**《程序设计基础课程设计》实验报告**

班级：09 姓名： 学号：

**第1题 BigInt**

**算法描述：**

高精度算法：

把数字拆开，拆成一位一位的（不一定）放进数组里，用一个数组表示一个数字，来进行运算。

运算真正加减判断：

对于两个数字x、y，每有一个数字是负数，便反转一次用户输入的运算。

即，若用户输入减，且x、y中有一个负数，则运算自动转为加法。

减法交换函数：

对于两个数字x、y，若其进行减法，则判断|x|、|y|的大小，若对于x-y，但|x|<|y|,则交换运算顺序，使程序做y-x，结果乘以-1

判断结果正负函数：

1、如果结果是0，则直接输出0，不需要再判断正负

2、如果实际上运行的是加法，则第一个数字为负数，最后结果也为负数，否则结果为正数

3、如果实际上进行的是减法：

若进行过被减数和减数交换，则若原来被减的数是正数，结果是负数，若原来被减的数是负数，结果是正数

若没有进行过被减数和减数的交换，则若原来被减的数是负数，结果是负数，若原来被减的数是正数，结果是正数

加法：

从最后一位开始，按位数相加，有进位则后面进行加法时+1，当一个数字加完（位数短）后，判断是否有进位后，再直接复制剩余的数字

减法：

用t做借位数，从最后一位开始，按位数相减，用t储存减法结果，往vector里面放入（t + 10） % 10，之后再进行减法时，对被减数 + t，即再令t = 被减数-减数+t，循环到最后即可

**源程序：**

#include <iostream>

#include <vector>

using namespace std;

// 判断实际上进行加法/减法的函数

bool AddSubjudge(string a, string b, int judge)

{

//如果输入2（进行减法），则变成-1，之后若输入的两个数字之中，每有一个负数，judge \* -1，最后若结果是 1 ，则为加法，否则减法

if(judge == 2)

judge = -1;

if(a[0] == '-')

judge \*= -1;

if(b[0] == '-')

judge \*= -1;

if(judge == 1)

return true;

return false;

}

// 进行减法运算时，判断两数字大小，决定是否交换做差顺序的函数

bool cmp(vector<int> &A, vector<int> &B)

{

//如果两个数字长度不一样，让长的做被减数

//若长度相同，则从最高位开始比较，比较大的做被减数

if (A.size() != B.size())

return A.size() > B.size();

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

if(A[i] != B[i])

return A[i] > B[i];

return true;

}

//判断运算结果正负的函数

void PlusMinusJudge(string a, string b, bool judge1, bool judge2, int if0)

{

//如果最后运算结果是0，则直接返回不需要再判断正负

if(if0 == 0)

return;

//如果judge1为true（进行加法），且第一个数字为负数，则结果必为负数，否则为正数

if(judge1 && a[0] == '-')

printf("-");

//如果judge1为false（进行减法）

//当judge2为false（被减数和减数进行交换），则原来的被减数是正数，结果是负数，原来被减数是负数，结果是正数

//当judge2为true（被减数和减数未进行交换）， 则原来的被减数是负数，结果是负数，原来被减数是正数，结果是正数

else if(!judge1)

{

if(a[0] == '-' && judge2)

{

printf("-");

return;

}

else if(isdigit(a[0]) && !judge2)

{

printf("-");

return;

}

}

}

//此处是实现高精度加法函数

vector<int> AddBigData(vector<int> &A, vector<int> &B)

{

//若B比较长，交换后相加

if(A.size() < B.size())

return AddBigData(B, A);

//定义t，作为中间介质，即借位数

vector<int> C;

int t = 0;

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

{

t += A[i];

if(i < B.size())

t += B[i];

C.push\_back(t % 10);

t /= 10;

}

//最后t若不等于0，则再放入C中

if(t)

C.push\_back(t);

return C;

}

//此处是实现高精度减法函数

vector<int> MinusBigData(vector<int> &A, vector<int> &B)

{

//定义t，做中间介质，即借位数

vector<int> C;

for(int i = 0, t = 0; i < A.size(); i ++ )

{

t = A[i] - t;

if(i < B.size())

t -= B[i];

C.push\_back((t + 10) % 10);

if(t < 0)

t = 1;

else

t = 0;

}

//此处将C中的0全部弹出

while(C.size() > 1 && C.back() == 0)

C.pop\_back();

return C;

}

//此处是将string转换为vector<int>的函数

vector<int> TypeConversion(string num)

{

vector<int> Num;

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

if(isdigit(num[i])) //除掉负号‘-’

Num.push\_back(num[i] - '0');

return Num;

}

int main()

{

//judge来判断用户表面上要进行加法/减法

int judge;

cout << "输入 1 进行加法\n输入 2 进行减法" << endl;

cin >> judge;

//num1，num2 为将要进行运算的两个数字的字符串形式

string num1, num2;

cout << "请依次输入进行计算的数字，数字间用空格分隔" << endl;

cin >> num1 >> num2;

//此处使用TypeConversion函数，将num1和num2转化成vector<int>形式，方便运算

vector<int> A = TypeConversion(num1);

vector<int> B = TypeConversion(num2);

//ans来接收答案

vector<int> ans;

//此处，AddSubjudge函数来判断，实际上的运算需要通过加法还是减法进行，其结果储存在ASjudge中

bool ASjudge = AddSubjudge(num1, num2, judge);

//此处，将进行减法时，判断两个数字大小的函数结果储存在BSjudge中，方便后续对结果正负判断函数 PlusMinusJudge 的使用

bool BSjudge = cmp(A, B);

//实际需要加法，调用AddBigData函数

if(ASjudge)

ans = AddBigData(A, B);

else

//实际需要减法，根据给定的两个数字大小，选择MinusBigData函数参数的传入

if(BSjudge)

ans = MinusBigData(A, B);

else

ans = MinusBigData(B, A);

//此处判断结果的正负，并输出负号

PlusMinusJudge(num1, num2, ASjudge, BSjudge, ans.back());

//输出结果

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

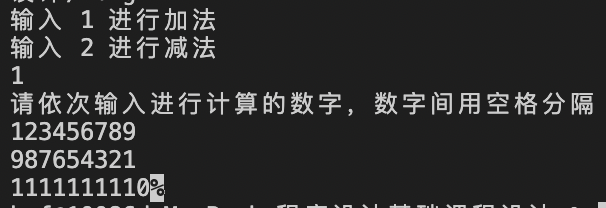
cout << ans[i];

return 0;

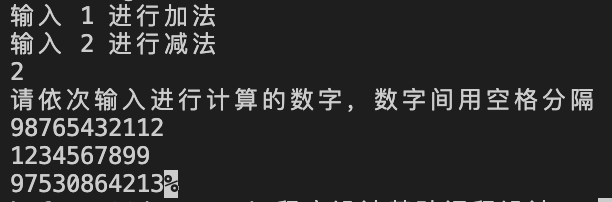
}

**测试数据（输入、输出）：（数据不用太多）**

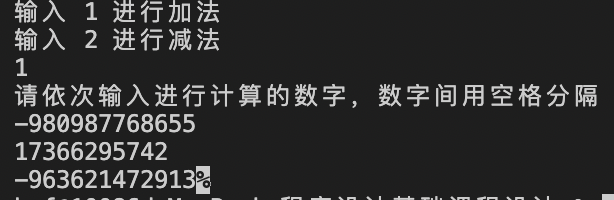
普通加：



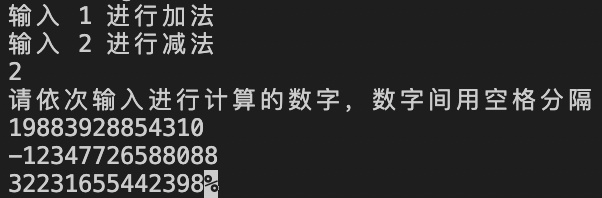
普通减：



带负号的加：



带负号的减：



**第2题 Stack**

**算法描述：**

**push**

**新建节点, 将该节点的值设为输入值, 指针指向当前栈顶元素. 将栈顶指针指向新建的节点.**

**代码中的 push() 即push的具体实现**

**top**

**直接调用栈顶指针 head , 返回其所指的元素的值. 代码中的 top() 即top的具体实现**

**pop**

**栈顶元素的 nxt 指针指向的是倒数第二个元素. 所以将栈顶指针指向该元素, 并将栈顶删除. 代码中的 pop() 即pop的具体实现**

**isFull**

**判断栈是否是满的，是则返回1，否则返回0**

**IsEmpty**

**判断栈是否为空，是则返回1，否则返回0**

**源程序：**

#include <iostream>

#include <cstdio>

using namespace std;

const int N = 100010;

int stack[N];

int idx, m;

// 栈的push操作

void push(int x)

{

// 将下一个元素放入数组中,当达到满栈时再放入会进行报错

if(idx < N)

stack[idx ++ ] = x;

else

printf("error");

}

//栈的pop操作

void pop()

{

// 在idx下标非零时，使下表减1，从而完成pop操作

if(idx)

idx -- ;

else

printf("error\n");

}

//获取栈顶元素

int top()

{

//返回下标为 idx-1 的元素，即为栈顶元素

return stack[idx - 1];

}

//判断是否满栈

int isFull()

{

//当下标 idx == N时，说明满栈

if(idx == N)

return 1;

return 0;

}

//判断是否空栈

int isEmpty()

{

//当下标 idx == 0时，说明空栈

if(idx == 0)

return 1;

return 0;

}

int main()

{

int n, cnt;

while(1)

{

//输出指令，供用户输入

printf("1.push\n2.pop\n3.top\n4.isFull\n5.isEmpty\nEnter -1 to stop the program\n");

scanf("%d", &n);

//设置退出循环的条件：n == -1

if(n == -1)

break;

switch(n)

{

//push时，先输入要放入栈的数字个数，再进行放入

case 1:

printf("please enter the number of numbers you want to push\n");

scanf("%d", &cnt);

printf("please enter the number one by one\n");

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

{

scanf("%d", &m);

push(m);

}

break;

//pop时，先输入要pop的数字个数，再弹出

case 2:

printf("please enter the number of numbers you want to pop\n");

scanf("%d", &cnt);

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

pop();

break;

//输出top元素

case 3:

printf("%d\n", top());

break;

//判断是否满栈

case 4:

if(isFull())

printf("the stack is full\n");

else

printf("the stack is not full\n");

break;

//判断是否空栈

case 5:

if(!isEmpty())

printf("the stack is not empty\n");

break;

//当输入指令在 1，2，3，4，5之外时，让用户重新输入

default:

printf("please enter the correct instruction\n");

break;

}

//每次操作后输出栈中所有元素

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

printf("%d ", stack[i]);

//当空栈时进行提醒/报错

if(idx == 0)

printf("the stack is empty");

cout << endl;

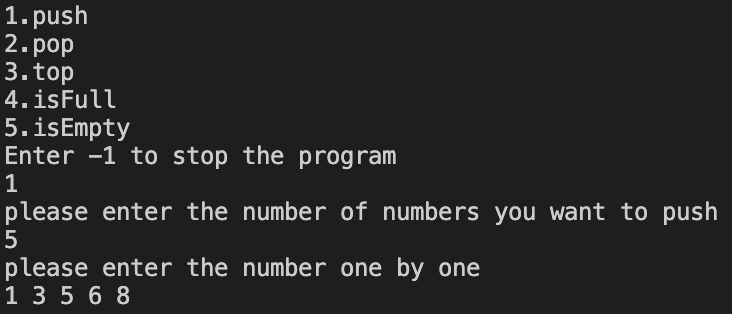
}

return 0;

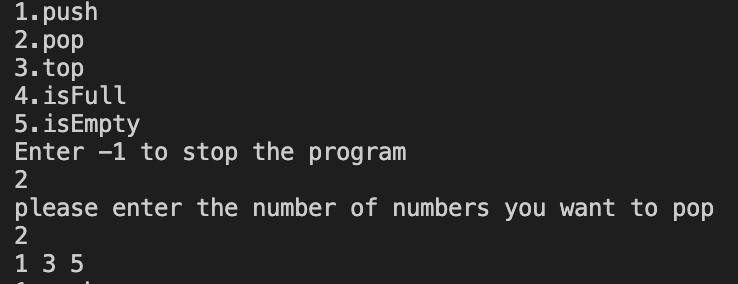
}

**测试数据（输入、输出）：（数据不用太多）**

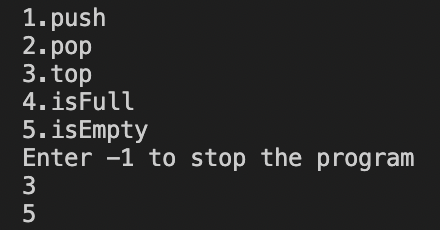
1、push

****

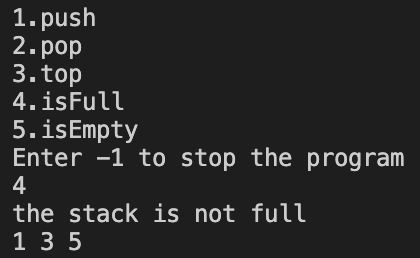
2、pop



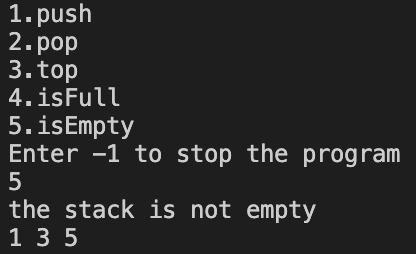
3、top



4、isFull



5、isEmpty



**第3题 zoom**

**算法描述：**

**位图文件由三部分组成:文件头 + 位图信息 + 位图像素数据 1、位图文件头。位图文件头主要用于识别位图文件。以下是位图文件头结构的定义:**

**typedef struct tagBITMAPFILEHEADER { // bmfh**

**WORD bfType;**

**DWORD bfSize;**

**WORD bfReserved1;**

**WORD bfReserved2;**

**DWORD bfOffBits;**

**} BITMAPFILEHEADER;**

**其中的bfType值应该是“BM”(0x4d42)，标志该文件是位图文件。bfSize的值是位图文件的大小。**

**2、位图信息中所记录的值用于分配内存，设置调色板信息，读取像素值等。 以下是位图信息结构的定 义:**

**typedef struct tagBITMAPINFO {**

**BITMAPINFOHEADER bmiHeader;**

**RGBQUAD bmiColors[1];**

**} BITMAPINFO;**

**可见位图信息也是由两部分组成的:位图信息头 + 颜色表 2.1位图信息头。位图信息头包含了单个像素所用字节数以及描述颜色的格式，此外还包括位图的宽度、**

**高度、目标设备的位平面数、图像的压缩格式。以下是位图信息头结构的定义:**

**typedef struct tagBITMAPINFOHEADER{ // bmih**

**DWORD biSize;**

**LONG biWidth;**

**LONG biHeight;**

**WORD biPlanes;**

**WORD biBitCount**

**DWORD biCompression;**

**DWORD biSizeImage;**

**LONG biXPelsPerMeter;**

**LONG biYPelsPerMeter;**

**DWORD biClrUsed;**

**DWORD biClrImportant;**

**} BITMAPINFOHEADER;**

**以下是对结构成员的说明：**

**biSize：结构BITMAPINFOHEADER的字节数，即sizeof(BITMAPINFOHEADER)\***

**biWidth：以像素为单位的图像宽度\***

**biHeight：以像素为单位的图像长度\***

**biplanes：目标设备的位平面数**

**biBitCount：每个像素的位数\*(1)**

**biCompression：图像的压缩格式(这个值几乎总是为0)**

**biSizeImage：以字节为单位的图像数据的大小(对BI\_RGB压缩方式而言)**

**biXPelsPermeter：水平方向上的每米的像素个数**

**biYpelsPerMeter：垂直方向上的每米的像素个数**

**biClrused：调色板中实际使用的颜色数(2)**

**biClrImportant：现实位图时必须的颜色数(3)**

**说明:\*是需要加以注意的部分，因为它们是我们在进行位图操作时经常参考的变量 (1)对于每个像素的字节数，分别有一下意义:**

**0，用在JPEG格式中**

**1，单色图，调色板中含有两种颜色，也就是我们通常说的黑白图片**

**4，16色图**

**8，256色图，通常说的灰度图 16，64K图，一般没有调色板，图像数据中每两个字节表示一个像素，5个或6个位表示一个RGB分 量 24，16M真彩色图，一般没有调色板，图像数据中每3个字节表示一个像素，每个字节表示一个 RGB分量**

**32，4G真彩色，一般没有调色板，每4个字节表示一个像素，相对24位真彩图而言，加入了一个透 明度，即RGBA模式**

**2)这个值通常为0，表示使用biBitCount确定的全部颜色，例外是使用的颜色树木小于制定的颜色深 度的颜色数目的最大值。**

**(3)这个值通常为0，表示所有的颜色都是必需的**

**2.2颜色表。颜色表一般是针对16位一下的图像而设置的，对于16位和16位以上的图像，由于其位图像 素数据中直接对对应像素的RGB(A)颜色进行描述，因而省却了调色板。而对于16位一下的图像，由于其 位图像素数据中记录的只是调色板索引值，因而需要根据这个索引到调色板去取得相应的RGB(A)颜色。 颜色表的作用就是创建调色板。**

**颜色表是由颜色表项组成的，颜色表项结构的定义如下:**

**typedef struct tagRGBQUAD { // rgbq**

**BYTE rgbBlue;**

**BYTE rgbGreen;**

**BYTE rgbRed;**

**BYTE rgbReserved;**

**} RGBQUAD;**

**其中需要注意的问题是，RGBQUAD结构中的颜色顺序是BGR，而不是平常的RGB。**

**3、位图数据。最后，在位图文件头、位图信息头、位图颜色表之后，便是位图的主体部分:位图数据。 根据不同的位图，位图数据所占据的字节数也是不同的，比如，对于8位位图，每个字节代表了一个像 素，对于16位位图，每两个字节代表了一个像素，对于24位位图，每三个字节代表了一个像素，对于32 位位图，每四个字节代表了一个像素。**

**由于比 8 位更小的位深一个像素不到一个字节, 处理需完全展开字节, 较为麻烦, 练习只处理 8 位以上的 位图.**

**头文件 windows.h 中定义了与位图有关的结构体, 直接拿来用即可.**

**放缩时，使用邻近插值法**

**由缩放比例r和缩放后的坐标(x, y), 可以得到原图中对应坐标(x/r, y/r). 其中, / 和c中的一致, 当x(y) > 0, r > 0时取下整**

**用原图中对应坐标像素填充缩放后的坐标, 这种简单的缩放方法成为临近插值法.**

**具体实现也很简单. 需要注意的是, 不同位深的位图存储一个像素的数据大小是不同的, 对于 bitCount 位**

**深, 每 bitCount/8 个字节代表一个像素. 在赋值的过程中需注意. 利用 memcopy 函数可以方便进行复制. 代码中 scale() 即为缩放的具体实现.**

**源程序：**

#include <windows.h>

#include <cstdio>

#include <cstring>

#include <cmath>

using namespace std;

class BitMap {

private:

BITMAPFILEHEADER fileHeader;

BITMAPINFOHEADER infoHeader;

LPRGBQUAD colorTable;

int width, height, bitCount;

int getLineSize();

int getDataSize();

int getHeaderSize();

int getColorTableSize();

void setFileHeader();

void setInfoHeader();

bool writeFile(FILE \*f);

public:

unsigned char \*dataBuf;

BitMap(){}

BitMap(int width, int height, int bitCount);

bool load(const char \*fileName);

bool save(const char \*fileName);

BitMap scale(const double rate);

int getBitCount();

};

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/ 构造函数

/ para width: 图片宽度

/ para height: 图片高度

/ para bitCount: 图片位深

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

BitMap::BitMap(int width, int height, int bitCount):

width(width), height(height), bitCount(bitCount) {

colorTable = NULL;

int dataSize = getDataSize();

dataBuf = (unsigned char\*)malloc(dataSize); // 分配数据空间

memset(dataBuf, 0, sizeof(dataSize));

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/ 计算一行的大小

/ return: 当前属性对应的一行的数值大小

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int BitMap::getLineSize() {

return (width \* bitCount / 8 + 3) / 4 \* 4;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/ 计算整个图像的大小

/ return: 当前属性对应的图像真实大小

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int BitMap::getDataSize() {

return sizeof(unsigned char) \* bitCount / 8 \* height \* getLineSize();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/ 计算位图文件的 header 大小

/ return: 当前属性对应的 header 大小

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int BitMap::getHeaderSize() {

return sizeof(BITMAPFILEHEADER) + sizeof(BITMAPINFOHEADER) + getColorTableSize();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/ 计算位图文件的调色板大小

/ return: 调色板大小

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int BitMap::getColorTableSize() {

if (colorTable == NULL)

return 0;

else

return sizeof(RGBQUAD) \* 256;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/ 设置 BITMAPFILEHEADER 的属性

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void BitMap::setFileHeader() {

fileHeader.bfType = 0x4D42;

fileHeader.bfSize = getHeaderSize() + getDataSize();

fileHeader.bfOffBits = getHeaderSize();

fileHeader.bfReserved1 = 0;

fileHeader.bfReserved2 = 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/ 设置 BITMAPINFOHEADER 的属性

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void BitMap::setInfoHeader() {

infoHeader.biWidth = width;

infoHeader.biHeight = height;

infoHeader.biBitCount = bitCount;

infoHeader.biSizeImage = getDataSize();

infoHeader.biSize = 40;

infoHeader.biPlanes = 1;

infoHeader.biCompression = 0;

infoHeader.biClrImportant = 0;

infoHeader.biXPelsPerMeter = 0;

infoHeader.biYPelsPerMeter = 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/ 读取位图

/ para fileName: 位图文件

/ return: 是否读取成功, 文件不存在读取失败

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

bool BitMap::load(const char \*fileName) {

FILE \*f = fopen(fileName, "rb");

if (!f)

return false;

fread(&fileHeader, sizeof(BITMAPFILEHEADER), 1, f); // 读取 BITMAPFILEHEADER

fread(&infoHeader, sizeof(BITMAPINFOHEADER), 1, f); // 读取 BITMAPINFOHEADER

width = infoHeader.biWidth;

height = infoHeader.biHeight;

bitCount = infoHeader.biBitCount;

// 位深小于等于8的位图还需读入调色板

if (bitCount <= 8) {

colorTable = new RGBQUAD[256];

fread(colorTable, 1, getColorTableSize(), f);

}

else

colorTable = NULL;

// 读取位图数据

dataBuf = new unsigned char[infoHeader.biSizeImage];

fread(dataBuf, 1, infoHeader.biSizeImage, f);

fclose(f);

return true;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/ 将位图所有数据写入文件

/ para f: 需要写如的目标文件

/ return: 是否写入成功, 文件不存在写入失败

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

bool BitMap::writeFile(FILE \*f) {

if (!f)

return false;

// 将header写入位图文件

fwrite(&fileHeader, sizeof(BITMAPFILEHEADER), 1, f);

fwrite(&infoHeader, sizeof(BITMAPINFOHEADER), 1, f);

// 将调色板写入位图文件

if (colorTable != NULL)

fwrite(colorTable, 1, getColorTableSize(), f);

// 将数据写入位图文件

fwrite(dataBuf, 1, infoHeader.biSizeImage, f);

return true;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/ 保存位图

/ para fileName: 位图文件

/ return: 是否保存成功

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

bool BitMap::save(const char \*fileName) {

FILE\* f = fopen(fileName, "wb"); // 创建或打开文件

if (!f)

return false;

// 设置Header属性

setFileHeader();

setInfoHeader();

// 将数据写入文件

if (!writeFile(f))

return false;

fclose(f);

return true;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/ 缩放位图

/ para ratio: 缩放比例

/ return: 缩放后的位图

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

BitMap BitMap::scale(const double ratio) {

// 新建一个缩放后的位图, 数据为空

BitMap bitmap(round(width \* ratio), round(height \* ratio), bitCount);

if (colorTable != NULL) {

bitmap.colorTable = new RGBQUAD[256];

memcpy(bitmap.colorTable, colorTable, getColorTableSize());

}

else

bitmap.colorTable = NULL;

int lineSize = bitmap.getLineSize(), originLineSize = getLineSize();

int l = bitCount / 8;

for(int i = 0; i < bitmap.height; i++) {

for(int j = 0; j < bitmap.width; j++) {

int x = j / ratio, y = i / ratio; // 原图像对应的点

if (x >= 0 && x < width && y >= 0 && y < height){

unsigned char \*dst = bitmap.dataBuf + (lineSize \* i + j \* l);

unsigned char \*src = dataBuf + (originLineSize \* y + x \* l);

memcpy(dst, src, l);

}

}

}

return bitmap;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/ 获取位深

/ return: 位图位深

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int BitMap::getBitCount() {

return bitCount;

}

int main(int argc, char const \*argv[]) {

const char \*inputFile = argv[1];

const char \*outputFile = argv[3];

double ratio = atoi(argv[2]) / 100.0;

BitMap input, output;

if (!input.load(inputFile)) {

printf("ERROR: File \"%s\" Not Exists\n", inputFile);

exit(EXIT\_FAILURE);

}

// 由于比 8 位更小的位深一个像素不到一个字节, 处理需完全展开字节, 较为麻烦, 所以不处理了 ^\_^

if (input.getBitCount() < 8) {

printf("ERROR: Unsupported File Format\n");

exit(EXIT\_FAILURE);

}

output = input.scale(ratio);

if (!output.save(outputFile)) {

printf("ERROR: Can't Save To File \"%s\"\n", outputFile);

exit(EXIT\_FAILURE);

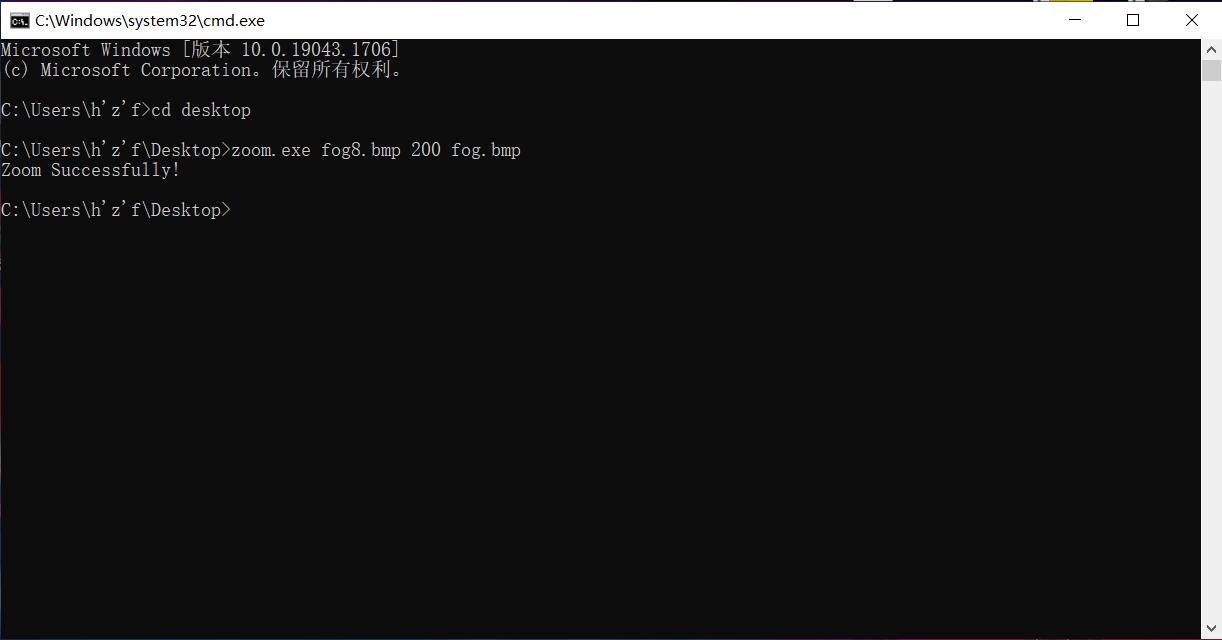
}

puts("Zoom Successfully!");

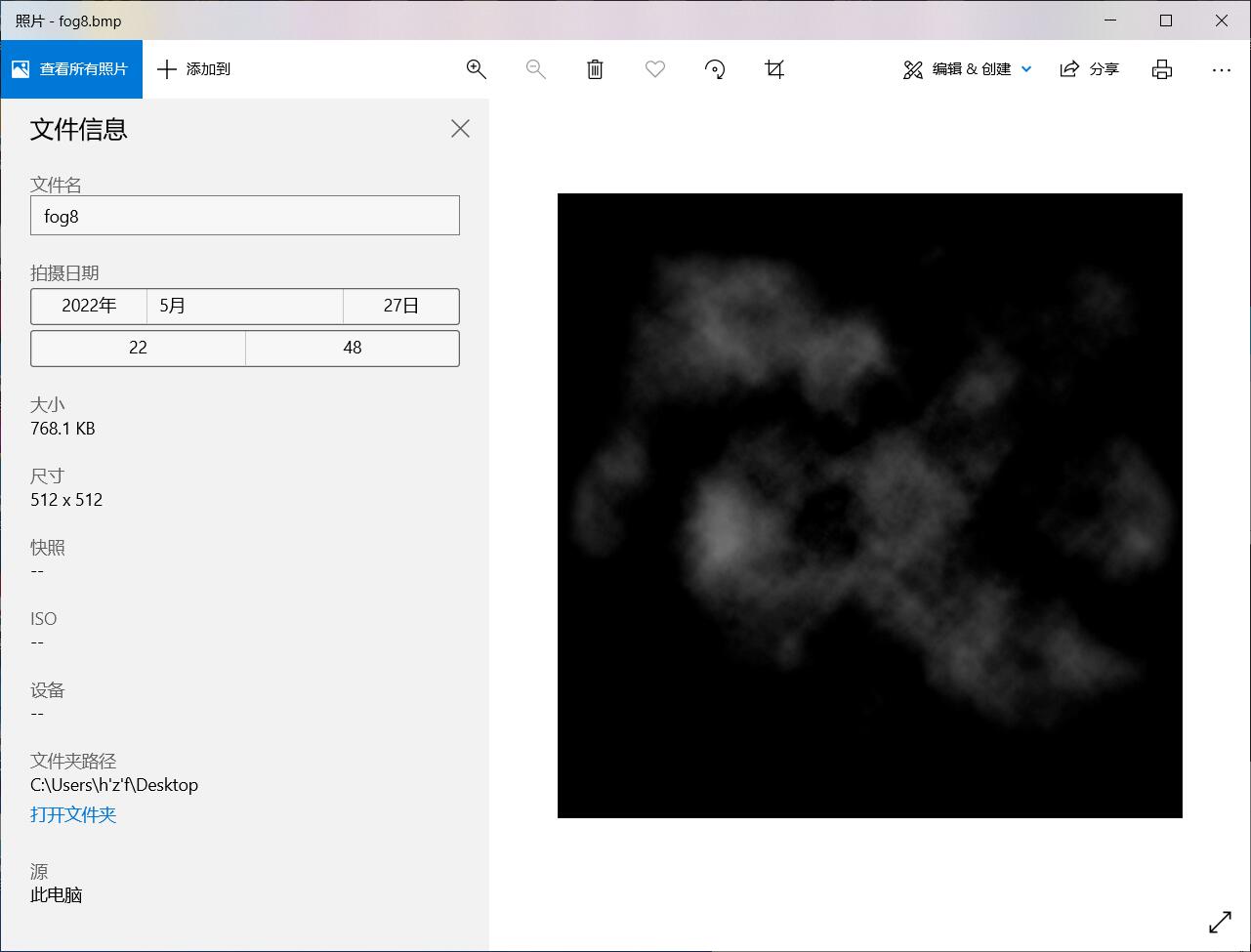
return 0;

}

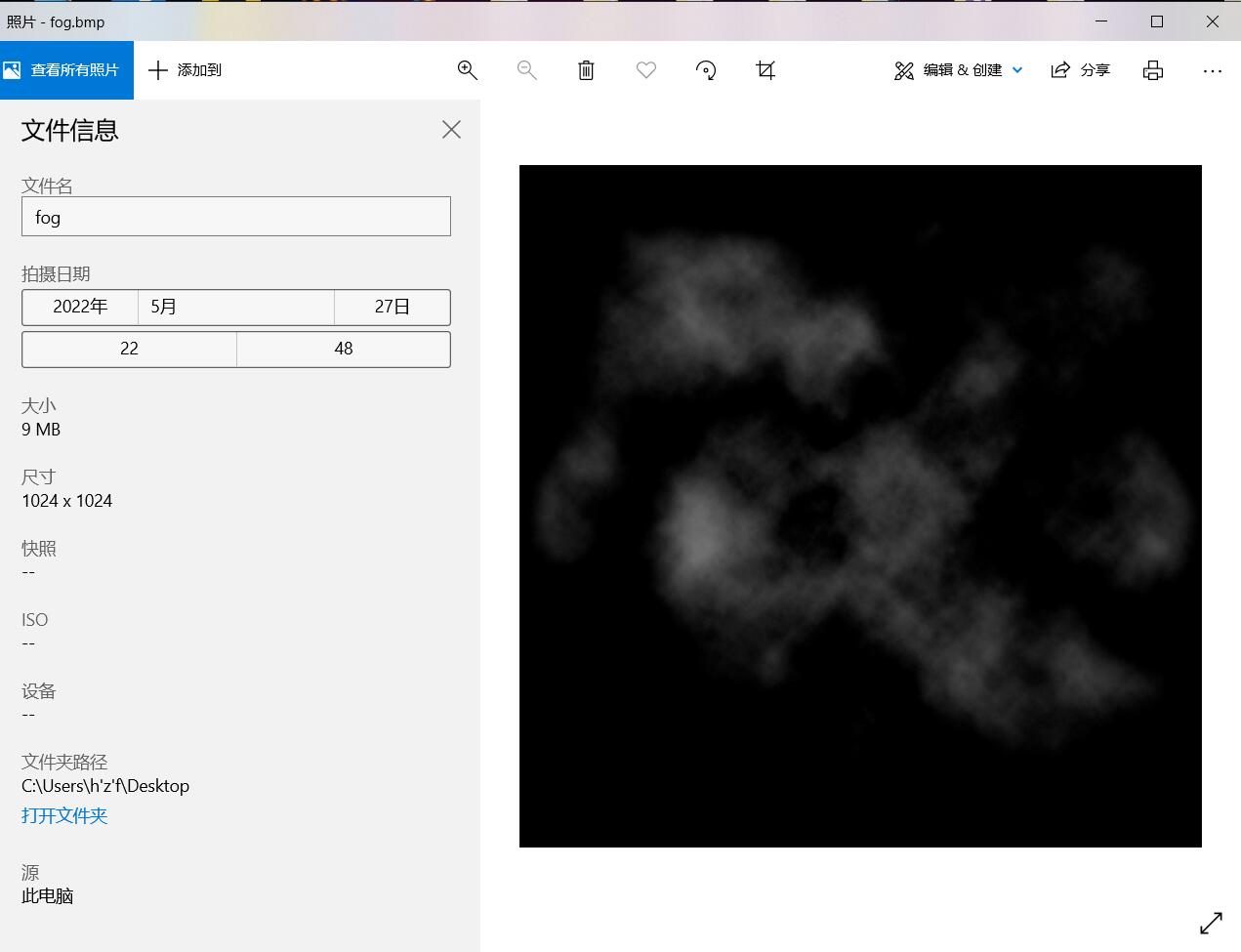
**测试数据（输入、输出）：（数据不用太多）**

****

压缩前：



压缩后



**第4题 rle**

**算法描述：**

**压缩：**

**记当前重复的元素为cur，记录当前位置元素为tmp，重复长度为relen，当cur！=tmp时，重新记录cur，并根据此时的relen，将relen转化为二进制数，并进行压缩**

**解压缩：**

**根据前面的二进制数，得到后面字符的重复次数，之后直接写入文件此重复字符**

**源程序：**

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

//实现压缩功能

void zip(char \*inname, char \*outname)

{

//定义指向文件的指针

FILE \*in = fopen(inname, "rb"), \*out = fopen(outname, "wb");

//定义重复字符的长度

int relen;

//定义当前字符cur，和记录的重复字符tmp

char cur, tmp;

//当指针为空，说明文件打开失败

if(in == NULL)

printf("文件打开失败");

else

{

//首先读取第一个字符，让储存在cur中，并令tmp = cur

cur = fgetc(in);

tmp = cur;

//初始化relen = 1

relen = 1;

//当没有到文件末尾时，进行压缩操作

while(!feof(in))

{

//读取下一个字符，并将此字符与上一个重复字符进行比较

//若相等，则说明重复，此时令relen ++

cur = fgetc(in);

if(cur == tmp)

relen ++ ;

//若不相等，则说明重复字符结束，此时对之前的重复字符进行压缩并写入out文件中

else

{

fputc(relen, out); //写入二进制数

fputc(tmp, out); //写入重复的字符

tmp = cur; //令tmp等于下一个非重复的字符

relen = 1; //重制relen = 1

}

}

}

//关闭文件

fclose(in);

fclose(out);

}

//实现解压功能

void unzip(char \*inname, char \*outname)

{

//定义指向文件的指针

FILE \*in = fopen(inname, "rb"), \*out = fopen(outname, "wb");

//定义并初始化relen = 1

int relen = 1;

char cur;

//当指针为空，说明文件打开失败

if(in == NULL)

printf("文件打开失败");

else

{

//当没有到达文件末尾时，进行解压操作

while(!feof(in))

{

//获取relen的值（int）

relen = fgetc(in);

if(feof(in)) break;

//获取重复的字符

cur = fgetc(in);

//进行写入

while(relen -- )

fputc(cur, out);

}

}

//关闭文件

fclose(in);

fclose(out);

}

int main(int argc, char \*argv[])

{

/\*

//判断指令，进行压缩/解压缩

if(!strcmp(argv[2], "-d"))

//unzip(argv[1], argv[3]);

unzip("toBeCompress.txt", "Compressed.txt");

else if(!strcmp(argv[2], "-c"))

//zip(argv[1], argv[3]);

zip("toBeCompress.txt", "Compressed.txt");

else

printf("输入有误，请重新输入");

\*/

//zip("text1.txt", "text2.txt");

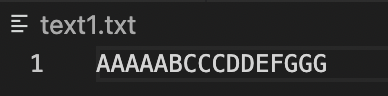
unzip("text2.txt", "text3.txt");

return 0;

}

**测试数据（输入、输出）：（数据不用太多）**

压缩前（text1里）：

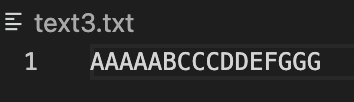


压缩文件（text2里）：

二进制数无法显示



对文件再次解压（text3里）：



和原来相同

**第5题 Libsys**

**算法描述：**

**使用结构体来储存信息：**

**BOOK：馆藏数量、剩余数量、图书名、图书编号、借阅条数、借阅记录**

**USER：借书数量、借书名称、用户ID、用户名字**

**ADMIN：管理员ID、管理员密码**

**之后使用vector来储存三个信息，即定义 vector<BOOK> book, vector<USER> user, vector<ADMIN> admin，来储存信息**

**操作的实现基本上都是遍历数据, 进行相应的修改. 比较简单, 不再赘述.**

**源程序：**

#include <iostream>

#include <algorithm>

#include <vector>

#include <cstring>

#include <cstdlib>

#include <fstream>

#include <cstdio>

#define BOOKFILE "book.txt"

#define USERFILE "user.txt"

#define ADMINFILE "admin.txt"

using namespace std;

typedef struct

{

int WholeNum; //馆藏数量

int LeftNum; //剩余数量

char name[100]; //图书名

char ID[100]; //图书编号

int RecordNum; //借阅条数

char record[100][100];//借阅记录

}BOOK;

typedef struct

{

int BorrowBookNum; //借书数量

char BorrowBook[10][100]; //借书名称,假令一个人同时最多借10本书

char ID[100]; //用户ID

char name[100]; //用户名字

}USER;

typedef struct

{

char ID[100]; //管理员ID

char password[100]; //管理员密码

}ADMIN;

vector<BOOK> book;

vector<USER> user;

vector<ADMIN> admin;

int AllUser;

int AllBook;

int AllAdmin;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

文件操作部分

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//book部分的读与写

void OpenBookFile()

{

FILE \*bookinfo = fopen(BOOKFILE, "r");

fscanf(bookinfo, "%d", &AllBook);

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

{

BOOK tmp;

fscanf(bookinfo, "%d%d%s%s%d", &tmp.WholeNum, &tmp.LeftNum, tmp.name, tmp.ID, &tmp.RecordNum);

for(int j = 0; j < tmp.RecordNum; j ++ )

fscanf(bookinfo, "%s", tmp.record[j]);

book.push\_back(tmp);

}

fclose(bookinfo);

}

void PrintBookFile()

{

FILE \*bookprint = fopen(BOOKFILE, "w");

fprintf(bookprint, "%d ", AllBook);

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

{

fprintf(bookprint, "%d %d %s %s %d ", book[i].WholeNum, book[i].LeftNum, book[i].name, book[i].ID, book[i].RecordNum);

for(int j = 0; j < book[i].RecordNum; j ++ )

fprintf(bookprint, "%s ", book[i].record[j]);

}

fclose(bookprint);

}

//admin部分的读与写

void OpenAdminFile()

{

FILE \*admininfo = fopen(ADMINFILE, "r");

fscanf(admininfo, "%d", &AllAdmin);

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

{

ADMIN tmp;

fscanf(admininfo, "%s%s", tmp.ID, tmp.password);

admin.push\_back(tmp);

}

fclose(admininfo);

}

void PrintAdminFile()

{

FILE \*adminprint = fopen(ADMINFILE, "w");

fprintf(adminprint, "%d ", AllAdmin);

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

fprintf(adminprint, "%s %s ", admin[i].ID, admin[i].password);

fclose(adminprint);

}

//user部分的读与写

void OpenUserFile()

{

FILE \*userinfo = fopen(USERFILE, "r");

fscanf(userinfo , "%d", &AllUser);

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

{

USER tmp;

fscanf(userinfo ,"%s%s%d", tmp.name, tmp.ID, &tmp.BorrowBookNum);

for(int j = 0; j < tmp.BorrowBookNum; j ++ )

fscanf(userinfo, "%s", tmp.BorrowBook[j]);

user.push\_back(tmp);

}

fclose(userinfo);

}

void PrintUserFile()

{

FILE \*userprint = fopen(USERFILE, "w");

fprintf(userprint, "%d ", AllUser);

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

{

fprintf(userprint, "%s %s %d ", user[i].name, user[i].ID, user[i].BorrowBookNum);

for(int j = 0; j < user[i].BorrowBookNum; j ++ )

fprintf(userprint, "%s ", user[i].BorrowBook[j]);

}

fclose(userprint);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

程序部分指令部分

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//初始化函数，对文件进行读取

void InitOpen()

{

OpenBookFile();

OpenAdminFile();

OpenUserFile();

}

//保存函数，每次操作后进行保存

void Save()

{

PrintBookFile();

PrintAdminFile();

PrintUserFile();

}

//退出系统函数，直接结束程序

void ExistSys()

{

cout << "感谢您的使用，再见！" << endl;

system("pause");

exit(0);

}

//展示书本信息函数

void ShowBookInfo(int i)

{

cout << "以下为该本图书的具体信息：" << endl;

cout << "【 书名 】: " << book[i].name << endl;

cout << "【 ID 】: " << book[i].ID << endl;

cout << "【 库存/馆藏 】: " << book[i].LeftNum << "/" << book[i].WholeNum << endl;

cout << "【 借阅记录条数 】: " << book[i].RecordNum << endl;

}

//展示用户信息函数

void ShowUserInfo(int i)

{

cout << "以下为该用户的具体信息：" << endl;

cout << "【 姓名 】: " << user[i].name << endl;

cout << "【 ID 】: " << user[i].ID << endl;

cout << "【 借阅书数 】: " << user[i].BorrowBookNum << endl;

cout << "【 具体书目如下 】: " << endl;

for (int j = 0; j < user[i].BorrowBookNum; j ++ )

cout << user[i].BorrowBook[j] << endl;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

管理员部分

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//输出管理员的目录函数

void AdminMenu(char\* id)

{

cout << "亲爱的" << id << "，欢迎！" << endl;

cout << "---------------请选择功能---------------" << endl;

cout << "-------------1.录入图书信息-------------" << endl;

cout << "-------------2.修改图书信息-------------" << endl;

cout << "-------------3.删除图书信息-------------" << endl;

cout << "-------------4.录入用户信息-------------" << endl;

cout << "-------------5.修改用户信息-------------" << endl;

cout << "-------------6.删除用户信息-------------" << endl;

cout << "---------------7.退出系统---------------" << endl;

}

//输入库存图书函数

void BookEnter()

{

int booknum;

BOOK tmp;

cout << "您好，请问您要录入几种书？" << endl;

cin >> booknum;

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

{

cout << "请输入第" << i + 1 << "种图书的数量: ";

cin >> tmp.WholeNum;

tmp.LeftNum = tmp.WholeNum;

cout << "请输入第" << i + 1 << "种图书的名字: ";

cin >> tmp.name;

cout << "请输入第" << i + 1 << "种图书的ID: ";

cin >> tmp.ID;

AllBook ++ ;

book.push\_back(tmp);

}

Save();

cout << "您已成功完成录入！";

system("pause");

system("clc");

}

//对库存图书进行修改函数

void BookModify()

{

int i;

char bookid[30];

int flag = 0;

cout << "输入您想要修改的书籍的编号：" ;

cin >> bookid;

for(i = 0; i < AllBook; i ++ )

{

if(!strcmp(bookid, book[i].ID))

{

flag = 1;

break;

}

flag = 0;

}

if(!flag)

cout << "未找到您要修改的图书，请检查后重新输入！" << endl;

else

{

ShowBookInfo(i);

cout << "请选择以下修改操作：" << endl;

cout << "1.修改书名" << endl;

cout << "2.修改ID" << endl;

cout << "3.修改馆藏数量" << endl;

int choice;

cin >> choice;

int originNum = book[i].WholeNum;

switch (choice)

{

case 1:

cout << "请输入新书名: ";

cin >> book[i].name;

break;

case 2:

cout << "请输入新ID: ";

cin >> book[i].ID;

break;

case 3:

cout << "请输入新馆藏数: ";

cin >> book[i].WholeNum;

book[i].LeftNum += book[i].WholeNum - originNum;

break;

default:

cout << "指令输入错误，请重新输入！";

break;

}

cout << "您已经成功完成修改！" << endl;

Save();

system("pause");

system("clc");

}

}

//对库存图书进行删除函数

void BookDelete()

{

int i;

char bookid[30];

int flag = 0;

cout << "输入您想要删除的书籍的编号：" ;

cin >> bookid;

for(i = 0; i < AllBook; i ++ )

{

if(!strcmp(bookid, book[i].ID))

{

flag = 1;

break;

}

flag = 0;

}

if(!flag)

cout << "未找到您要删除的图书，请检查后重新输入！" << endl;

else

{

book.erase(book.begin() + i);

AllBook -- ;

cout << "您已成功完成删除！" << endl;

Save();

system("pause");

system("clc");

}

}

//输入用户函数

void UserEnter()

{

int userNum;

cout << "您好，请问您要输入几位用户的信息？" << endl;

cin >> userNum;

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

{

USER tmp;

cout << "请输入第" << i + 1 << "位用户的名字: " << endl;

cin >> tmp.name;

cout << "请输入第" << i + 1 << "位用户的ID: " << endl;

cin >> tmp.ID;

cout << "请输入第" << i + 1 << "位用户借书数: " << endl;

cin >> tmp.BorrowBookNum;

if(tmp.BorrowBookNum != 0)

{

for(int j = 0; j < tmp.BorrowBookNum; j ++ )

{

char bookID[30];

int flag = 0, k = 0;

cout << "请输入第" << j + 1 << "本书的编号: ";

cin >> bookID;

for(k = 0; k < AllBook; k ++ )

{

if(!strcmp(bookID, book[k].ID))

{

flag = 1;

break;

}

flag = 0;

}

if(!flag)

cout << "输入失败，不存在此图书，请检查后重新输入！";

else

{

book[k].LeftNum -- ;

strcpy(book[k].record[book[k].RecordNum], tmp.ID);

book[k].RecordNum ++ ;

strcpy(tmp.BorrowBook[j], bookID);

}

}

}

AllUser ++ ; //注意AllUser进行扩增

user.push\_back(tmp);

}

Save();

cout << "您已完成录入！" << endl;

system("pause");

system("clc");

}

//对用户借书信息进行修改函数

void ModifyUserBook(int num)

{

int j = 0, k = 0;

int flag = 0;

for(int i = 0; i < user[num].BorrowBookNum; i ++ )

{

for(j = 0; j < AllBook; j ++ )

{

flag = 0;

if(!strcmp(user[num].BorrowBook[i], book[i].ID))

{

flag = 1;

break;

}

}

if(!flag)

continue;

else

{

book[j].LeftNum ++ ;

}

}

cout << "请先输入新的借书数: " << endl;

cin >> user[num].BorrowBookNum;

for(int i = 0; i < user[num].BorrowBookNum; i ++ )

{

char bookid[30];

cout << "请输入第" << i + 1 << "本图书编号" << endl;

cin >> bookid;

for(k = 0; k < AllBook; k ++ )

{

flag = 0;

if(!strcmp(bookid, book[k].ID))

{

flag = 1;

break;

}

}

if(!flag)

continue;

else

{

book[k].LeftNum -- ;

strcpy(book[k].record[book[k].RecordNum], user[num].ID);

book[k].RecordNum ++ ;

}

}

}

//对用户信息进行修改函数

void UserModify()

{

char userid[30];

char bookid[30];

int flag = 0;

int i = 0;

cout << "输入您要更改用户的ID: " << endl;

cin >> userid;

for(i = 0; i < AllUser; i ++ )

{

if(!strcmp(userid, user[i].ID))

{

flag = 1;

break;

}

flag = 0;

}

if(!flag)

cout << "查询失败，不存在此ID用户，请检查后再重新输入！" << endl;

else

{

ShowUserInfo(i);

cout << "请选择修改操作：" << endl;

cout << "1.修改姓名" << endl;

cout << "2.修改ID" << endl;

cout << "3.修改借阅书籍" << endl;

int choice;

cin >> choice;

switch (choice)

{

case 1:

cout << "请输入新姓名: " << endl;

cin >> user[i].name;

break;

case 2:

cout << "请输入新ID: " << endl;

cin >> user[i].ID;

break;

case 3:

ModifyUserBook(i);

break;

default:

cout << "指令输入错误，请重新输入！" << endl;

break;

}

cout << "您已成功完成修改！" << endl;

Save();

system("pause");

system("clc");

}

}

//删除用户函数

void UserDelete()

{

char userid[30];

int flag = 0, i = 0;

cout << "请输入您要删除用户的ID: " << endl;

cin >> userid;

for(i = 0; i < AllUser; i ++ )

{

flag = 0;

if(!strcmp(userid, user[i].ID))

{

flag = 1;

break;

}

}

//若用户有未归还图书则不能删除用户

if(!flag)

cout << "删除失败，不存在该用户，请检查ID后重新输入！";

else

{

if(user[i].BorrowBookNum != 0)

cout << "删除失败，该用户仍有未归还图书！" << endl;

else

{

user.erase(user.begin() + i);

cout << "您已经成功删除该用户！" << endl;

Save();

system("pause");

system("clc");

}

}

}

//判断管理员密码是否正确函数

bool isCorrectAdminPassword(char \*id, char \*password)

{

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

if(!strcmp(id, admin[i].ID))

if(!strcmp(password, admin[i].password))

return true;

return false;

}

//管理员功能的集合

void AdminMain(char\* id)

{

int choice;

while(true)

{

AdminMenu(id);

cin >> choice;

switch (choice)

{

case 1:

BookEnter();

break;

case 2:

BookModify();

break;

case 3:

BookDelete();

break;

case 4:

UserEnter();

break;

case 5:

UserModify();

break;

case 6:

UserDelete();

break;

case 7:

ExistSys();

break;

default:

cout << "请输入正确的指令！";

break;

}

system("pause");

system("cls");

}

}

//判断是否为管理员

bool isAdmin(char\* id)

{

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

if(id == admin[i].ID)

return true;

return false;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

用户部分

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//输出用户的目录函数

void UserMenu(char\* id)

{

cout << "亲爱的" << id << "，欢迎！" << endl;

cout << "---------------请选择功能---------------" << endl;

cout << "---------------1.我要借书---------------" << endl;

cout << "---------------2.我要还书---------------" << endl;

cout << "-------------3.查询图书信息-------------" << endl;

cout << "---------------4.退出系统---------------" << endl;

}

//用户借书的函数

void BorrowBook(int j)

{

char bookid[30];

int flag = 0, i = 0;

cout << "请输入您要借的书的编号: " << endl;

cin >> bookid;

for(i = 0; i < AllBook; i ++ )

{

flag = 0;

if(!strcmp(bookid, book[i].ID))

{

flag = 1;

break;

}

}

if(!flag)

cout << "该图书不存在，请检查后重新输入！" << endl;

else

{

if(book[i].LeftNum > 0)

{

book[i].LeftNum -- ;

strcpy(user[j].BorrowBook[user[j].BorrowBookNum], bookid);

user[j].BorrowBookNum ++ ;

strcpy(book[i].record[book[i].RecordNum], user[j].ID);

book[i].RecordNum ++ ;

cout << "成功借出" << endl;

}

else

{

cout << "抱歉，该书已经全部借出，借书失败！" << endl;

return;

}

}

Save();

system("pause");

system("clc");

}

//用户还书函数

void ReturnBook(int j)

{

char bookid[30];

int flag = 0, i = 0;

cout << "请输入您想归还的书的ID: " << endl;

cin >> bookid;

for(i = 0; i < AllBook; i ++ )

{

flag = 0;

if(!strcmp(bookid, book[i].ID))

{

flag = 1;

break;

}

}

if(!flag)

cout << "您要归还的图书不存在！" << endl;

else

{

int flag2 = 0, k = 0;

for(k = 0; k < user[j].BorrowBookNum; k ++ )

{

flag2 = 0;

if(!strcmp(bookid, user[j].BorrowBook[k]))

{

flag2 = 1;

break;

}

}

if(!flag2)

cout << "您并未借阅该图书，无法归还！" << endl;

else

{

for(int l = k; l < user[j].BorrowBookNum; l ++ )

strcpy(user[j].BorrowBook[l], user[j].BorrowBook[l + 1]);

user[j].BorrowBookNum -- ;

book[i].LeftNum ++ ;

cout << "还书成功！" << endl;

}

}

Save();

system("pause");

system("clc");

}

//找书函数

void SearchBook()

{

char bookid[30];

int flag = 0, i = 0;

cout << "请输入您要查询图书的ID: " << endl;

cin >> bookid;

for(i = 0; i < AllBook; i ++ )

{

flag = 0;

if(!strcmp(bookid, book[i].ID))

{

flag = 1;

break;

}

}

if(!flag)

cout << "您查询的图书不存在！" << endl;

else

ShowBookInfo(i);

system("pause");

system("clc");

}

//寻找对应id的用户，在vector中的下角标

int findUser(char \*id)

{

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

if(!strcmp(id, user[i].ID))

return i;

return 0;

}

//user功能的集合

void UserMain(char\* id)

{

int i = findUser(id);

int choice;

while(true)

{

UserMenu(id);

cin >> choice;

switch (choice)

{

case 1:

BorrowBook(i);

break;

case 2:

ReturnBook(i);

break;

case 3:

SearchBook();

case 4:

ExistSys();

break;

default:

cout << "请输入正确的指令！";

break;

}

}

}

//判断是否是用户

bool isUser(char\* id)

{

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

if(!strcmp(id, user[i].ID))

return true;

return false;

}

int main(int argc, char \*argv[])

{

cout << "您好，欢迎来到图书管理系统" << endl;

InitOpen();

char \*id = argv[2];

//判断用户输入是否正确，若正确则转到对应菜单

if(!strcmp(argv[1], "-a")) //判断是否为管理员

{

//if(isAdmin(id))

AdminMain(id);

//else

// cout << "未查询到您的管理员信息，您没有权限访问" << endl;

}

else if(!strcmp(argv[1], "-u")) //判断是否为用户

{

if(isUser(id))

UserMain(id);

else

cout << "未查询到您的用户信息，您没有权限访问" << endl;

}

else //若指令错误直接返回错误的信息

cout << "错误的信息" << endl;

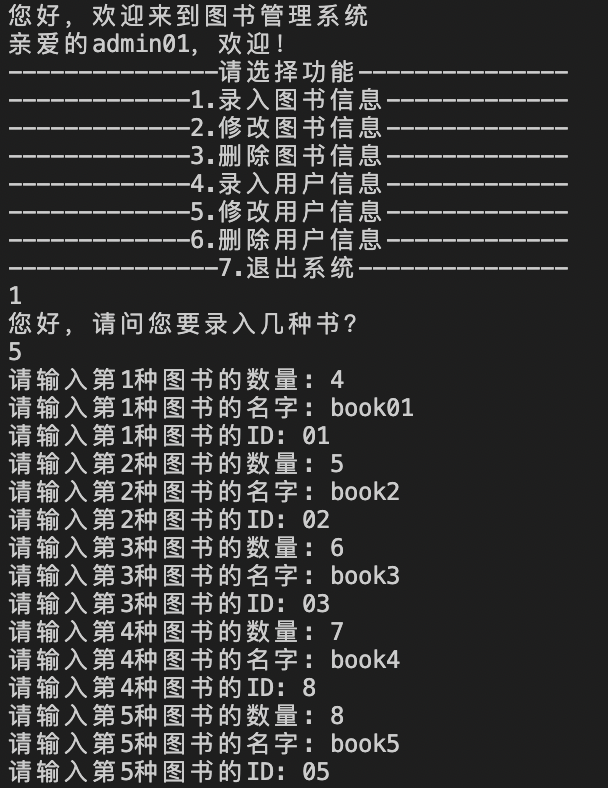
return 0;

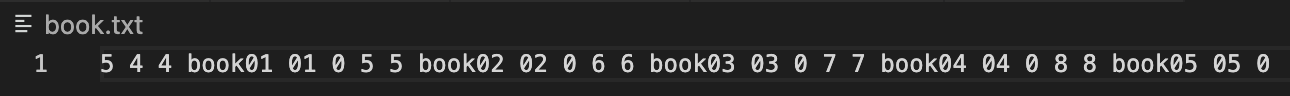
}

**测试数据（输入、输出）：（数据不用太多）**

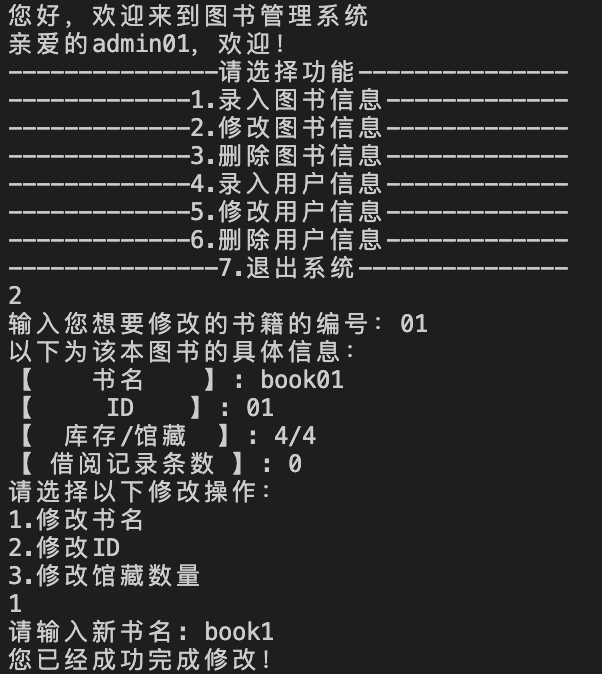
**管理员：**

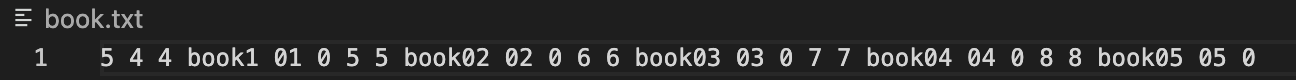
图书录入：



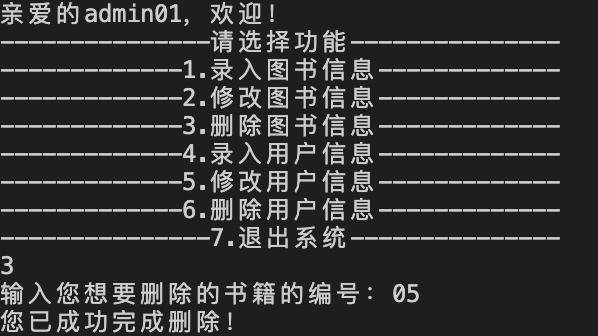


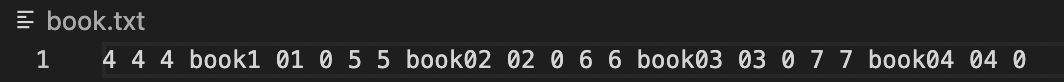
修改图书信息：



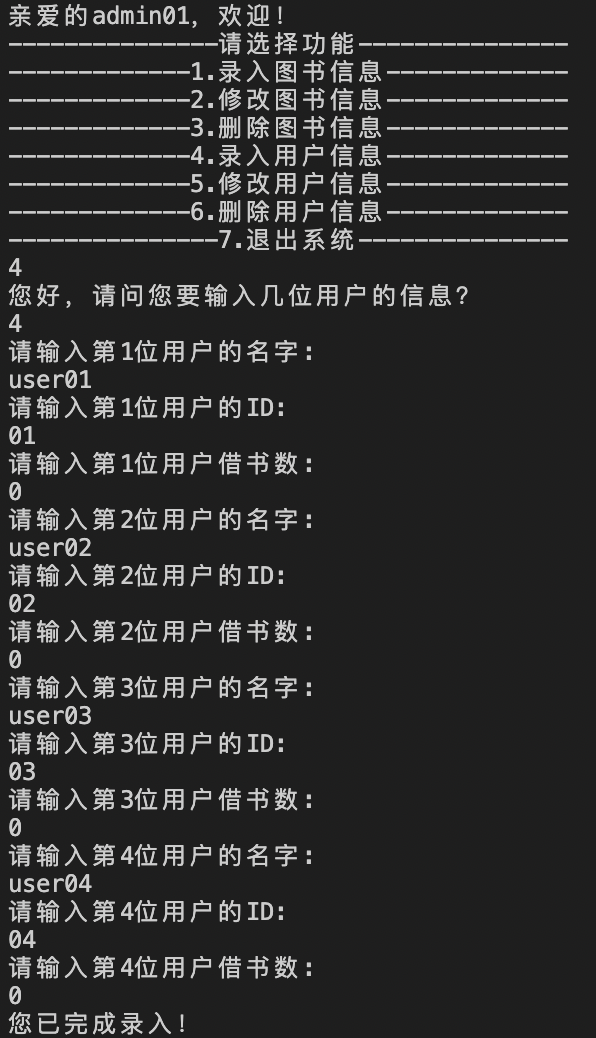


删除图书信息：



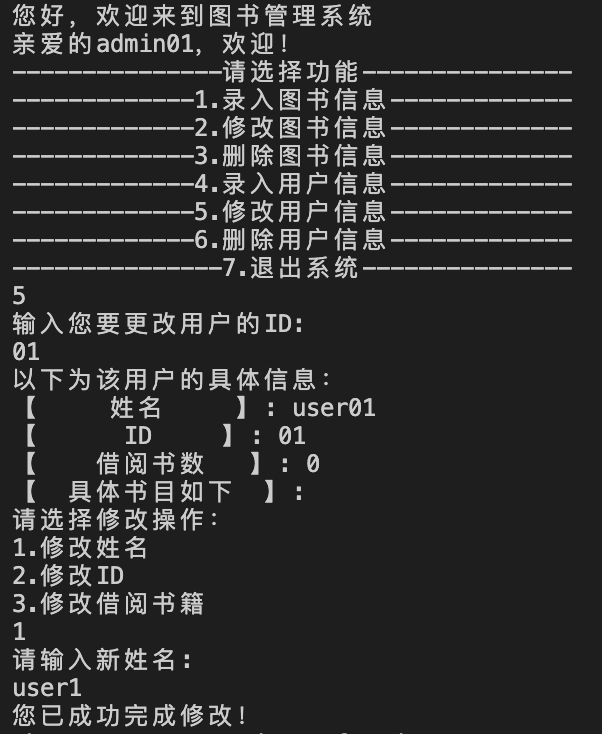


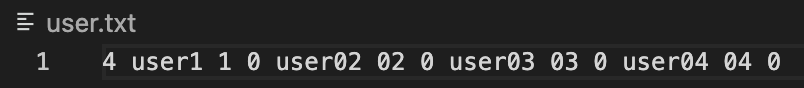
录入用户信息：



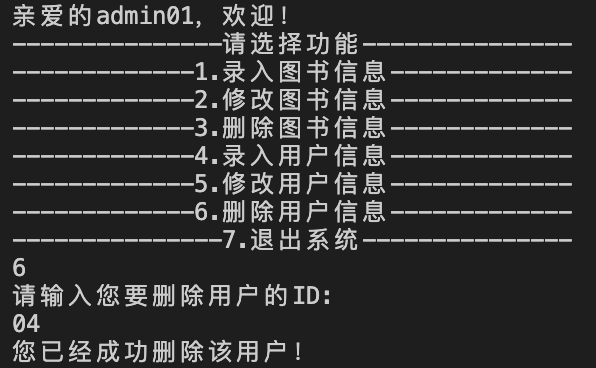


修改用户信息





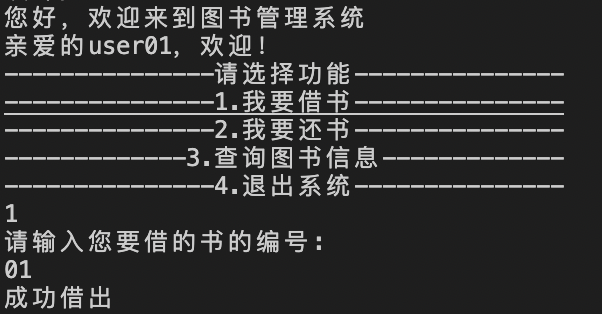
删除用户信息

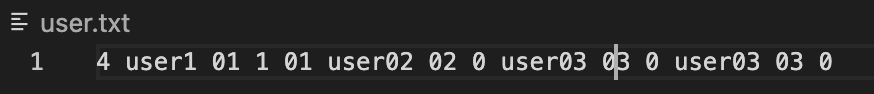




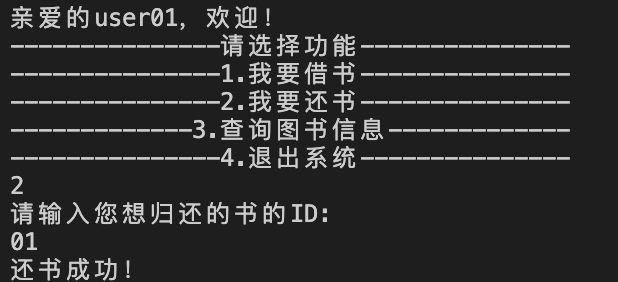
**用户：**

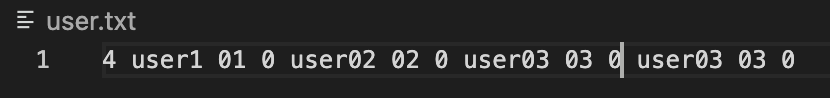
借书



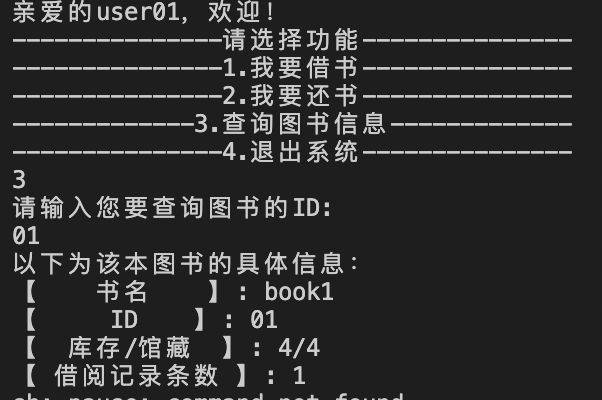


还书：





查询图书信息：



**课程设计小结**

1.学会了做长段代码的项目时的规划

2.学会了C语言文件操作的基本方法

3.学会了堆栈、位图、压缩的基本知识

4.学会了注释的基本方法

感想：难，但很快乐