

上海财经大学《程序设计基础》上机考试卷(A)

参考答案及评分标准

(2020 至 2021 学年 第 1 学期)

1. (循环控制。 15 分)

编写程序计算三角函数 $\cos x$ 近似值。请根据 $\cos x$ 泰勒展开式，应用循环编写程序。

$$\cos x = \sum_{k=0}^{\infty} \frac{(-1)^k}{(2k)!} x^{2k} = 1 - \frac{1}{2!} x^2 + \frac{1}{4!} x^4 - \frac{1}{6!} x^6 + \dots$$

当 $\left| \frac{(-1)^k}{(2k)!} x^{2k} \right| < 10^{-9}$ 时，循环结束。

程序如下：

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
    double x, cosX = 0;
    cout << "Please input the value of x: ";
    cin >> x;
    // 请补充完成以下缺少的代码 15 分
    double t = 1; // 2 分
    for(int k=1; fabs(t)>=1E-9; k++){ // 4 分
        cosX += t; // 3 分
        t *= x*x; // 2 分
        t /= 2*k*(2*k-1); // 2 分
        t *= -1; // 2 分
    }
    cout << "cos(" << x << ") = " << cosX << endl;
    return 0;
}
```

2. (函数调用。 15 分)

求 n 个整数的中位数。中位数是指一群数据里的一半的数据比它大，而另外一半数据比它小。

计算中位数的方法如下：

- 1) 把一组数据按照大小的顺序排列；
- 2) 如果该组数据的个数是奇数，则中间那个数就是该组数据的中位数；
- 3) 如果该组数据的个数是偶数，则中间那两个数的算术平均值就是该群数据的中位数。

//主函数如下所示，不得更改。编写 calcMedian 函数，将下面程序补充完整。

```
#include <iostream>
```

```
using namespace std;
```

```
int calcMedian(int a[], int n); // 计算数组 a 中前 n 个元素的中位数
```

```
int main() {
```

```
    int data[] = {-51, 45, 108, 132, 754, 10, 189, 909, 1200, -2};
```

```
    cout << "Median of the Array(10 elements): " << calcMedian(data, 10) << endl;
```

```
    cout << "Median of the Array(9 elements): " << calcMedian(data, 9) << endl;
```

```
    return 0;
```

```
}
```

```
int calcMedian(int a[], int n)
```

```
{
```

```
    // 请修改并补充缺少的代码
```

```
    int b[10], temp; // 1 分
```

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

```
        b[i] = a[i];
```

```
    for(int i = 0; i < n-1; i++) // 2 分
```

```
        for(int j = 0; j < n-1-i; j++) // 2 分
```

```
            if(b[j] > b[j+1]){ // 3 分
```

```
                temp = b[j]; b[j] = b[j+1]; b[j+1] = temp;
```

```
            }
```

```
    /*
```

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

```
            cout << b[i] << " ";
```

```
        cout << endl;
```

```
    */
```

```
    return (n%2==0)?(b[n/2-1] + b[n/2])/2 : b[n/2]; // 5 分
```

```
}
```

3. (改错题, 数组与字符串, 15 分)

国际标准书号 ISBN 用来唯一标识一本合法出版的图书。它包含 13 位数字, 由 5 部分组成。分别是 3 位 ENA (欧洲商品编号) 图书产品代码 “978”, 1 位国家编号, 2 位出版商编号, 6 位图书编号, 和 1 位校验数字。例如, 978-7-115-18309-5 (处理的时候可忽略横线)。其校验方法是计算加权和, 即每一位数字乘上一个系数, 然后相加求和, 系数表是固定的 1313131313131; 然后看加权和能否被 10 整除, 如果可以, 则合法。对于 978-7-115-18309-5, 校验的结果是: $(9 \times 1 + 7 \times 3 + 8 \times 1 + 7 \times 3 + 1 \times 1 + 1 \times 3 + 5 \times 1 + 1 \times 3 + 8 \times 1 + 3 \times 3 + 0 \times 1 + 9 \times 3 + 5 \times 1) \% 10 = 0$ 。以下程序可以对 13 位的 ISBN 号进行校验, 请修改其中的错误。(15 分)

```
#include <iostream>
#include <ctype.h>
#include <cstring>    // 使用 strlen 函数, 需要该头文件, 2 分
using namespace std;
// 检验 13 位 ISBN 是否合法的函数;
// 参数为仅包含数字的 ISBN 字符串;
// 如果合法则返回 true, 反之 false。
bool checkISBN13(char str[]);
int main(){
    char str1[] = "9787115183095";
    cout << str1 << "是" << (checkISBN13(str1) ? "合法" : "不合法")
        << "的 ISBN 号。" << endl;
    char str2[] = "8787115183096";
    cout << str2 << "是" << (checkISBN13(str2) ? "合法" : "不合法")
        << "的 ISBN 号。" << endl;
    system("pause");
    return 0;
}

bool checkISBN13(char str[]){
    int s[13] = {1,3,1,3,1,3,1,3,1,3,1,3,1};
    if(strlen(str) != 13){           // 改为 strlen 函数, 3 分;
        return false;               // 如果不是 13 位, 则不合法;
    }else if(str[0] != '9' || str[1] != '7' || str[2] != '8'){    // 9, 7, 8 改为字符, 3 分;
        return false;               // 如果不以 978 打头, 则不合法
    }else{
        int sum = 0;
        for(int i = 0; i < 13; i++){
            if(!isdigit(str[i])){    // 取反, 2 分;
                return false;        // 如果出现非数字的情况, 则不合法
            }
            sum += (str[i] - '0') * s[i];    // 计算加权和 字符转数字, 3 分。
        }
        if(sum % 10 != 0){           // 改为取余号, 2 分。
```

```

        return false;        // 如果加权和不能被 10 整除，则不合法
    }
}
return true;        // 如果以上条件都满足，则为合法 13 位 ISBN 号。
}

```

4. (指针与动态数组, 15 分)

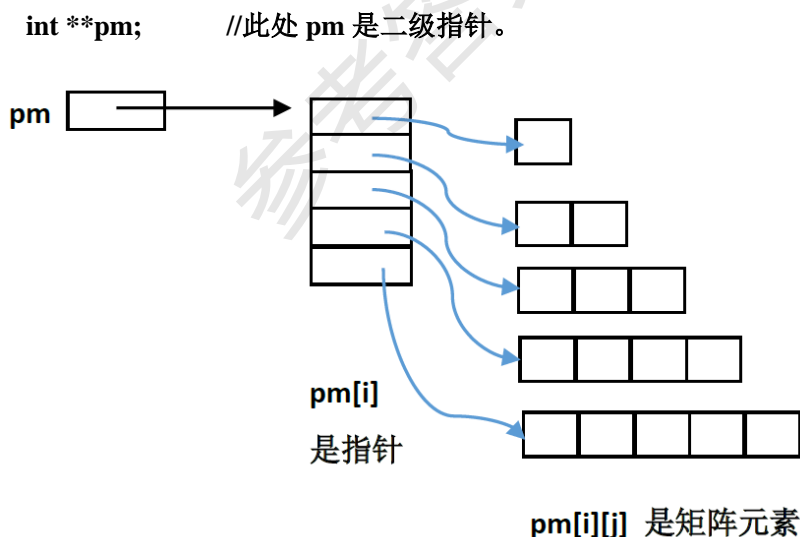
请编写程序使用动态内存来存储一个阶数为 n 的下三角矩阵（数据结构如图所示，阶数 n 在运行时由用户输入）。矩阵的元素为 $0,1,2, \dots, 9$ 的随机数。然后打印该下三角矩阵，程序结束之前，释放动态数组所占用的内存。

此处，所谓下三角矩阵是指矩阵中的上三角（不包括对角线）中的元素均为 0 。例如一个 5×5 的下三角矩阵如下

$$\begin{pmatrix} 2 & 0 & 0 & 0 & 0 \\ 8 & 6 & 0 & 0 & 0 \\ 6 & 4 & 5 & 0 & 0 \\ 8 & 7 & 3 & 7 & 0 \\ 2 & 8 & 9 & 4 & 9 \end{pmatrix}$$

请补充完成函数 `int **generatem(int n)`。该函数首先建立一个长度为 n 的动态数组，数组中每个元素为指针（该数组元素为 `int *` 类型）。然后，让这些指针依次指向随后申请得到的长度分别从 1 到 n 的动态数组（这些数组的元素均为 `int` 类型，元素的值为 $0,1,2, \dots, 9$ 的随机数）。该函数返回一个二级指针。

当下三角矩阵的阶数为 5 时，本题的数据结构示意图如下（注意：因为上三角的部分元素均为 0 ，所以此处不需要存储这部分元素。）



```

#include <iostream>
#include <cstdlib>
#include <ctime>
int ** generatem(int n);

```

```
void printm(int **p, int n);
void freememory(int **p, int n);
```

```
using namespace std;
```

```
int main(){
    int ** pm;
    int order;

    cout << "please input the order:";
    cin >> order;

    pm= generatem(order);    // 生成下三角矩阵
    printm(pm, order);       //打印下三角矩阵
    freememory(pm, order);    // 释放下三角矩阵所占内存
    return 0;
}
```

// 生成下三角矩阵，矩阵中每个元素的值，用 0,1,2, ..., 9 的随机数填充。

```
int **generatem(int n){
    int ** p;
    srand(time(NULL));
    p=new int * [n];
    // 请补充以下缺少的代码。 注：随机数可以使用函数 rand()
    for (int i=0;i<n;i++)          // 5 分
        p[i] = new int[i+1];
    for(int i=0;i<n;i++)           // 4 分
        for(int j=0;j<=i;j++){    // 3 分
            p[i][j]=rand()%10;    // 3 分
        }
    return p;
}
```

//打印下三角矩阵

```
void printm(int **p, int n){
    for(int i=0;i<n;i++){
        for(int j=0;j<=i;j++){
            cout << p[i][j];
            if(j<i) cout << ",";
        }
    }
}
```

```

        cout << endl;
    }
}
// 释放下三角矩阵所占内存
void freememory(int **p, int n){
    for(int i=0;i<n;i++)
        delete p[i];
    delete p;
}

```

5. (递归。 15 分)

假设有“ABCDE”5 个字母，如果从中任意选出 3 个字母，进行全排列。注意：用一个长度为 n 的字符串做排列，可以先选第一个元素，有 n 种选择，而对每种选择，排列后面 $n-1$ 个元素；然后再对长度为 $n-1$ 个字符串的排列：选择它的第一个元素，有 $n-1$ 种选择，而对每种选择，排列后面 $n-2$ 个元素。依此类推。请采用递归的方式实现程序，输出所有可能的结果。可以参考课本和课堂 ppt 的函数递归部分。

// 需要对书上 135 页求全排列的算法进行改造，即可得到答案。

```
#include <iostream>
```

```
#include <cstring>    // 1 分，如果使用了 strlen 函数的话。
```

```
using namespace std;
```

```
// permute 递归函数
```

// str 代表所有的字母组成的字符串，m 表示选取的字母数量，k 代表从第 k 个字母开始

```
void permute(char str[], int m, int k);
```

```
void swap(char str[], int k, int i);    // 2 分
```

```
int main(){
```

```
    char str[] = "ABCDE";
```

```
    permute(str, 3, 0);
```

```
    return 0;
```

```
}
```

```
void permute(char str[], int m, int k){
```

//请用递归的方式实现该函数。

```
    if(k == m){    // 2 分，对书上代码改造获得
```

```
        for(int i = 0; i < m; i++){    // 3 分，由于只输出部分结果，不能直接用 cout<<str;
```

```
            cout << str[i];
```

```
        }
```

```
        cout << endl;
```

```
    }else{    // 5 分，可从书上直接获得。
```

```
        for(int i = k; i < strlen(str); i++){
```

```

        swap(str, k, i);
        permute(str, m, k+1);
        swap(str, k, i);
    }
}
}

```

```

void swap(char str[], int k, int i){    // 3 分，可从书上直接抄得。
    char c = str[k];
    str[k] = str[i];
    str[i] = c;
}

```

6. （类。 13 分）

建立一个水壶的类。水壶的数据成员包含目前的水量 `vol` 以及水壶的最大容量 `cap`，均以整数表示。水壶提供以下操作：

- 1) `Bottle::Bottle(int c)`
构造函数，建立一个最大容量为 `c` 的空水壶。
- 2) `void Bottle::fill()`
填满水壶，把水量设至最大容量。
- 3) `void Bottle::empty()`
清空水壶，把水量设至 0。
- 4) `void Bottle::pour(Bottle &b)`
把水壶的水倒到水壶 `b` 里，直到水壶空了或者水壶 `b` 装满了。例子：假如两个水壶 `a` 跟 `b` 的最大容量都是 5，目前水量都是 4，那 `a.pour(b)`；之后 `a` 的水量为 3 而 `b` 的水量为 5。
- 5) `int Bottle::get_vol()`
返回水壶目前的水量。

请补充完成以下代码中有关水壶操作的实现（请勿改变主函数）。

```

#include <iostream>
using namespace std;

```

```

class Bottle {
private:
    int vol, cap;
public:
    Bottle(int);
    void fill();
    void empty();
    void pour(Bottle &);
    int get_vol();
};

```

```

void display(Bottle &, Bottle &);
int main() {
    Bottle a(5), b(3);
    a.fill(); display(a, b);
    a.pour(b); display(a, b);
    b.empty(); display(a, b);
    a.pour(b); display(a, b);
    a.fill(); display(a, b);
    a.pour(b); display(a, b);
    return 0;
}
void display(Bottle &a, Bottle &b) {
    cout << "The volumes: " << a.get_vol() << ", " << b.get_vol() << endl;
}

```

// 2 分

```

Bottle::Bottle(int c) : cap(c), vol(0) {
}

```

// 2 分

```

void Bottle::fill() {
    vol = cap;
}

```

// 2 分

```

void Bottle::empty() {
    vol = 0;
}

```

// 5 分

```

void Bottle::pour(Bottle &b) {
    if (vol + b.vol > b.cap) {
        vol -= b.cap - b.vol;
        b.vol = b.cap;
    } else {
        b.vol += vol;
        vol = 0;
    }
}

```

// 2 分

```

int Bottle::get_vol() {
    return vol;
}

```


7. (链表。12 分)

// 答案如下:

```
#include <iostream>
#include <cstring>
using namespace std;

struct Student{
    int no;
    char name[30];
    int score;
    Student *next;
};
struct StudentLinkedList{
    Student *head;
    Student *tail;
};
// 输入学生信息, 生成学生信息链表
void createStudentLinkedList(StudentLinkedList &sl);
// 输出特定成绩区间的学生信息
void rangeSearchStudentLinkedList(const StudentLinkedList &sl, int low, int high);
// 释放学生信息链表占用的存储空间
void cleanStudentLinkedList(StudentLinkedList &sl);

int main()
{
    StudentLinkedList studlist = {NULL, NULL};
    createStudentLinkedList(studlist);
    cout << endl << "-----学生信息链表生成!-----" << endl << endl;
    rangeSearchStudentLinkedList (studlist,60,90);
    cleanStudentLinkedList (studlist);
    cout << endl << "----- 学生信息链表清理完成! -----" << endl << endl;
    return 0;
}
// 通过循环逐个读入学生信息, 生成学生结点, 然后插入到链表尾部, 从而建立学生信息链表。
void createStudentLinkedList (StudentLinkedList &sl)
{
    // 生成学生信息链表的头结点
    Student *pStud = new Student;
    cout << "请输入学生的学号、姓名和成绩:";
    cin >> pStud->no >> pStud->name >> pStud->score;
    pStud->next = NULL;
    if(pStud->no >= 0)
        sl.head = sl.tail = pStud;
```

```

// 生成新的学生结点并插入到链表尾部
// 请补充代码
while(1){
    cout << "请输入学生的学号、姓名和成绩:"; // 1 分
    Student *pStud = new Student;
    cin >> pStud->no >> pStud->name >> pStud->score; // 1 分
    pStud->next = NULL;

    if(pStud->no >= 0) // 1 分
    {
        sl.tail->next = pStud;
        sl.tail = pStud;
    }
    Else // 1 分
        break;
}

void rangeSearchStudentLinkedList (const StudentLinkedList &sl, int low, int high)
{
    Student *pStud = sl.head;
    int counter = 0;
    cout << endl << "----- 成绩在" << low << " 和 " << high << "之间的学生信息
-----" << endl;
    // 遍历整个学生信息链表，输出特定成绩区间[low,high]的学生信息
    // 包括：学生学号、姓名、成绩和区间内学生总数
    // 请补充代码
    while(pStud != NULL){ // 1 分
        if(pStud->score > low && pStud->score < high){ // 1 分
            cout << " 学号: " << pStud->no
                << " 姓名: " << pStud->name
                << " 成绩: " << pStud->score
                << endl;
            counter++; // 1 分
        }
        pStud = pStud->next; // 1 分
    }
    cout << "学生总人数: " << counter << endl;
}

void cleanStudentLinkedList(StudentLinkedList &sl)
{
    // 释放学生信息链表占用的存储空间
    // 请补充代码, 4 分

```

```
Student *pStud = sl.head, *delNode = NULL;    // 1 分
while(pStud != NULL){                          // 1 分
    delNode = pStud;                            // 1 分
    pStud = pStud->next;                        // 1 分
    delete delNode;
}
}
```

参考答案及评分标准