**《程序设计基础课程设计》报告**

班级：XX

成员：XX

所做题目：1，2，3，4，5

报告撰写日期 XX

**报告一**

**第1题**

高精度计算

**算法描述：**

将长整数作为字符串输入，然后使用函数将字符串转换为int数组，数组的第0位储存符号，第1位储存数字长度（用于减法）。计算时按照手算的方法进行计算。

**源程序：**

no1.c

/\*

\* 程序名：长整数计算器

\* 功能：实现长整数加减法，可操作正负数

\* 成员：XX

\*/

#define LENGTH 1000 //长整数位数

#define POSITIVE 0 //正号标记

#define NEGATIVE 1 //负号标记

#include <stdio.h>

#include <ctype.h>

//数组指针交换函数

void swap(int\*\* a, int\*\* b)

{

    int\* t = \*a;

    \*a = \*b;

    \*b = t;

}

//储存长整数函数

int saveLongInteger(int\* integerArray, char\* charArray)

{

    int j = 2; //长整数数组索引，数组前两位用于储存符号和位数，因此从2开始

    for (int i = LENGTH; i > 0; i--)

    {

        if (charArray[i] != 0)

        {

            if (isdigit(charArray[i])) //检查输入正确性

            {

                integerArray[j] = charArray[i] - '0';

                j++;

            }

            else

            {

                return -1;

            }

        }

    }

    switch (charArray[0]) //填充符号位

    {

    case '+':

        integerArray[0] = POSITIVE;

        break;

    case '-':

        integerArray[0] = NEGATIVE;

        break;

    default: //输入省略正号的情况

        if (isdigit(charArray[0]))

        {

            integerArray[j] = charArray[0] - '0';

            j++;

            integerArray[0] = POSITIVE;

        }

        else

        {

            return -1;

        }

        break;

    }

    integerArray[1] = j - 2;

    return 0;

}

//长整数加法函数

void addLongInteger(int\* result, int\* num\_1, int\* num\_2)

{

    int t = 0; //储存进位

    for (int i = 2; i < LENGTH; i++)

    {

        result[i] = num\_1[i] + num\_2[i] + t;

        t = result[i] / 10;

        result[i] %= 10;

    }

    result[0] = num\_1[0]; //默认两数同号，因此将其中一个数的符号作为结果的符号

}

//长整数减法函数

void subtractLongInteger(int\* result, int\* num\_1, int\* num\_2)

{

    result[0] = num\_1[0];

    //将 num\_1, num\_2 按绝对值由大到小排序

    if (num\_1[1] < num\_2[1]) //num\_1 的位数小于 num\_2 的情况

    {

        swap(&num\_1, &num\_2);

        result[0] = NEGATIVE;

    }

    else if (num\_1[1] == num\_2[1]) //位数相同，逐位比较大小

    {

        for (int i = LENGTH; i > 1; i--)

        {

            if (num\_1[i] > num\_2[i])

            {

                break;

            }

            else if (num\_1[i] < num\_2[i])

            {

                swap(&num\_1, &num\_2);

                result[0] = NEGATIVE;

                break;

            }

        }

    }

    if (num\_1[1] == num\_2[1] && num\_1[num\_1[1] + 1] < num\_2[num\_2[1] + 1]) //num\_1 与 num\_2 位数相同,但 num\_1 的首位比 num\_2 小的情况

    {

        swap(&num\_1, &num\_2);

        result[0] = NEGATIVE;

    }

    int t = 0; //储存借位

    for (int i = 2; i < LENGTH; i++)

    {

        result[i] = num\_1[i] - num\_2[i] - t;

        if (result[i] < 0)

        {

            result[i] += 10;

            t = 1;

        }

        else

        {

            t = 0;

        }

    }

}

int main()

{

    int longInteger\_1[LENGTH + 1] = { 0 }; //长整数数组，+1 防止数组越界

    int longInteger\_2[LENGTH + 1] = { 0 };

    char charLongInteger[LENGTH + 1] = { 0 }; //长整数字符串

    printf("输入数字1：\n>");

    scanf("%s", charLongInteger);

    if (saveLongInteger(longInteger\_1, charLongInteger) == -1) //将长整数字符串转换为数组，同时处理输入错误的情况

    {

        printf("输入错误\n");

        return -1;

    }

    for (int i = 0; i < LENGTH; i++) //重置长整数字符串，以便重新输入

    {

        charLongInteger[i] = 0;

    }

    printf("输入数字2：\n>");

    scanf("%s", charLongInteger);

    if (saveLongInteger(longInteger\_2, charLongInteger) == -1)

    {

        printf("输入错误\n");

        return -1;

    }

    int mode;

    printf("操作：\n1. 相加  2. 相减\n>");

    scanf("%d", &mode);

    int result[LENGTH + 1] = { 0 };

    switch (mode)

    {

    case 1: //加法

        if (longInteger\_1[0] == longInteger\_2[0]) //判断两数符号是否相同

        {

            addLongInteger(result, longInteger\_1, longInteger\_2);

        }

        else //符号不同时将加法转换为减法

        {

            if (longInteger\_2[0] == NEGATIVE) //numz-2 为负时相当于 num\_1 - |num\_2|

            {

                longInteger\_2[0] = POSITIVE;

                subtractLongInteger(result, longInteger\_1, longInteger\_2);

            }

            else //num\_1 为负时相当于 num\_2 - |num\_1|

            {

                longInteger\_1[0] = POSITIVE;

                subtractLongInteger(result, longInteger\_2, longInteger\_1);

            }

        }

        printf("它们的和是：");

        break;

    case 2: //减法

        if (longInteger\_1[0] == longInteger\_2[0]) //同号情况

        {

            subtractLongInteger(result, longInteger\_1, longInteger\_2);

        }

        else

        {

            if (longInteger\_2[0] == NEGATIVE) //num\_2 为负，相当于 num\_1 + |num\_2|

            {

                longInteger\_2[0] = POSITIVE;

                addLongInteger(result, longInteger\_1, longInteger\_2);

            }

            else //num\_1 为负，相当于 -(|num\_1| + num\_2)

            {

                longInteger\_2[0] = NEGATIVE;

                addLongInteger(result, longInteger\_2, longInteger\_1);

            }

        }

        printf("它们的差是：");

        break;

    default:

        printf("输入错误\n");

        return -1;

    }

    if (result[0] == NEGATIVE) //如果是负数先输出'-'号

    {

        printf("-");

    }

    int flag = 0; //用于判断结果的首位是否为零

    for (int i = LENGTH; i > 1; i--)

    {

        if (result[i] == 0 && flag == 0) //首位为零则不输出数字，跳到下一次循环

        {

            continue;

        }

        printf("%d", result[i]);

        flag = 1;

    }

    if (flag == 0)

    {

        printf("0"); //若未输出任何数字则答案为零

    }

    printf("\n");

    return 0;

}

**测试数据（输入、输出）：**

111 - 123 = -12

1666666666666666666666666666666 + (-5555555555555555555) = 1666666666661111111111111111111

500000000000000 – 900020000000000 = -400020000000000

**总结：**

编程时要尽量将各种可能情况考虑全面

**报告二**

**第2题**

简单数据结构-堆栈模拟

**算法描述：**

使用链表模拟栈，节点中储存值val和指向下一个节点的指针next。入栈时使用malloc函数创建新节点，将输入的值储存到新节点的val变量，将之前的链表头储存到next指针变量。然后将新节点的指针作为新的链表头。出栈时输出节点头的val值，然后将next中储存的指针作为新的链表头，然后使用free函数释放旧链表头占用的空间。

**源程序：**

no2.c

/\*

\* 程序名：堆栈模拟

\* 功能：利用链表模拟堆栈

\* 成员：XX

\*/

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

//用于模拟栈的链表，val 存放元素的值，next为下一元素的地址

typetef struct Zhan {

    int val;

    struct Zhan\* next;

}ZHAN;

//入栈函数

ZHAN\* push(ZHAN\* head, int val) {

    ZHAN\* newNode = (ZHAN\*)malloc(sizeof(ZHAN));

    if (newNode == NULL) return NULL;

    newNode->val = val;

    newNode->next = head;

    return newNode;

}

//出栈函数

ZHAN\* pop(ZHAN\* head) {

    ZHAN\* nextNode = head->next;

    free(head);

    return nextNode;

}

//展示栈中所有元素

void show(ZHAN\* head) {

    if (head == NULL) {

        printf("栈为空\n");

    }

    else

    {

        printf("栈中元素：\n[");

        while (head != NULL) {

            printf("%d, ", head->val);

            head = head->next;

        }

        printf("\b\b]\n");

    }

}

int main()

{

    ZHAN\* head = NULL; //存放链表头

    char cmd[10] = { 0 }; //存放要处理的命令

    do

    {

        printf("入栈输入\"push + 数值\"，出栈输入\"pop\"，退出输入\"q\"\n>");

        scanf("%s", cmd);

        if (strcmp(cmd, "push") == 0) {

            int val;

            scanf("%d", &val);

            head = push(head, val);

            if (head != NULL)

            {

                printf("成功入栈：\n%d\n", val);

            }

            else

            {

                printf("入栈失败");

            }

            show(head);

        }

        else if (strcmp(cmd, "pop") == 0)

        {

            if (head != NULL)

            {

                int val = head->val;

                head = pop(head);

                printf("出栈元素：%d\n", val);

                show(head);

            }

            else

            {

                printf("栈为空\n");

            }

        }

        else if (cmd[0] != 'q')

        {

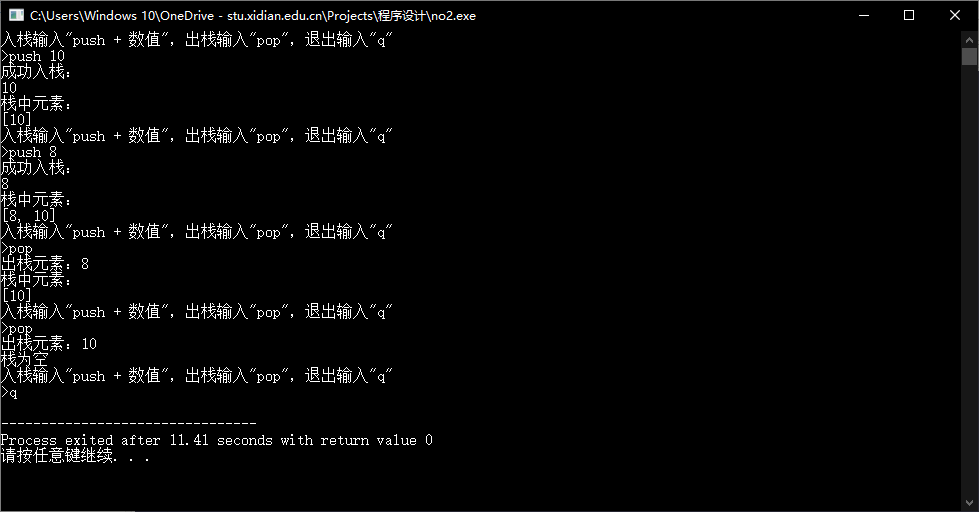
            printf("命令错误\n");

        }

    } while (cmd[0] != 'q');

}

**测试数据（输入、输出）：**



**总结：**

申请内存空间时要注意内存分配失败的情况

**报告三**

**第3题**

位图图像文件缩放

**算法描述**

近邻取样，将缩放后的像素坐标映射到原图，取最近的像素点作为参考点，直接插入目标文件。

**源程序：**

BmpFormat.h //bmp文件结构体

# ifndef BMP\_H

# define BMP\_H

#pragma pack(1) //手动更改内存对齐的值，避免读取bmp文件时产生数据错位

// 文件信息头结构体

typedef struct tagBITMAPFILEHEADER

{

    unsigned short bfType;        // 19778，必须是BM字符串，对应的十六进制为0x4d42,十进制为19778，否则不是bmp格式文件

    unsigned int   bfSize;        // 文件大小 以字节为单位(2-5字节)

    unsigned short bfReserved1;   // 保留，必须设置为0 (6-7字节)

    unsigned short bfReserved2;   // 保留，必须设置为0 (8-9字节)

    unsigned int   bfOffBits;     // 从文件头到像素数据的偏移  (10-13字节)

} BITMAPFILEHEADER;

//图像信息头结构体

typedef struct tagBITMAPINFOHEADER

{

    unsigned int    biSize;          // 图像信息头的大小 (14-17字节)

    long            biWidth;         // 图像的宽  (18-21字节)

    long            biHeight;        // 图像的高  (22-25字节)

    unsigned short  biPlanes;        // 图片的平面属  (26-27字节)

    unsigned short  biBitCount;      // 一像素所占的位数  (28-29字节)

    unsigned int    biCompression;   // 图象数据压缩的类型，0为不压缩。 (30-33字节)

    unsigned int    biSizeImage;     // 图像数据所占大小, 应该等于上面文件头结构中bfSize-bfOffBits (34-37字节)

    long            biXPelsPerMeter; // 水平分辨率，单位像素/米。一般为0 (38-41字节)

    long            biYPelsPerMeter; // 垂直分辨率，单位像素/米。一般为0 (42-45字节)

    unsigned int    biClrUsed;       // 位图实际使用的彩色表中的颜色索引数（设为0的话，则说明使用所有调色板项）。 (46-49字节)

    unsigned int    biClrImportant;  // 对图象显示有重要影响的颜色索引数，如果是0，表示都重要。(50-53字节)

} BITMAPINFOHEADER;

//本来是调色板，但24位位图没有调色板，为了方便我用它储存像素点的RGB数据

typedef struct \_PixelInfo

{

    unsigned char rgbBlue;     //该颜色的蓝色分量  (值范围为0-255)

    unsigned char rgbGreen;    //该颜色的绿色分量  (值范围为0-255)

    unsigned char rgbRed;      //该颜色的红色分量  (值范围为0-255)

    //unsigned char rgbReserved; // 保留，必须为0

} PixelInfo;

#pragma pack()

#endif

zoom.c

/\*

\* 程序名：位图缩放

\* 功能：缩放24位位图

\* 成员：XX

\*/

#include <stdio.h>

#include <stdlib.h>

#include "BmpFormat.h"

BITMAPFILEHEADER src\_fh;

BITMAPINFOHEADER src\_ih;

BITMAPFILEHEADER dst\_fh;

BITMAPINFOHEADER dst\_ih;

FILE\* src;

FILE\* dst;

int main(int argc, char\*\* argv)

{

    //打开文件

    if (argc < 3)

    {

        printf("参数错误！\n");

        return -1;

    }

    src = fopen(argv[1], "rb");

    if (src == NULL)

    {

        printf("打开\"%s\"失败！\n", argv[1]);

        return -1;

    }

    //获取源bmp文件信息

    fread(&src\_fh, sizeof(BITMAPFILEHEADER), 1, src);

    fread(&src\_ih, sizeof(BITMAPINFOHEADER), 1, src);

    if (src\_fh.bfType == 0x4d42) //检验打开的文件是不是bmp文件

    {

        double zoom = atoi(argv[2]) / 100.0; //将缩放参数字符串转换为double格式

        //创建目标bmp文件信息

        dst\_fh = src\_fh;

        dst\_ih = src\_ih;

        //修改目标bmp文件图像大小等信息并储存

        dst\_ih.biWidth \*= zoom;

        dst\_ih.biHeight \*= zoom;

        long srcRowSize = ((24 \* src\_ih.biWidth + 31) / 32) \* 4; //每行的字节数；因为有自动补零，行字节数不一定等于像素实际大小，所以需要单独计算

        long dstRowSize = ((24 \* dst\_ih.biWidth + 31) / 32) \* 4;

        long srcSkipSize = srcRowSize % 3; //每行需要跳过的补零字节数

        long dstSkipSize = dstRowSize % 3;

        long srcRowPixSize = srcRowSize - srcSkipSize; //每行图像数据真正占用的字节数（去除补零）

        long dstRowPixSize = dstRowSize - dstSkipSize;

        dst\_ih.biSizeImage = dstRowSize \* dst\_ih.biHeight;

        dst\_fh.bfSize = dst\_ih.biSizeImage + dst\_fh.bfOffBits;

        //将源dmp文件图像数据储存到src\_pix数组

        long srcPixSize = src\_ih.biHeight \* srcRowPixSize; //图像数据占用的总字节数（去除补零）

        long dstPixSize = dst\_ih.biHeight \* dstRowPixSize;

        PixelInfo\* src\_pix = (PixelInfo\*)malloc(srcPixSize);

        PixelInfo\* dst\_pix = (PixelInfo\*)malloc(dstPixSize);

        if (src\_pix == NULL || dst\_pix == NULL)

        {

            printf("内存申请失败（考虑降低放大倍数）\n");

            return -1;

        }

        long i = 0;

        for (long row = 0; row < src\_ih.biHeight; row++)

        {

            fread(&src\_pix[i], sizeof(PixelInfo), src\_ih.biWidth, src); //每次将一行的数据读入src\_pix

            fseek(src, srcSkipSize, SEEK\_CUR); //跳过补零数据

            i += src\_ih.biWidth;

        }

        //将缩放后对应的图像数据储存到dst\_pix数组

        for (long row = 0; row < dst\_ih.biHeight; row++)

        {

            for (long col = 0; col < dst\_ih.biWidth; col++)

            {

                dst\_pix[row \* dst\_ih.biWidth + col] = src\_pix[(long)(row / zoom) \* src\_ih.biWidth + (long)(col / zoom)];

            }

        }

        //将数据存入目标文件

        dst = fopen(argv[3], "wb");

        fwrite(&dst\_fh, sizeof(BITMAPFILEHEADER), 1, dst);

        fwrite(&dst\_ih, sizeof(BITMAPINFOHEADER), 1, dst);

        char\* buffer = (char\*)calloc(1, dstSkipSize);

        i = 0;

        for (int row = 0; row < dst\_ih.biHeight; row++)

        {

            fwrite(&dst\_pix[i], dstRowPixSize, 1, dst);

            fwrite(buffer, 1, dstSkipSize, dst);

            i += dst\_ih.biWidth;

        }

        //结束程序前的清理

        free(src\_pix);

        free(dst\_pix);

        fclose(src);

        fclose(dst);

        printf("缩放成功\n");

    }

    else

    {

        printf("\"%s\"不是有效的bmp文件\n", argv[1]);

        return -1;

    }

    return 0;

}

**测试数据（输入、输出）：**

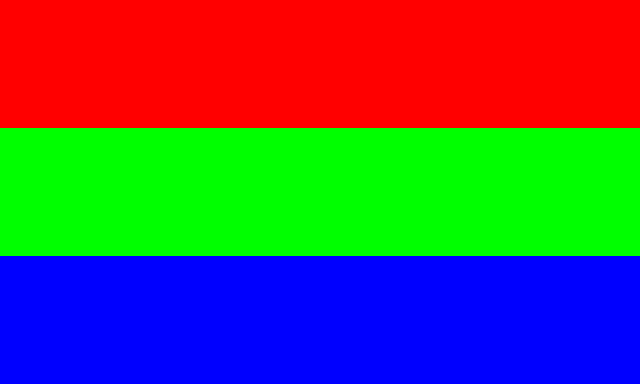
zoom 1.bmp 800 2.bmp

zoom Arthur.bmp 10 smallArthur.bmp

1.bmp



2.bmp



Arthur.bmp



smallArthur.bmp



**总结：**

在对结构体、类等使用sizeof时一定要注意内存对齐问题。

在读取二进制文件时要注意补零的问题。

当读取到奇怪的数据时很可能是数组越界。

对指定二进制文件进行操作时应检查文件类型。

数据先保存到内存，全部处理完后再一起存入硬盘的运行速度会快很多。

**报告四**

**第4题**

RLE压缩解压算法

**算法描述：**

注：压缩文件格式：[长度指示字节][数据]

如果有3个及以上的连续字节，就将这些重复字节压缩为一个，放置在数据位。然后将长度指示字节首位设置为1，表示其后跟的一个字节为重复字节。低位设置为重复次数。如果是不重复的字节，就将长度指示字节首位设置为0，低位设置为不重复数据长度n。其后n个字节都是这些不重复的数据。

**源程序：**

rle.c

/\*

\* 程序名：RLE算法文件压缩

\* 成员：XX

\*/

#define BUFFERSIZE 512\*1024 //最大缓存大小，在解压模式中建议大于源文件大小

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

typedef unsigned char BYTE;

FILE\* src;

FILE\* dst;

int   fileSize;      //文件大小

int readFromFile(BYTE\* srcBuffer, int bufferSize)

{

    static int readed = 0; //已读取数据大小

    int remaining = fileSize - readed;

    int toBuffer = remaining > bufferSize ? bufferSize : remaining; //确定一次读入缓存的数据量。若剩余数据大小大于缓存容量，则读入缓存容量大小的数据。否则读入全部剩余数据。

    fread(srcBuffer, 1, toBuffer, src);

    readed += toBuffer;

    return toBuffer;

}

//将[begin, end)的数据存到dst

void storeToBuffer(BYTE\* dstBuffer, BYTE index, BYTE\* begin, BYTE\* end)

{

    \*dstBuffer = index;

    dstBuffer++;

    memcpy(dstBuffer, begin, end - begin);

}

//压缩操作

int compress(BYTE\* srcBuffer, int bufferSize, BYTE\* dstBuffer)

{

    BYTE\* end = srcBuffer + bufferSize;

    if (fileSize < 3) //如果数据长度<3则不用判断，直接原样存入目标缓存区。

    {

        storeToBuffer(dstBuffer, (BYTE)bufferSize, srcBuffer, end);

        return fileSize + 1; //多了一个长度指示字节

    }

    int toFile = 0; //返回值

    BYTE\* bufferPointer = dstBuffer;

    BYTE index; //用于指示压缩类型和长度的变量，和压缩后的数据一起写入目标文件。其中最高位用于表示压缩类型，后七位表示长度。

    BYTE\* cur = srcBuffer + 2; //从原始数据的第三个字节开始判断是否与前两个字节相同

    BYTE\* mark = srcBuffer; //用于标记某次判断的起始位置

    while (cur < end)

    {

        if (\*cur == \*(cur - 1) && \*cur == \*(cur - 2)) //如果当前数据与前两个数据相同，则继续向后检索直到不同。然后压缩。

        {

            //先将前面相同的数据存入缓存

            if (cur - 2 - mark > 0)

            {

                index = cur - 2 - mark;

                storeToBuffer(bufferPointer, index, mark, cur - 2);

                toFile += index + 1;

                bufferPointer += index + 1;

            }

            mark = cur - 2;

            while (\*cur == \*mark && cur - mark < 127 && cur < end && bufferPointer - dstBuffer + cur - mark < BUFFERSIZE) //数据相同 && 数据长度 < 127 && cur未到达srcBuffer末尾 && dstBuffer存得下

            {

                cur++;

            }

            index = (cur - mark) | 0b10000000; //将最高位变为1，表示后面的字节是连续字节。

            storeToBuffer(bufferPointer, index, mark, mark + 1);

            toFile += 2; //一旦遇到3个及以上的连续数据，都会被压缩成两个字节。因此存入文件的数据量+2。

            bufferPointer += 2;

            mark = cur;

            cur++;

        }

        else if (!(cur - mark < 127 && cur < end && bufferPointer - dstBuffer + cur - mark < BUFFERSIZE)) //超出范围，将数据存入缓冲，重新开始下一轮检索。

        {

            index = cur - mark;

            storeToBuffer(bufferPointer, index, mark, cur);

            toFile += index + 1;

            bufferPointer += cur - mark + 1;

            index = 1;

            mark = cur;

            cur++;

        }

        cur++;

    }

    //到最后可能出现cur >= end但还剩几个字节没有处理的情况，此时应将剩余字节存入dstBuffer。

    if (mark < end)

    {

        index = end - mark;

        storeToBuffer(bufferPointer, index, mark, end);

        toFile += index + 1;

    }

    return toFile;

}

//解压操作

int decompress(BYTE\* srcBuffer, int bufferSize, BYTE\* dstBuffer)

{

    int toFile = 0;

    BYTE\* end = dstBuffer + BUFFERSIZE;

    BYTE\* bufferPointer = dstBuffer;

    BYTE\* cur = srcBuffer;

    BYTE index;

    while (cur < srcBuffer + bufferSize)

    {

        index = \*cur;

        BYTE flag = index & 0b10000000;

        cur++;

        //flag的首位为1说明后面是重复的数据

        if (flag == 0b10000000)

        {

            index &= 0b01111111; //将指示变量转换成重复次数

            //如果存得下就memcpy到dstBuffer

            if (bufferPointer + index < end)

            {

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

                {

                    memcpy(bufferPointer, cur, 1);

                    bufferPointer++;

                }

            }

            //存不下就先将dstBuffer的数据存入文件，然后再存。

            else

            {

                fwrite(dstBuffer, 1, bufferPointer - dstBuffer, dst);

                bufferPointer = dstBuffer;

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

                {

                    memcpy(bufferPointer, cur, 1);

                    bufferPointer++;

                }

            }

            toFile += index;

        }

        else

        {

            if (bufferPointer + index < end)

            {

                memcpy(bufferPointer, cur, index);

                bufferPointer += index;

            }

            else

            {

                fwrite(dstBuffer, 1, bufferPointer - dstBuffer, dst);

                memcpy(dstBuffer, cur, index);

                bufferPointer = dstBuffer + index;

            }

            cur += index - 1;

            toFile += index;

        }

        cur++;

    }

    return toFile;

}

int main(int argc, char\*\* argv)

{

    //打开文件

    if (argc == 4)

    {

        src = fopen(argv[1], "rb");

        dst = fopen(argv[3], "wb");

        if (src == NULL)

        {

            printf("File open failur!\n");

            return -1;

        }

    }

    else

    {

        printf("I need some more parameters!\n");

        return -1;

    }

    //获取文件大小，以便确定缓存大小

    fseek(src, 0L, SEEK\_END);

    fileSize = ftell(src);

    rewind(src);

    //给缓存分配内存

    BYTE\* srcBuffer = malloc(BUFFERSIZE);

    BYTE\* dstBuffer = malloc(BUFFERSIZE);

    if (strcmp(argv[2], "-c") == 0) //压缩模式

    {

        while (ftell(src) != fileSize)

        {

            int toBuffer = readFromFile(srcBuffer, BUFFERSIZE > fileSize ? fileSize : BUFFERSIZE);

            int toFile = compress(srcBuffer, toBuffer, dstBuffer);

            fwrite(dstBuffer, 1, toFile, dst);

        }

        printf("File \"%s\" has been compressed to \"%s\"!\n", argv[1], argv[3]);

    }

    else if (strcmp(argv[2], "-d") == 0)

    {

        while (ftell(src) != fileSize)

        {

            int toBuffer = readFromFile(srcBuffer, BUFFERSIZE > fileSize ? fileSize : BUFFERSIZE);

            int toFile = decompress(srcBuffer, toBuffer, dstBuffer);

            fwrite(dstBuffer, 1, toFile, dst);

        }

        printf("File \"%s\" has been decompressed to \"%s\"!\n", argv[1], argv[3]);

    }

    else

    {

        printf("Wrong command! Check it again!\n");

        return -1;

    }

    //释放缓存内存

    free(srcBuffer);

    free(dstBuffer);

    return 0;

}

**测试数据（输入、输出）：**

compress src.txt compressed.txt

decompress compressed.txt decompressed.txt

src.txt:

aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaskjdfhjklsdfhfdjskghsdhkljgbbbbbbbbbbbbbbbsldkfjlskfjflksdfjfsldjfcccsdlfakkkkfslkdfgjddddddddddddddddsfd

compressed.txt:

skjdfhjklsdfhfdjskghsdhkljg廱sldkfjlskfjflksdfjfsldjf僣sdlfa刱fslkdfgj恉sfd

decompressed.txt:

aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaskjdfhjklsdfhfdjskghsdhkljgbbbbbbbbbbbbbbbsldkfjlskfjflksdfjfsldjfcccsdlfakkkkfslkdfgjddddddddddddddddsfd

**总结：**

只有当文件指针已经到文件尾，并且再次要求读入一个字符时，feof() 函数才会返回1。在使用 feof() 判断是否到文件尾时要注意这一点。

尽量不要在多处更改全局变量的值，因为这样容易导致全局变量的值难以确定。

**报告五**

**第5题**

简单文件数据库-模拟图书馆管理系统

**算法描述：**

使用结构体存放图书、读者信息，使用两个双向链表分别存放图书和用户结构体以满足任意增加、删除图书信息和用户信息的需求。

**源程序：**

no5.h //图书、用户信息结构体，链表，相关函数

#pragma once

#define MAX 10 //每本书的最大储量

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <ctype.h>

typedef struct BOOK {

    int ID;

    char name[10];

    int inventory;

    int borrowedBy[MAX];

}Book;

typedef struct USER {

    int ID;

    char name[10];

    int borrowed;

}User;

typedef struct BOOKLIST {

    Book book;

    struct BOOKLIST\* previous;

    struct BOOKLIST\* next;

}BookList;

typedef struct USERLIST {

    User user;

    struct USERLIST\* previous;

    struct USERLIST\* next;

}UserList;

//显示用户菜单

void showUserMenu()

{

    printf(".               User               .\n");

    printf(". 1-> Borrow book                  .\n");

    printf(". 2-> Return book                  .\n");

    printf(". 3-> Search book                  .\n");

    printf(". 4-> Search the borrowed book     .\n");

    printf(". 5-> Change user                  .\n");

    printf(". 6-> Quit                         .\n");

    printf(". Please select a number           .\n>");

}

//显示管理员菜单

void showAdminMenu()

{

    printf(".            Administor            .\n");

    printf(". 1-> Add book information         .\n");

    printf(". 2-> Modify book information      .\n");

    printf(". 3-> Delete book                  .\n");

    printf(". 4-> Add User information         .\n");

    printf(". 5-> Modify User information      .\n");

    printf(". 6-> Delete  User                 .\n");

    printf(". 7-> Change user                  .\n");

    printf(". 8-> Quit                         .\n");

    printf(". Please select a number           .\n>");

}

//查询书籍，可使用ID或书名查找

BookList\* searchBook(const char\* str, BookList\* list)

{

    BookList\* result = NULL;

    if (isalpha(str[0])) //通过书名查找

    {

        const char\* name = str;

        while (list != NULL)

        {

            if (strcmp(name, list->book.name) == 0)

            {

                result = list;

                break;

            }

            list = list->next;

        }

    }

    else //通过ID查找

    {

        int ID = atoi(str);

        while (list != NULL)

        {

            if (list->book.ID == ID)

            {

                result = list;

                break;

            }

            list = list->next;

        }

    }

    return result;

}

//查找用户，可使用ID或用户名查找

UserList\* searchUser(const char\* str, UserList\* list)

{

    UserList\* result = NULL;

    if (isalpha(str[0])) //通过用户名查找

    {

        const char\* name = str;

        while (list != NULL)

        {

            if (strcmp(name, list->user.name) == 0)

            {

                result = list;

                break;

            }

            list = list->next;

        }

    }

    else //通过ID查找

    {

        int ID = atoi(str);

        while (list != NULL)

        {

            if (list->user.ID == ID)

            {

                result = list;

                break;

            }

            list = list->next;

        }

    }

    return result;

}

//管理员函数

//增加书目

BookList\* addBook(int ID, const char\* name, int inventory, int borrowedBy[], BookList\* head)

{

    BookList\* bookNode = (BookList\*)calloc(1, sizeof(BookList));

    if (bookNode == NULL)

    {

        return NULL;

    }

    bookNode->next = head;

    bookNode->previous = NULL;

    bookNode->book.ID = ID;

    bookNode->book.inventory = inventory;

    strcpy(bookNode->book.name, name);

    if (borrowedBy != NULL)

    {

        for (int i = 0; i < MAX; i++)

        {

            bookNode->book.borrowedBy[i] = borrowedBy[i];

        }

    }

    if (head != NULL)

    {

        head->previous = bookNode;

    }

    return bookNode;

}

//增加用户

UserList\* addUser(int ID, const char\* name, int borrowed, UserList\* head)

{

    UserList\* userNode = (UserList\*)calloc(1, sizeof(UserList));

    if (userNode == NULL)

    {

        return NULL;

    }

    userNode->next = head;

    userNode->previous = NULL;

    userNode->user.ID = ID;

    userNode->user.borrowed = borrowed;

    strcpy(userNode->user.name, name);

    if (head != NULL)

    {

        head->previous = userNode;

    }

    return userNode;

}

//删除书目

void deleteBook(BookList\* target)

{

    BookList\* previous = target->previous;

    BookList\* next = target->next;

    if (previous != NULL)

    {

        previous->next = next;

    }

    if (next != NULL)

    {

        next->previous = previous;

    }

    free(target);

}

//删除用户

void deleteUser(UserList\* target)

{

    UserList\* previous = target->previous;

    UserList\* next = target->next;

    if (previous != NULL)

    {

        previous->next = next;

    }

    if (next != NULL)

    {

        next->previous = previous;

    }

    free(target);

}

//修改书籍信息（不应该随意更改书本ID，因为ID是书本的唯一标识符，只能更改书名）

void editBook(const char\* name, int inventory, Book\* target)

{

    strcpy(target->name, name);

}

//修改用户信息

void editUser(const char\* name, User\* target)

{

    strcpy(target->name, name);

}

//用户函数

//借书

int borrowBook(Book\* book, User\* user)

{

    if (book->inventory == 0) //该书籍已借完

    {

        return -1;

    }

    book->inventory -= 1;

    user->borrowed = book->ID;

    for (int i = 0; i < MAX; i++)

    {

        if (book->borrowedBy[i] == 0)

        {

            book->borrowedBy[i] = user->ID;

            return 0;

        }

    }

}

//还书

const char\* returnBook(BookList\* bookList, User\* user)

{

    BookList\* target = searchBook(user->borrowed, bookList);

    target->book.inventory += 1;

    user->borrowed = 0;

    for (int i = 0; i < MAX; i++)

    {

        if (target->book.borrowedBy[i] == user->ID)

        {

            target->book.borrowedBy[i] = 0;

            break;

        }

    }

    return target->book.name;

}

//存储图书管理信息文件

void saveArchive(BookList\* bookList, UserList\* userList)

{

    FILE\* archive = fopen("archive.txt", "w");

    BookList\* bhead = bookList;

    int count = 0;

    while (bhead != NULL)

    {

        count++;

        bhead = bhead->next;

    }

    fprintf(archive, "----------Book----------\n[%d books]\n", count);

    while (bookList != NULL)

    {

        fprintf(archive, "------------------\nID: %d\nName: %s\nInventory: %d\n", bookList->book.ID, bookList->book.name, bookList->book.inventory);

        fprintf(archive, "Borrowed by:\n");

        for (int i = 0; i < MAX; i++)

        {

            if (bookList->book.borrowedBy[i] != 0)

            {

                fprintf(archive, "%d\n", bookList->book.borrowedBy[i]);

            }

        }

        fprintf(archive, "%d\n", 0); //结束标记

        fprintf(archive, "------------------\n");

        bookList = bookList->next;

    }

    UserList\* uhead = userList;

    count = 0;

    while (uhead != NULL)

    {

        count++;

        uhead = uhead->next;

    }

    fprintf(archive, "----------User----------\n[%d users]\n", count);

    while (userList != NULL)

    {

        fprintf(archive, "------------------\nID: %d\nName: %s\nBorrowed: %d\n", userList->user.ID, userList->user.name, userList->user.borrowed);

        fprintf(archive, "------------------\n");

        userList = userList->next;

    }

    fclose(archive);

}

//读取图书管理信息文件

void loadArchive(BookList\*\* bookList, UserList\*\* userList)

{

    printf("Loading archive from file...\n");

    FILE\* archive = fopen("archive.txt", "r");

    if (archive == NULL)

    {

        printf("File not found.\n");

        return;

    }

    int     count;

    int     ID;

    int     inventory;

    int     borrowed;

    int     borrowedBy[MAX] = { 0 };

    char    name[10] = { 0 };

    char    buffer[30] = { 0 };

    if (fscanf(archive, "%s", buffer) == EOF)

    {

        printf("Empty file.\n");

        return;

    }

    printf("Books...\n");

    if (strcmp(buffer, "----------Book----------") != 0)

    {

        printf("Format error!\n");

        return;

    }

    fscanf(archive, "%\*c%\*c%d%\*s", &count);

    for (int i = 0; i < count; i++)

    {

        fscanf(archive, "%s", buffer);

        if (strcmp(buffer, "------------------") != 0)

        {

            printf("Format error!\n");

            return;

        }

        fscanf(archive, "%\*s%d%\*s%s%\*s%d", &ID, name, &inventory);

        fscanf(archive, "%\*s%\*s"); //读掉"Borrowed by:"

        for (int i = 0; i < MAX; i++)

        {

            fscanf(archive, "%d", &borrowedBy[i]);

            if (borrowedBy[i] == 0)

            {

                break;

            }

        }

        fscanf(archive, "%s", buffer);

        if (strcmp(buffer, "------------------") != 0)

        {

            printf("Format error!\n");

            return;

        }

        \*bookList = addBook(ID, name, inventory, borrowedBy, \*bookList);

        //显示读取到的信息

        printf("ID: %d\nName: %s\nInventory: %d\n", ID, name, inventory);

        printf("Borrowed by:\n");

        int flag = 0;

        for (int i = 0; i < MAX; i++)

        {

            if (borrowedBy[i] != 0)

            {

                printf("%d\n", borrowedBy[i]);

                borrowedBy[i] = 0;

                flag = 1;

            }

        }

        if (flag == 0)

        {

            printf("null\n");

        }

    }

    printf("Users...\n");

    fscanf(archive, "%s", buffer);

    if (strcmp(buffer, "----------User----------") != 0)

    {

        printf("Format error!\n");

        return;

    }

    fscanf(archive, "%\*c%\*c%d%\*s", &count);

    for (int i = 0; i < count; i++)

    {

        fscanf(archive, "%s", buffer);

        if (strcmp(buffer, "------------------") != 0)

        {

            printf("Format error!\n");

            return;

        }

        fscanf(archive, "%\*s%d%\*s%s%\*s%d", &ID, name, &borrowed);

        fscanf(archive, "%s", buffer);

        if (strcmp(buffer, "------------------") != 0)

        {

            printf("Format error!\n");

            return;

        }

        \*userList = addUser(ID, name, borrowed, \*userList);

        //显示读取到的信息

        printf("ID: %d\nName: %s\nBorrowed: %d\n", ID, name, borrowed);

    }

    printf("All of the information has been loaded.\n");

    fclose(archive);

}

no5.c

/\*

\* 程序名：模拟图书管理系统

\* 成员：XX

\*/

#include "Libsim.h"

#include <string.h>

#include <stdio.h>

#include <stdlib.h>

#include <ctype.h>

int main(int argc, char\*\* argv)

{

    //图书、用户链表头

    BookList\* bookList = NULL;

    UserList\* userList = NULL;

    BookList\* targetBook;

    UserList\* targetUser;

    int  mode;

    int  ID;

    char name[10];

    char buffer[10];

    int  inventory;

    //获取命令

    if (argc < 2)

    {

        printf("Using the command line parameters to open!");

        return -1;

    }

    char userType[3] = { 0 };

    char userName[10] = { 0 };

    strcpy(userType, argv[1]);

    if (argc >= 3)

    {

        strcpy(userName, argv[2]);

    }

    //读取图书信息文件

    loadArchive(&bookList, &userList);

    while (1)

    {

        if (strcmp(userType, "-a") == 0) //管理员模式

        {

            printf("User: administor\n");

            showAdminMenu();

            scanf("%d", &mode);

            switch (mode)

            {

            case 1: //增加书目

                //输入并检查

                printf("--------Add book information-------\n");

                printf("Book ID:\n>");

                scanf("%d", &ID);

                if (ID <= 0)

                {

                    printf("Input error!\n");

                    break;

                }

                \_itoa(ID, buffer, 10);

                if (searchBook(buffer, bookList) != NULL)

                {

                    printf("The book is already existing\n");

                    break;

                }

                printf("Book name:\n>");

                scanf("%s", name);

                if (searchBook(name, bookList) != NULL)

                {

                    printf("The book is already existing\n");

                    break;

                }

                printf("Book inventory:\n>");

                scanf("%d", &inventory);

                if (inventory < 0)

                {

                    printf("Input error!\n");

                    break;

                }

                //执行操作

                bookList = addBook(ID, name, inventory, NULL, bookList);

                if (bookList == NULL)

                {

                    printf("Error: Memory allocation failure!\nBook not added.\n");

                    break;

                }

                printf("Book \"%s\" has been added\n", name);

                break;

            case 2: //修改书籍信息

                //输入并检查

                printf("------Modify book information------\n");

                printf("Book ID\\name:\n>");

                scanf("%s", buffer);

                targetBook = searchBook(buffer, bookList);

                if (targetBook == NULL)

                {

                    printf("No such book\n");

                    break;

                }

                printf("New name:\n>");

                scanf("%s", name);

                if (searchBook(name, bookList) != NULL)

                {

                    printf("The book is already existing\n");

                    break;

                }

                printf("New book inventory:\n>");

                scanf("%d", &inventory);

                if (inventory < 0)

                {

                    printf("Input error!\n");

                    break;

                }

                //执行操作

                editBook(name, inventory, &(targetBook->book));

                printf("Book information modified successfully\n");

                break;

            case 3: //删除书目

                //输入并检查

                printf("------------Delete book------------\n");

                printf("Book ID\\name\n>");

                scanf("%s", buffer);

                targetBook = searchBook(buffer, bookList);

                if (targetBook == NULL)

                {

                    printf("No such book\n");

                    break;

                }

                //执行操作

                strcpy(name, targetBook->book.name);

                deleteBook(targetBook);

                printf("Book \"%s\" has been deleted\n", name);

                break;

            case 4: //新增用户

                //输入并检查

                printf("--------Add User information-------\n");

                printf("User ID:\n>");

                scanf("%d", &ID);

                if (ID <= 0)

                {

                    printf("Input error!\n");

                    break;

                }

                \_itoa(ID, buffer, 10);

                if (searchUser(buffer, userList) != NULL)

                {

                    printf("The user is already existing\n");

                    break;

                }

                printf("User name:\n>");

                scanf("%s", name);

                if (searchUser(name, userList) != NULL)

                {

                    printf("The user is already existing\n");

                    break;

                }

                //执行操作

                userList = addUser(ID, name, 0, userList);

                if (userList == NULL)

                {

                    printf("Error! Memory allocation failure!\n");

                    return -1;

                }

                printf("User \"%s\" has been added\n", name);

                break;

            case 5: //修改用户信息

                //输入并检查

                printf("------Modify User information------\n");

                printf("User ID\\name:\n");

                scanf("%s", buffer);

                targetUser = searchUser(buffer, userList);

                if (targetUser == NULL)

                {

                    printf("No such user\n");

                    break;

                }

                printf("New name:\n>");

                scanf("%s", name);

                if (searchUser(name, userList) != NULL)

                {

                    printf("The user is already existing\n");

                    break;

                }

                //执行操作

                editUser(name, &(targetUser->user));

                printf("User information modified successfully\n");

                break;

            case 6: //删除用户信息

                //输入并检查

                printf("------------Delete user------------\n");

                printf("User ID\\name\n>");

                scanf("%s", buffer);

                targetUser = searchUser(buffer, userList);

                if (targetUser == NULL)

                {

                    printf("No such user\n");

                    break;

                }

                //执行操作

                strcpy(name, targetUser->user.name);

                deleteUser(targetUser);

                printf("User \"%s\" has been deleted\n", name);

                break;

            case 7: //更改当前用户

                printf("------------Change user------------\n");

                printf("Log in: <-userType userName>\n>");

                scanf("%s", userType);

                if (strcmp(userType, "-u") == 0)

                {

                    scanf("%s", userName);

                }

                break;

            case 8: //退出程序

                return 0;

            default:

                printf("Wrong number!\n");

                break;

            }

        }

        else if (strcmp(userType, "-u") == 0) //用户模式

        {

            if (argc < 2)

            {

                printf("User name:\n>");

                scanf("%s", userName);

                break;

            }

            targetUser = searchUser(userName, userList); //确认用户信息

            if (targetUser == NULL)

            {

                printf("No such user\n");

                printf("------------Change user------------\n");

                printf("Log in: <-userType userName>\n>");

                if (strcmp(userType, "-u") == 0)

                {

                    scanf("%s", userName);

                }

                continue;

            }

            printf("User: %s\n", userName);

            showUserMenu();

            scanf("%d", &mode);

            switch (mode)

            {

            case 1: //借书

                printf("------------Borrow book------------\n");

                if (targetUser->user.borrowed != 0) //如果用户还有未还书籍，则此时不能借书

                {

                    printf("You have already borrowed a book, return it before borrowing a new one.\n");

                    break;

                }

                //输入并检查

                printf("Book ID\\name:\n>");

                scanf("%s", buffer);

                targetBook = searchBook(buffer, bookList);

                if (targetBook == NULL)

                {

                    printf("No such book\n");

                    break;

                }

                //执行操作

                strcpy(name, targetBook->book.name);

                if (borrowBook(&(targetBook->book), &(targetUser->user)) == -1)

                {

                    printf("Book \"%s\" is borrowed out.\nTry again later.\n", name);

                    break;

                }

                printf("You have borrowed \"%s\"\n", name);

                break;

            case 2: //还书

                printf("------------Return book------------\n");

                if (targetUser->user.borrowed == 0) //若用户没有未还书籍，则此时不用还书

                {

                    printf("You have not borrowed any book.\n");

                    break;

                }

                strcpy(name, returnBook(bookList, &(targetUser->user)));

                printf("You have returned \"%s\"\n", name);

                break;

            case 3: //查找书籍信息

                printf("------------Search book------------\n");

                printf("Book ID\\name:\n>");

                scanf("%s", buffer);

                targetBook = searchBook(buffer, bookList);

                if (targetBook == NULL)

                {

                    printf("No such book\n");

                    break;

                }

                printf("-------------Book info-------------\n");

                printf("ID:\t\t%d\n", targetBook->book.ID);

                printf("Name:\t\t%s\n", targetBook->book.name);

                printf("Inventory:\t%d\n", targetBook->book.inventory);

                printf("Borrowed by:\n");

                int flag = 0;

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

                {

                    if (targetBook->book.borrowedBy[i] != 0)

                    {

                        itoa(targetBook->book.borrowedBy[i], buffer, 10);

                        targetUser = searchUser(buffer, userList);

                        printf("%s\n", targetUser->user.name);

                        flag = 1;

                    }

                }

                if (flag == 0)

                {

                    printf("null\n");

                }

                printf("----------------end----------------\n");

                break;

            case 4: //查询已借书籍

                printf("------Search the borrowed book-----\n");

                if (targetUser->user.borrowed == 0)

                {

                    printf("You have not borrow any book\n");

                    break;

                }

                \_itoa(targetUser->user.borrowed, buffer, 10);

                targetBook = searchBook(buffer, bookList);

                printf("You have borrowed \"%s\".\n", targetBook->book.name);

                break;

            case 5: //更改当前用户

                printf("------------Change user------------\n");

                printf("Log in: <-userType userName>\n>");

                scanf("%s", userType);

                if (strcmp(userType, "-u") == 0)

                {

                    scanf("%s", userName);

                }

                break;

            case 6: //退出程序

                return 0;

            default:

                printf("You entered wrong number\n");

                break;

            }

        }

        else

        {

            printf("Wrong parameter!\n");

            return -1;

        }

        saveArchive(bookList, userList);

        printf("\n");

    }

}

**测试数据（输入、输出）：**



**总结：**

在操作指针时要注意指针的合法性，避免野指针、空指针访问。

将具有唯一性元素存放到的数据结构中时应检查是否存在相同元素。

在管理具有数目属性的元素时要注意不能让数目小于0 。