《计算机程序设计》实验报告

姓名：宋亦寒 学号：PB25000200 实验日期：2025 年 11 月 24 日

实验名称	C 程序设计入门

一、 上机程序（有注释）

1. 实验 P126-7-删除字符

【问题描述】

编写函数实现删除字符串中所有指定字符的功能。

【编程要求】

(1) 字符串和要删除的字符都从键盘输入；

(2) 字符串的读入及字符删除操作都用函数实现；

(3) 编写主程序完成函数功能的测试；

(4) 函数原型参考：

```
void getString(char[]);
void deleteCharInString(char[], char);
```

```
#include <stdio.h>
#include <string.h>

void getString(char origin[])
{
    int i=0;
    while (1){
        char ch=getchar();          // 读取一个字符
        if (ch=='\n') break;        // 遇到换行结束输入
        origin[i]=ch;               // 保存字符
        i++;
    }
    origin[i]='\0';                 // 添加字符串结束符
}

void deleteCharInString(char a[], char ch)
{
    char b[100];
    int lengtha=strlen(a);          // 获取原字符串长度
    int j=0;
    for (int i=0; i<lengtha; i++){
        if (a[i] != ch){            // 保留非目标字符
            b[j]=a[i];
            j++;
        }
    }
```

```
    b[j]='\0';                          // 新字符串结束符
    strcpy(a,b);                        // 覆盖回原字符串
}

int main()
{
    char a[100];
    char ch;
    getString(a);                       // 输入字符串（读到换行）
    scanf("%c", &ch);                   // 输入要删除的字符
    deleteCharInString(a,ch);           // 删除字符
    printf("%s", a);                    // 输出结果
}
```

2. 实验 P150-10-图书信息管理

【问题描述】

编写函数，按出版时间由早到晚的顺序，对图书信息进行排序。在 main 函数中输入图书信息，调用该函数排序，并输出排序后的图书信息。

【编程要求】

(1) 日期结构体类型及别名定义如下：

```
typedef struct date {
    int year;
    int month;
} DATE;
```

(2) 图书信息结构体类型定义如下：

```
struct book {
    int  num;
    char title[20];
    DATE ptime;
};
```

(3) 设图书数量为 6，符号常量定义如下：

```
#define N 6
```

(4) 函数原型参考：

```
void sortBook(struct book[], int);
```

(5) 主函数中图书数组定义及函数调用参考：

```
struct book lib[N];
sortBook(lib, N);
```

```
#include <stdio.h>
#define N 6

typedef struct date {
```

```
    int year;
    int month;
} DATE;

struct book {
    int num;
    char title[20];
    DATE ptime;
};

void sortBook(struct book list[], int n);

int main()
{
    struct book lib[N];
    int i;
    for (i=0; i<N; i++){
        scanf("%d %s %d %d", &lib[i].num, lib[i].title,
              &lib[i].ptime.year, &lib[i].ptime.month);  // 读入信息
    }
    sortBook(lib, N);                                    // 按时间排序
    for (i=0; i<N; i++){
        printf("%d %s %d %d\n", lib[i].num, lib[i].title,
               lib[i].ptime.year, lib[i].ptime.month);  // 输出排序结果
    }
}

void sortBook(struct book list[], int n) {
    int i, j;
    struct book temp;

    for (i=0; i<n-1; i++){                               // 冒泡排序
        for (j=0; j<n-1-i; j++) {
            // 按年份排序，同年按月份排序
            if (list[j].ptime.year > list[j+1].ptime.year ||
               (list[j].ptime.year == list[j+1].ptime.year &&
                list[j].ptime.month > list[j+1].ptime.month)){
                temp = list[j];
                list[j] = list[j+1];
                list[j+1] = temp;                        // 交换
            }
        }
    }
}
```

3. 实验 P133-2-求解完数

【问题描述】

若一个正整数正好等于其因子之和（包括 1，但不包括其本身），则这个数称为完数。编程求解 1~*n*（*n* 为正整数）范围内的完数，输出完数及其因子。

```
#include <stdio.h>

int factorsNum=0;

//求解数的因子并判断是否为完数的函数原型声明
int isPerfectNumber (int num,int factorsList[]);
//输出完数及其因子列表(不包括完数自身)的函数原型声明
void print (int perfectNumber,int factorsList[]);

int main()
{
    int i,upperNumber,factorsList[100];
    scanf("%d",&upperNumber);
    for (i=1;i<=upperNumber;i++)
    {
        if (isPerfectNumber(i,factorsList))
        {
            print(i,factorsList);
        }
    }
    return 0;
}

/*
*功能:求解数的因子并判断是否为完数
*参数:num:待处理的数,factorsList:保存完数的因子
*返回值:若为完数,返回 1,否则返回 0
*/
int isPerfectNumber (int num,int factorsList[])
{
    int i=0,sum=0;
    factorsNum=0;
    for (i=1; i<num; i++)
    {
        if (num%i==0)
        {
            factorsList[factorsNum]=i;
            factorsNum++;
            sum+=i;
```

```
            }
        }

        if (num==sum)
        {
            return 1;
        }
        return 0;
    }

    /*
    *功能:输出完数及其因子列表(不包括完数自身)
    *参数:perfectNumber:完数,factorsList:保存完数因子
    *返回值:无,在函数体输出
    */
    void print (int perfectNumber,int factorsList[])
    {
        int i;
        printf("%d: ",perfectNumber);
        for (i=0;i<factorsNum;i++)
        {
            printf("%d ", factorsList[i]);
        }
        printf("\n");
    }
```

4. 多项式相乘

【问题描述】编写一个 C 语言程序，实现多个多项式的乘法运算。每个多项式通过输入系数数组来定义，数组元素从高次项到低次项排列。多项式个数不定，乘法运算需要单独写成函数。

```
    #include <stdio.h>

    int multiply(int a[], int m, int b[], int n)
    {
        int c[200];
        int a_reversed[100];
        int b_reversed[100];

        // 将多项式系数倒序（便于按最低次开始卷积）
        for (int i=0; i<m; i++)
```

```
        a_reversed[i] = a[m-1-i];
    for (int i=0; i<n; i++)
        b_reversed[i] = b[n-1-i];

    // 初始化结果数组
    for (int k=0; k<200; k++)
        c[k]=0;

    // 多项式卷积
    for (int s=0; s<m; s++){
        for (int t=0; t<n; t++){
            c[s+t]+=a_reversed[s]*b_reversed[t];
        }
    }

    int len=m+n-1;              // 结果项数

    // 倒序放回 a（高次在前）
    for (int i=0;i<len;i++){
        a[i]=c[len-1-i];
    }

    return len;
}

int main()
{
    int poly[100][100];         // 存储多行多项式
    int countline[100];         // 每行系数个数
    int i=0, j=0;

    int ch;
    int num=0;
    int in_number=0;            // 标记是否正在读一个数字
    int sgn=1;                  // 处理正负号
    int count_line=0;           // 当前行的系数个数

    // 逐字符读入多项式系数
    while (1) {
        ch=getchar();
        if (ch=='-'){
            sgn=-1;             // 读到负号
            in_number=1;
        } else if (ch>='0' &&ch<='9'){  // 多位数字
```

```
            num=num*10+(ch-'0');
            in_number=1;
        } else if (ch==' ' || ch=='\n' || ch==EOF){  // 数字结束
            if (in_number==1) {
                poly[i][j++]=sgn*num; // 存入一项
                count_line++;

                num=0;
                in_number=0;
                sgn=1;
            }

            // 行结束
            if (ch=='\n') {
                if (count_line==1 && poly[i][0]==0)   // “0”行结束输入
                    break;
                countline[i]=count_line;
                i++;
                j=0;
                count_line=0;
            }
        }
    }

    int result[200];
    int res_len=countline[0];    // 初始结果为第一行多项式

    for (int k=0; k<res_len; k++){
        result[k]=poly[0][k];
    }

    // 从第二行开始依次做乘法
    for (int k=1; k<i; k++){
        res_len=multiply(result, res_len, poly[k], countline[k]);
    }

    // 输出最后结果（格式化）
    for (int k=0; k<res_len; k++){
        int x0=result[k];
        int pow=res_len-1-k;
        if (x0==0)
            continue;
        if (k>0 && x0>0){
            printf(" + ");
```

```
            } else if (x0<0){
                if (k>0) printf(" - ");
                else printf("-");
                x0=-x0;
            }
            if (x0!=1 || pow==0) printf("%d", x0);
            if (pow>0){
                printf("x");
                if(pow>1) printf("^%d", pow);
            }
        }
    }
```

5.实验 P140-3-递归法求字符串长度
用递归和非递归的方法分别实现求字符串长度的函数 udf_strlen。

```
int udf_strlen(char str[])
{
    int i=0;
    while (str[i]!='\0'){
        i++;
    }
    return i;
}

int udf_strlen(char str[], int index)
{
    if (str[index]=='\0')
        return 0;
    return 1+udf_strlen(str, index+1);
}
```

二、 调试中的问题及解决方法（字数不限）

1.逐字符解析数字的状态管理（负号、多位数、连着的空格、EOF）容易出错 - 解决方案：设置变量（如 in_number、sgn）统计输入状态

2.：多项式系数顺序与卷积方向搞混（高次/低次混排导致错位） - 解决方案：明确并统一约定存放顺序（高次在前或后），在卷积前后做反转或使用索引映射，写注释标明约定并用小例验证。

3.输出格式化的边界情况（全部系数为 0、系数为 ±1、首项符号、项间空格）处理容易遗漏 - 解决方案：把输出逻辑分成明确分支（首项与非首项、系数绝对值为 1 且非常数项、系数为 0 跳过）