个实根相等，则输出格式为：x1=x2=...，数字、符号之间没有空格，结果保留2位小数。  
   若是两个实根，则输出格式为：x1=...;x2=...，数字、符号之间没有空格，结果保留2位小数。  
   若是虚根，则输出：Imaginary root。  
   【输入示例】  
   2.1 8.93.5  
   【输出示例】  
   x1=-0.44;x2=-3.80  
   【提示】  
   可以使用数学库中的sqrt函数和fabs函数。方程的根以及其它中间变量用double类型变量表示。  
   【来源】  
   《程序设计基础——以C++为例》第2章实验8强化练习。﻿﻿

（10分）

**我的答案：**

#include <iostream>

#include <cstdio>

#include <cmath>

using namespace std;

int main() {

double a, b, c;

scanf("%lf %lf %lf", &a, &b, &c);

double delta = b\*b-4\*a\*c;

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

printf("x=%.2f\n", -c/b);

return 0;

} else if (a == 0 && b == 0 && c == 0) {

cout << "Infinitely solution" << endl;

return 0;

} else if (a == 0 && b == 0 && c != 0) {

cout << "No solution" << endl;

return 0;

}

if (delta > 0) {

printf("x1=%.2f;x2=%.2f\n", (-b+sqrt(delta))/(2\*a), (-b-sqrt(delta))/(2\*a));

} else if (delta == 0) {

printf("x1=x2=%.2f\n", -b/(2\*a));

} else {

cout << "Imaginary root" << endl;

}

return 0;

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   输入一个四位正整数，将该整数每一位上的数字加9，然后除以10取余，作为该位上的新数字，最后将千位和十位上的数字互换，百位和个位上的数字互换，组成变换后的新四位正整数并输出。题目保证转换后的数的千位不会为0。  
   【输入】  
   输入一个四位正整数。  
   【输出】  
   输出变换后的新四位正整数。  
   【输入示例】  
   1257  
   【输出示例】  
   4601  
   【来源】  
   《程序设计基础——以C++为例》第2章实验4强化练习。

（10分）

**我的答案：**

#include <iostream>

using namespace std;

int main() {

int a;

cin >> a;

cout << (a/10%10+9)%10 << (a%10+9)%10 << (a/1000+9)%10 << (a/100%10+9)%10 << endl;

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   输入学生的考试成绩（0～100），将学生的成绩划分等级并输出。学生的成绩可分为5个等级：90～100为A级，80～89为B级，70～79为C级，60～69为D级，0～59为E级。要求用switch语句。  
   【输入】  
   输入学生的考试成绩（0～100）。  
   【输出】  
   输出等级。  
   【输入示例】  
   91  
   【输出示例】  
   A  
   【提示】  
   假设学生的成绩为score，则score/10的一个值就代表了10种情况，例如score/10为7时，就代表成绩70～79的情况。  
   【来源】  
   《程序设计基础——以C++为例》第2章实验9。

（10分）

**我的答案：**

#include <iostream>

using namespace std;

int main() {

int score;

cin >> score;

switch(score/10) {

case 10: cout << "A"; break;

case 9: cout << "A"; break;

case 8: cout << "B"; break;

case 7: cout << "C"; break;

case 6: cout <<"D"; break;

default: cout << "E"; break;

}

return 0;

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   某校学生成绩的绩点计算方法如下（百分制折合为绩点）：  
   （A）90～100分，折合4.0～5.0绩点。  
   （B）80～89分，折合3.0～3.9绩点。  
   （C）70～79分，折合2.0～2.9绩点。  
   （D）60～69分，折合1.0～1.9绩点。  
   （E）59分及其以下为不及格。  
   要求用switch语句。  
   【输入】  
   输入一个整数表示学生的成绩（0～100）。  
   【输出】  
   如果输入的学生成绩不合法，则输出“Invalid”。  
   如果输入的学生成绩为不及格，则输出“Failed”。  
   否则，输出成绩对应的绩点，结果保留一位小数。  
   【输入示例】  
   74  
   【输出示例】  
   2.4

（10分）

**我的答案：**

#include <iostream>

using namespace std;

int main() {

int score;

cin >> score;

if (score > 100 || score < 0) {

cout << "Invalid";

return 0;

}

switch(score/10) {

case 10: cout << "5.0"; break;

case 9: cout << "4." << score%10; break;

case 8: cout << "3." << score%10; break;

case 7: cout << "2." << score%10; break;

case 6: cout << "1." << score%10; break;

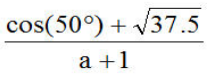
default: cout << "Failed"; break;

}

return 0;

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   输入一个整数a（a不为-1），求如下表达式的值，结果保留2位小数。  
   ****  
   【输入】  
   输入一个整数a。  
   【输出】  
   输出表达式的值，结果保留2位小数。  
   【输入示例】  
   2  
   【输出示例】  
   2.26  
   【提示】  
   求三角函数cos的值可以使用数学库中的cos(x)函数，x为弧度值。求平方根可以使用数学库中的sqrt(x)函数。  
   【来源】  
   《程序设计基础——以C++为例》第2章实验15。﻿﻿

（10分）

**我的答案：**

#include <iostream>

#include <iomanip>

#include <cmath>

using namespace std;

int main() {

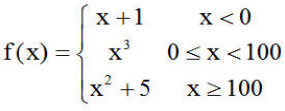
int a;

cin >> a;

cout << fixed << setprecision(2) << (cos(50\*3.14159/180) + sqrt(37.5)) / (a+1);

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   输入一个整数，输出分段函数的值。分段函数的数学定义如下：  
   ****  
   【输入】  
   输入一个整数。  
   【输出】  
   输出分段函数的值。  
   【输入示例】  
   100  
   【输出示例】  
   10005  
   【来源】  
   《程序设计基础——以C++为例》第2章实验7。﻿﻿

（10分）

**我的答案：**

#include <iostream>

using namespace std;

int main() {

int x;

cin >> x;

if (x < 0) {

cout << x + 1;

} else if (x < 100) {

cout << x\*x\*x;

} else {

cout << x\*x + 5;

}

return 0;

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   比较两个整数之间的大于、小于、等于、不等于关系。  
   【输入】  
   输入在一行中给出2个整数a和b。  
   【输出】  
   分行输出整数a和b之间的大于、小于、等于、不等于关系。  
   【输入示例】  
   5 3  
   【输出示例】  
   true  
   false  
   false  
   true  
   【来源】  
   《程序设计基础——以C++为例》第2章实验1。

（10分）

**我的答案：**

#include <iostream>

#include <cstdio>

using namespace std;

int main() {

int a, b;

cin >> a >> b;

cout << boolalpha << (a>b) << endl;

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

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

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

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   计算三个整数的和、平均值、最小值和最大值。  
   要求用条件运算符求最小值和最大值。  
   【输入】  
   输入在一行中给出整数a、b、c。  
   【输出】  
   分行输出整数a、b、c的和、平均值、最小值和最大值。  
   【输入示例】  
   5 3 -1  
   【输出示例】  
   7  
   2.33333  
   -1  
   5  
   【来源】  
   《程序设计基础——以C++为例》第2章实验2强化练习。

（10分）

**我的答案：**

#include <iostream>

using namespace std;

int main() {

int a, b, c;

cin >> a >> b >> c;

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

cout << (a+b+c)/3.0 << endl;

cout << min(min(a, b), c) << endl;

cout << max(max(a, b), c);

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

**《C++程序设计实践》实验1**

**一、程序设计题（共10小题，100分）**

|  |  |
| --- | --- |
| 题型得分 | 100 |

1. 【描述】  
   在屏幕上显示“Welcome to C++！”并换行。  
   【输入】  
   没有输入。  
   【输出】  
   Welcome to C++!  
   【来源】  
   《程序设计基础——以C++为例》第1章实验1。（10分）

**我的答案：**

#include <iostream>

using namespace std;

int main() {

cout << "Welcome to C++!" << endl;

return 0;

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   在屏幕上显示下列图案。  
   \*\*\*\*  
   \*\*\*  
   \*\*  
   \*  
   【输入】  
   没有输入。  
   【输出】  
   \*\*\*\*  
   \*\*\*  
   \*\*  
   \*  
   【来源】  
   《程序设计基础——以C++为例》第1章实验2。（10分）

**我的答案：**

#include <iostream>

using namespace std;

int main() {

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

for(int j = i; j < 4; j++) {

cout << "\*";

}

cout << endl;

}

return 0;

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   将摄氏温度转换为华氏温度。  
   转换公式为：  
     
   f表示华氏温度，c表示摄氏温度。  
   【输入】  
   输入一个摄氏温度。  
   【输出】  
   输出对应的华氏温度。  
   【输入示例】  
   100  
   【输出示例】  
   212  
   【来源】  
   《程序设计基础——以C++为例》第1章实验3。（10分）

**我的答案：**

#include <iostream>

using namespace std;

int main() {

double f, c;

cin >> c;

cout << 9.0\*c/5 + 32;

return 0;

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   编写程序，计算并输出两个正整数的和、差、积、商。题目保证输入和输出全部在整型范围内。  
   【输入】  
   输入在一行中给出2个正整数a和b。  
   【输出】  
   按照格式“A运算符B=结果”顺序输出分行输出两个正整数的和、差、积、商。  
   【输入示例】  
   5 3  
   【输出示例】  
   5+3=8  
   5-3=2  
   5\*3=15  
   5/3=1  
   【来源】  
   《程序设计基础——以C++为例》第1章实验4。（10分）

**我的答案：**

#include <iostream>

using namespace std;

int main() {

int a, b;

cin >> a >> b;

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

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

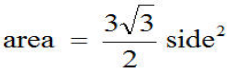
cout << a << "\*" << b << "=" << a\*b << endl;

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

return 0;

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   输入六边形的边长side，求六边形的面积area。  
   利用下面的公式计算六边形的面积：  
     
   【输入】  
   输入六边形的边长side。  
   【输出】  
   输出对应的六边形面积。  
   【输入示例】  
   5.5  
   【输出示例】  
   78.5918  
   【提示】  
   求平方根可以使用数序库中的sqrt(x)函数。  
   【来源】  
   《程序设计基础——以C++为例》第1章实验5。

（10分）

**我的答案：**

#include <iostream>

#include <cmath>

using namespace std;

int main() {

double side;

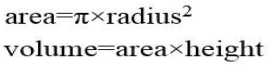
cin >> side;

cout << 3\*sqrt(3)\*side\*side/2 << endl;

return 0;

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   计算圆柱体的体积。  
   假设PI为3.14159。  
     
   【输入】  
   输入圆柱体的半径和高。  
   【输出】  
   输出对应的圆柱体体积  
   【输入示例】  
   2.5 3.5  
   【输出示例】  
   68.7223  
   【来源】  
   《程序设计基础——以C++为例》第1章实验6。（10分）

**我的答案：**

#include <iostream>

using namespace std;

int main() {

double radius, height;

cin >> radius >> height;

cout << 3.14159\*radius\*radius\*height << endl;

return 0;

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   输入一个圆环的内外半径，定义和调用函数：double computeArea(double outside, double inside)，计算圆环的面积，inside和outside分别为圆环的内外半径，题目保证外半径大于内半径，函数返回圆环的面积。  
   假设PI为3.14159。  
   【输入】  
   输入圆环的外半径和内半径。  
   【输出】  
   输出对应的圆环面积  
   【输入示例】  
   3.5 2.5  
   【输出示例】  
   18.8495  
   【来源】  
   《程序设计基础——以C++为例》第1章实验6强化练习。

（10分）

**我的答案：**

#include <iostream>

using namespace std;

const double PI = 3.14159;

double computeArea(double outside, double inside) {

double outsideArea = PI\*outside\*outside;

double insideArea = PI\*inside\*inside;

return outsideArea - insideArea;

}

int main() {

double outside, inside;

cin >> outside >> inside;

cout << computeArea(outside, inside) << endl;

return 0;

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   一只大象口渴了，要喝20升水才能解渴，但现在只有一个深h厘米，底面半径为r厘米的小圆桶（h和r都是整数）。问大象至少要喝多少桶水才会解渴。假设PI为3.14159。  
   【输入】  
   输入小圆桶的深h和底面半径r，单位厘米。  
   【输出】  
   输出大象至少要喝多少桶水（整数）。  
   【输入示例】  
   23 11  
   【输出示例】  
   3  
   【提示】  
   1升等于1000立方厘米。取整可以使用数学库中的ceil(x)函数。例如，ceil(2.1)，向上取整结果为3.0；ceil(-2.1)，向上取整结果为-2.0。

（10分）

**我的答案：**

#include <iostream>

#include <cmath>

using namespace std;

const double PI = 3.14159;

int main() {

double height, radius, volume;

cin >> height >> radius;

volume = height \* radius\*radius\*PI;

cout << ceil(20000 / volume\*1.0) << endl;

return 0;

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   编写程序，读取投资总额、年利率和年数，然后使用如下公式计算未来投资金额。  
     
   【输入】  
   一行中给出投资总额、年利率和年数，其间以空格分隔。  
   【输出】  
   一行中输出未来投资金额，结果保留2位小数。  
   【输入示例】  
   1000 3.25 1  
   【输出示例】  
   1032.99  
   【提示】  
   可以使用数学库中的pow函数来计算a的b次幂。  
   年利率转换为月利率，年数转换为月数。  
   结果x保留2位小数：  
   #include <iomanip>  
   cout << fixed << setprecision(2) << x << endl;

（10分）

**我的答案：**

#include <iostream>

#include <cmath>

#include <iomanip>

using namespace std;

int main() {

double amount, yearRate, year;

cin >> amount >> yearRate >> year;

year \*= 12;

yearRate /= 1200;

double value = amount \* pow(1+yearRate, year);

cout << fixed << setprecision(2) << value << endl;

}

|  |  |
| --- | --- |
| 题目得分 | 10 |

1. 【描述】  
   编写程序，根据火车的出发时间和达到时间计算整个旅途所用的时间。  
   【输入】  
   在一行中给出两个正整数，其间以空格分隔，分别表示火车的出发时间和到达时间。每个时间的格式为两位小时数（00～23）和两位分钟数（00～59），假设出发和到达在同一天内。  
   【输出】  
   在一行中输出该旅途所用的时间，格式为“hh:mm”，其中hh为两位小时数、mm为两位分钟数。  
   【输入示例】  
   1201 1530  
   【输出示例】  
   03:29  
   【提示】  
   #include <iomanip>  
   设置小时hour的输出宽度和填充字符：  
   cout << setw(2) << setfill('0') << hour << endl;（10分）

**我的答案：**

#include <iostream>

#include <iomanip>

using namespace std;

int main() {

int time1, time2;

int hour1, hour2;

int totalTime;

cin >> time1 >> time2;

hour1 = time1 / 100;

hour2 = time2 / 100;

totalTime = (hour2\*60 + time2%100) - (hour1\*60 + time1%100);

cout << setw(2) << setfill('0') << totalTime/60 << ":" << setw(2) << setfill('0') << totalTime%60;

return 0;

}

|  |  |
| --- | --- |
| 题目得分 | 10 |