

**课 程 实 验 报 告**

**课程名称： 计算机系统基础**

**实验名称： 数据的表示**

**院 系 ：计算机科学与技术学院**

**专业班级 ： 计算机本硕博2301班**

**学 号 ： U202315763**

**姓 名 ： 王家乐**

**指导教师 ： 李海波**

**2024 年 9 月 24 日**

**一、实验目的与要求**

⑴ 熟练掌握程序开发的基本方法，包括程序的编译、链接和调试；

⑵ 熟悉地址的计算方法、地址的内存转换；

⑶ 熟悉数据的表示形式。

**二、实验内容**

**任务1 数据存放的压缩与解压编程**

定义了 结构 student ，以及结构数组变量old\_s[N], new\_s[N]; (N=5)

struct student {

char name[8];

short age;

float score;

char remark[200]; // 备注信息

};

编写程序，输入N个学生的信息到结构数组old\_s中。将 old\_s[N] 中的所有信息依次紧凑(压缩)存放到一个字符数组message中，然后从 message 解压缩到结构数组 new\_s[N]中。打印压缩前(old\_s)、解压后(new\_s)的结果，以及压缩前、压缩后存放数据的长度。

要求：

1. 输入的第0个人姓名(name)为自己的名字，分数为学号的最后两位；
2. 编写指定接口的函数完成数据压缩

压缩函数有两个： int pack\_student\_bytebybyte(student\* s, int sno, char \*buf);

int pack\_student\_whole(student\* s, int sno, char \*buf);

s为待压缩数组的起始地址； sno 为压缩人数； buf 为压缩存储区的首地址；两个函数的返回均是调用函数压缩后的字节数。pack\_student\_bytebybyte要求一个字节一个字节的向buf中写数据；pack\_student\_whole要求对short、float字段都只能用一条语句整体写入，用strcpy实现串的写入。

1. 使用指定方式调用压缩函数

old\_s数组的前N1（N1=2）个记录压缩调用pack\_student\_bytebybyte 完成；后N2（N2==3）个记录压缩调用pack\_student\_whole，两种压缩函数都只调用1次。

（4） 使用指定的函数完成数据的解压

解压函数的格式：int restore\_student(char \*buf, int len, student\* s);

buf 为压缩区域存储区的首地址；len为buf中存放数据的长度；s为存放解压数据的结构数组的起始地址； 返回解压的人数。解压时不允许使用函数接口之外的信息（即不允许定义其他全局变量）

（5）仿照调试时看到的内存数据，以十六进制的形式，输出message的前20个字节的内容，并与调试时在内存窗口观察到的message的前20个字节比较是否一致。

（6）对于第0个学生的score，根据浮点数的编码规则指出其个部分的编码，并与观察到的内存表示比较，验证是否一致。  
 (7) 指出结构数组中个元素的存放规律，指出字符串数组、short类型的数、float型的数的存放规律。

**任务2 编写位运算程序**

按照要求完成给定的功能，并**自动判断程序**的运行结果是否正确。（从逻辑电路与门、或门、非门等等角度，实现CPU的常见功能。所谓自动判断，即用简单的方式实现指定功能，并判断两个函数的输出是否相同。）

1. int absVal(int x); 返回 x 的绝对值

仅使用 !、 ~、 &、 ^、 |、 +、 <<、 >>， 运算次数不超过 10次

判断函数： int absVal\_standard(int x) { return (x < 0) ? -x : x;}

1. int negate(int x); 不使用负号，实现 -x

判断函数： int netgate\_standard(int x) { return -x;}

1. int bitAnd(int x, int y); 仅使用 ~ 和 |，实现 &

判断函数： int bitAnd\_standard(int x, int y) { return x & y;}

1. int bitOr(int x, int y); 仅使用 ~ 和 &，实现 |
2. int bitXor(int x, int y); 仅使用 ~ 和 &，实现 ^
3. int isTmax(int x); 判断x是否为最大的正整数（7FFFFFFF），

只能使用 !、 ~、 &、 ^、 |、 +

1. int bitCount(int x); 统计x的二进制表示中 1 的个数

只能使用，! ~ & ^ | + << >> ，运算次数不超过 40次

1. int bitMask(int highbit, int lowbit); 产生从lowbit 到 highbit 全为1，其他位为0的数。例如bitMask(5,3) = 0x38 ；要求只使用 ! ~ & ^ | + << >> ；运算次数不超过 16次。
2. int addOK(int x, int y); 当x+y 会产生溢出时返回1，否则返回 0

仅使用 !、 ~、 &、 ^、 |、 +、 <<、 >>， 运算次数不超过 20次

1. int byteSwap(int x, int n, int m); 将x的第n个字节与第m个字节交换，返回交换后的结果。 n、m的取值在 0~3之间。  
   例：byteSwap(0x12345678, 1, 3) = 0x56341278

byteSwap(0xDEADBEEF, 0, 2) = 0xDEEFBEAD

仅使用 !、 ~、 &、 ^、 |、 +、 <<、 >>， 运算次数不超过 25次

**三、实验记录及问题回答**

**（1）任务 1 的算法思想、运行结果等记录**

1. 数据结构定义

struct student：定义了一个学生的结构体，包含四个字段：

name：学生姓名（字符数组，最多8个字符）

age：学生年龄（短整型）

score：学生成绩（浮点型）

remark：备注信息（字符数组，最多200个字符）

2. 输入函数 input

该函数通过循环接收用户输入，填充学生信息。对于每个学生，它依次输入姓名、年龄、成绩和备注，并使用 cin.ignore() 处理输入流中的换行符。

3. 打包函数

pack\_student\_bytebybyte：

使用逐字节的方式将学生信息打包到一个字节数组中。通过memcpy或直接赋值，将每个字段的字节拷贝到缓冲区 buf 中，并更新长度计数器 len。

pack\_student\_whole：

该函数使用更高层次的操作，利用 strcpy 直接拷贝字符串，将数据按结构体的顺序打包到缓冲区中。每个字段的大小和位置都通过指针算术计算。

4. 恢复函数 restore\_student

该函数将打包的字节缓冲区解包回学生结构体数组 s 中。它通过 memcpy 或直接赋值从缓冲区逐个字段恢复每个学生的信息，并更新位置指针 pos。

5. 输出函数 output

该函数负责打印学生的信息，包括姓名、年龄、成绩和备注。

6. 消息打印函数 print\_message

该函数打印缓冲区中的前20个字节，方便调试和查看打包数据的状态。

7. 主函数 main

主函数中，首先声明了学生数组和字节消息数组，然后调用输入函数收集学生信息。清屏后，通过调用打包函数将学生信息打包到消息缓冲区中。调用 print\_message 打印打包后的前20个字节。然后通过 restore\_student 恢复学生信息，并输出原始和恢复后的学生信息。

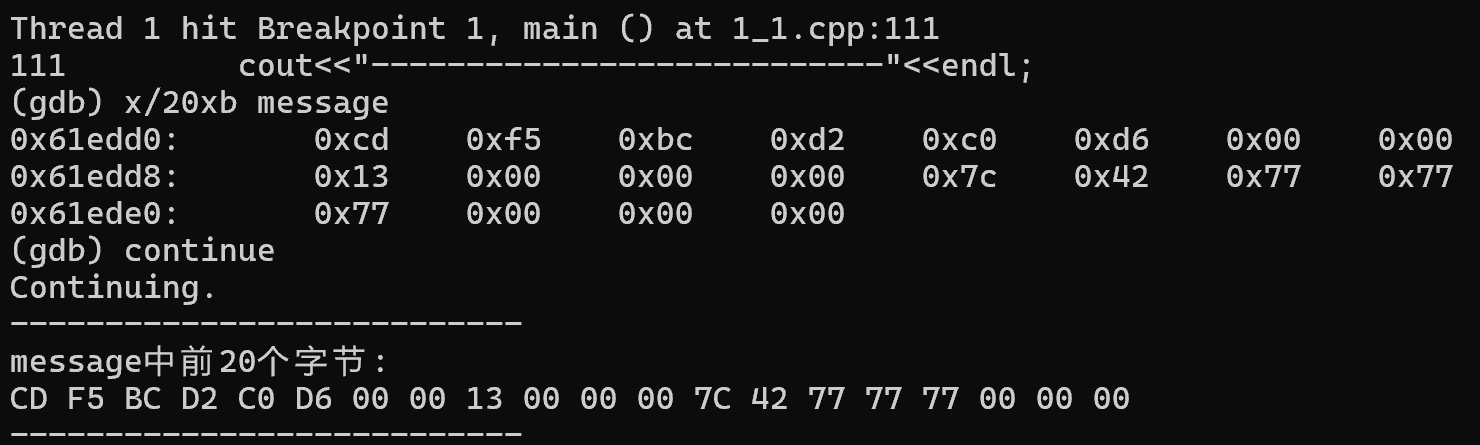


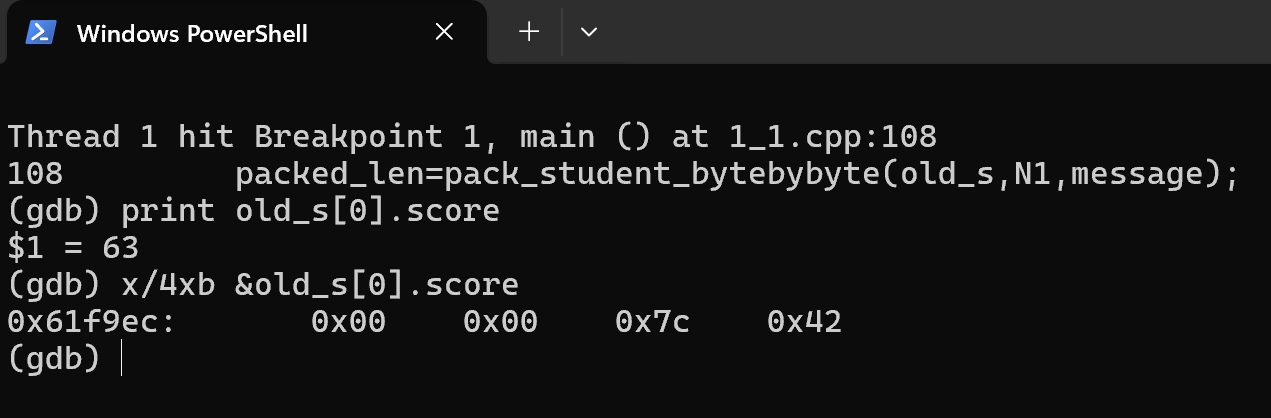
图1-内存窗口观察到的及打印的message的前20个字节

图2-内存窗口观察到的第0个学生score的编码

第0个学生的成绩为63(学号后两位),采用IEEE754标准编码：

63的二进制表示为111111=1.11111\*2^5,阶码为127+5=132，即10000100，且符号位为0，所以最终的二进制表示为0 10000100 11111000000000000000000，转换为十六进制0x427c0000，再以小端序表示0x00007c42，与内存中观察的编码一致。

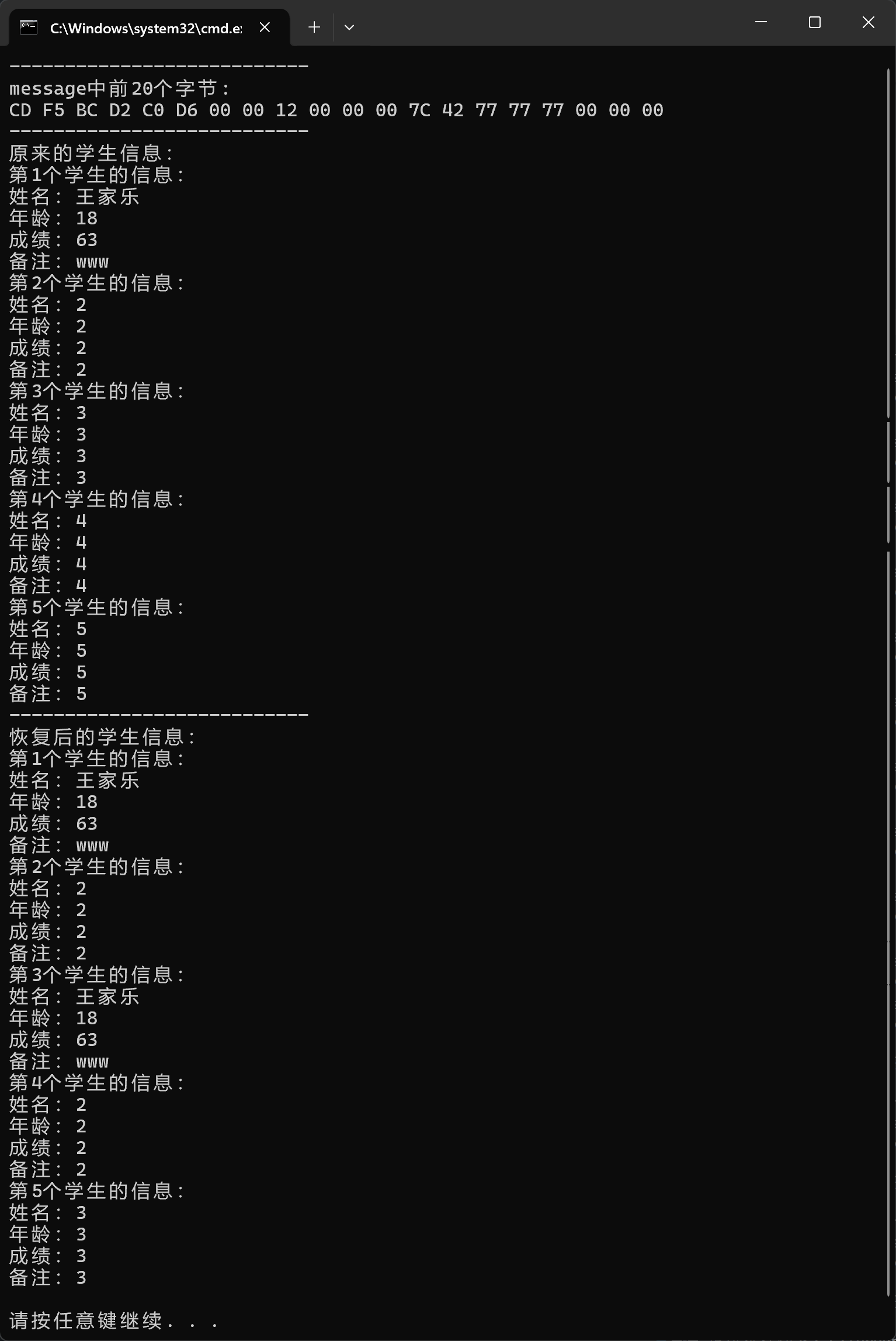


图3-任务一运行结果

**（2）任务 2 的算法思想、运行结果等记录**

1. absVal(int x)

通过右移操作获取符号位。如果 x 为负数，mask 为 -1（全为 1），如果 x 为非负数，mask 为 0。然后使用异或操作来决定是取反加一还是保持不变。

2. my\_negate(int x)

利用按位取反和加法来计算负数。~x 是 x 的按位取反，加上 1 形成负数。

3. bitAnd(int x, int y)

使用 De Morgan 定律，将与操作表示为按位取反和按位或的组合。x & y 可以表示为 ~(~x | ~y)。

4. bitOr(int x, int y)

同样使用 De Morgan 定律，将或操作表示为按位取反和按位与的组合。x | y 可以表示为 ~(~x & ~y)。

5. bitXor(int x, int y)

使用按位与和按位取反来实现异或。x ^ y 可以表示为 ~(x & y) & ~(~x & ~y)。

6. isTmax(int x)

判断 x 是否为最大正整数 0x7FFFFFFF。如果 x + 1 为 0，则 x 为最大正整数。通过符号位的比较确定条件。

7. bitCount(int x)

使用“分组统计”的方式，首先将每两位相加，然后每四位相加，以此类推，最终统计所有 1 的个数。

8. bitMask(int highbit, int lowbit)

通过创建高位和低位掩码，结合按位与和按位取反的操作，生成从 lowbit 到 highbit 全为 1 的位掩码。

9. addOK(int x, int y)

判断 x + y 是否会溢出。通过检查 x 和 y 的符号位及结果的符号位，判断是否发生溢出。

10. byteSwap(int x, int n, int m)

通过位移和掩码操作，提取出 n 和 m 字节，然后清除这两个字节，再将提取的字节交换位置并放回原数中。

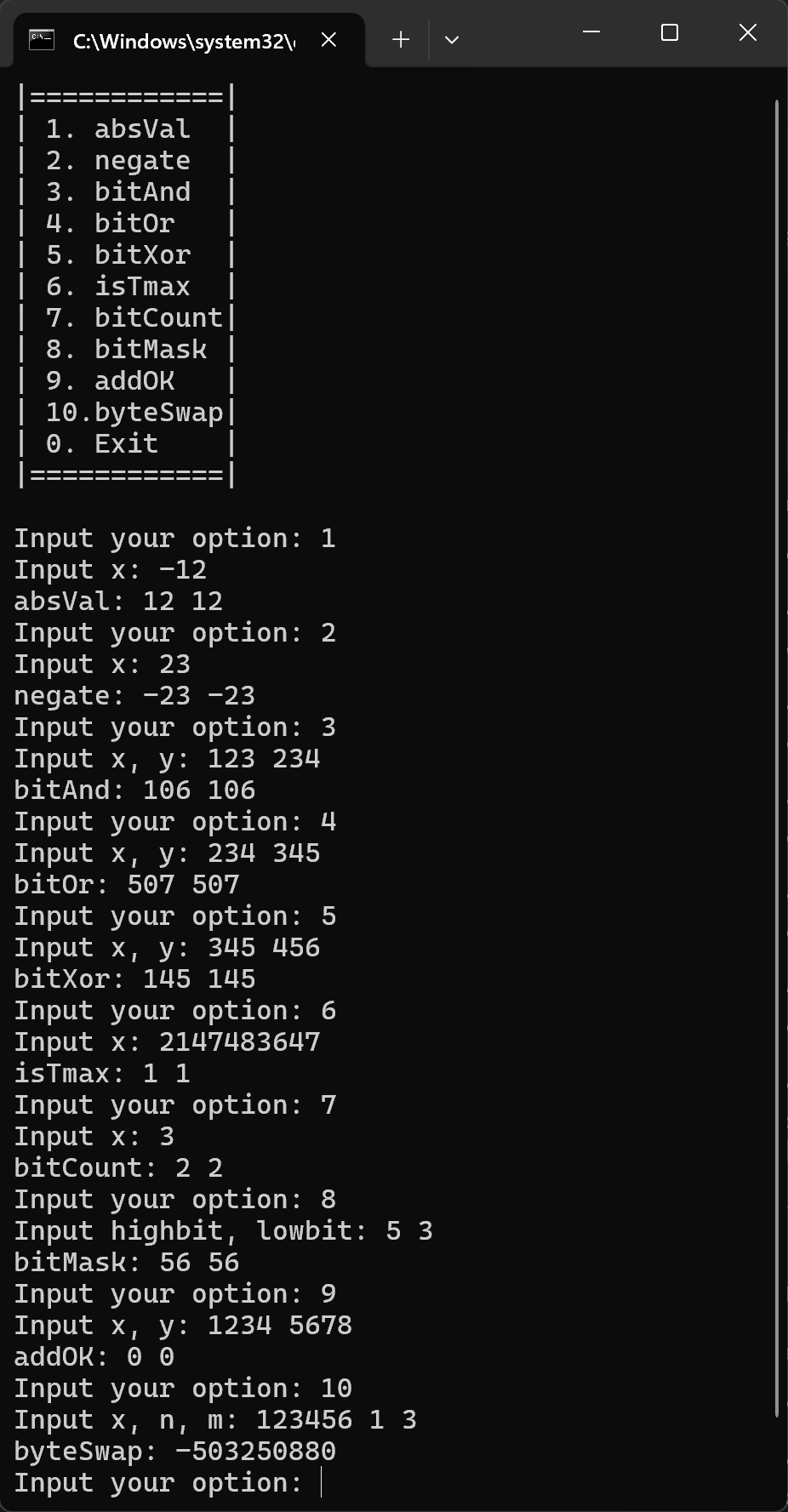


图4-任务二运行结果

**四、体会**

通过这次实验，我对数据压缩和位运算有了深入的理解。在数据压缩部分，我学会了如何将学生信息存储到字符数组中，并通过逐字节和整体压缩的方法进行解压。这让我掌握了数据结构在内存中的布局，尤其是字符数组、short和float类型的存储规律。

在位运算实验中，我利用逻辑电路的角度实现了如绝对值、按位与、按位或等功能。这些位运算让我认识到它们在效率和硬件处理方面的重要性，尤其是在实现absVal和bitCount函数时，锻炼了我对位操作的灵活运用能力。

此次实验增强了我对程序开发基本技能的掌握，提升了我在内存管理和底层运算方面的能力，为我今后的编程实践奠定了基础。

**五、源码**

任务一：

#include <iostream>

#include <cstring>

using namespace std;

#define N 5

#define N1 2

#define N2 3

struct student{

    char name[8];

    short age;

    float score;

    char remark[200]; // 备注信息

};

void input(student \*s){

    for(int i=0;i<N;++i){

        cout<<"请输入第"<<i+1<<"个学生的信息："<<endl;

        cout<<"姓名：";

        cin>>s[i].name;

        cout<<"年龄：";

        cin>>s[i].age;

        cout<<"成绩：";

        cin>>s[i].score;

        cout<<"备注：";

        cin.ignore(); // 忽略换行符

        cin.getline(s[i].remark, 200);

    }

}

int pack\_student\_bytebybyte(student \*s,int sno,char \*buf){

    int len=0;

    for(int i=0;i<sno;++i){

        // memcpy(buf+len,s[i].name,8);

        // len+=8;

        // memcpy(buf+len,&s[i].age,sizeof(short));

        // len+=sizeof(short);

        // memcpy(buf+len,&s[i].score,sizeof(float));

        // len+=sizeof(float);

        // memcpy(buf+len,s[i].remark,200);

        // len+=200;

        for (int j = 0; j < 8; ++j) buf[len++] = s[i].name[j];

        buf[len++] = (char)(s[i].age & 0xFF);

        buf[len++] = (char)((s[i].age >> 8) & 0xFF);

        char\* p = (char\*)&s[i].score;

        for (int j = 0; j < static\_cast<int>(sizeof(float)); ++j) buf[len++] = p[j];

        for (int j = 0; j < 200; ++j) buf[len++] = s[i].remark[j];

    }

    return len;

}

int pack\_student\_whole(student \*s,int sno,char \*buf){

    int len=0;

    for(int i=0;i<sno;++i){

        strcpy(buf+len,s[i].name);

        len+=8;

        \*(short\*)(buf + len) = s[i].age;

        len += sizeof(short);

        \*(float\*)(buf + len) = s[i].score;

        len += sizeof(float);

        strcpy(buf + len, s[i].remark);

        len += 200;

    }

    return len;

}

int restore\_student(char \*buf, int len, student\* s){

    int num=0;

    int pos=0;

    while(pos<len){

        memcpy(s[num].name,buf+pos,8);

        pos+=8;

        s[num].age=\*(short\*)(buf+pos);

        pos+=sizeof(short);

        s[num].score=\*(float\*)(buf+pos);

        pos+=sizeof(float);

        memcpy(s[num].remark,buf+pos,200);

        pos+=200;

        num++;

    }

    return num;

}

void output(student \*s,int count){

    for(int i=0;i<count;++i){

        cout<<"第"<<i+1<<"个学生的信息："<<endl;

        cout<<"姓名："<<s[i].name<<endl;

        cout<<"年龄："<<s[i].age<<endl;

        cout<<"成绩："<<s[i].score<<endl;

        cout<<"备注："<<s[i].remark<<endl;

    }

}

void print\_message(char \*buf, int len){

    cout<<"message中前20个字节:"<<endl;

    for(int i=0;i<len&&i<20;++i){

        printf("%02X ", (unsigned char)buf[i]);

    }

    cout<<endl;

}

int main(){

    struct student old\_s[N], new\_s[N];

    char message[2000];

    int packed\_len=0;

    input(old\_s);

    system("cls");

    packed\_len=pack\_student\_bytebybyte(old\_s,N1,message);

    packed\_len+=pack\_student\_whole(old\_s,N2,message+packed\_len);

    cout<<"---------------------------"<<endl;

    print\_message(message,packed\_len);

    int num=restore\_student(message,packed\_len,new\_s);

    cout<<"---------------------------"<<endl;

    cout<<"原来的学生信息："<<endl;

    output(old\_s,N);

    cout<<"---------------------------"<<endl;

    cout<<"恢复后的学生信息："<<endl;

    output(new\_s,num);

    return 0;

}

任务二：

#include <iostream>

#include <limits>

using namespace std;

/\* 返回 x 的绝对值 \*/

int absVal(int x){

    // 获取符号位：x为负数时mask为全1(-1),为正数时mask为0

    int mask = x >> 31;

    // x ^ mask 等价于 x 的取反或保持不变

    // 若x<0,则|x| = ~x + 1

    return (x ^ mask) - mask;

}

int absVal\_standard(int x){

    return (x < 0) ? -x : x;

}

/\* 不使用负号实,现 -x \*/

int my\_negate(int x){

    return ~x + 1;

}

int negate\_standard(int x){

    return -x;

}

/\* 仅使用 ~ 和 |,实现 & \*/

int bitAnd(int x, int y){

    return ~(~x | ~y);

}

int bitAnd\_standard(int x, int y){

    return x & y;

}

/\* 仅使用 ~ 和 &,实现 | \*/

int bitOr(int x, int y){

    return ~(~x & ~y);

}

int bitOr\_standard(int x, int y){

    return x | y;

}

/\* 仅使用 ~ 和 &,实现 ^ \*/

int bitXor(int x, int y){

    return ~(x & y) & ~(~x & ~y);

}

int bitXor\_standard(int x, int y){

    return x ^ y;

}

/\* 判断x是否为最大的正整数(7FFFFFFF)

只能使用 !、 ~、 &、 ^、 |、 + \*/

int isTmax(int x){

    // 计算 x + 1，并检查它是否为 0

    int x\_plus\_1 = x + 1;

    // 使用符号位判断，signX = x >> 31，如果 x 是正数，signX 为 0

    int signX = x >> 31;

    // 使用符号位判断，signSum = x\_plus\_1 >> 31，如果 x + 1 为负数，则 signSum 为 -1

    int signSum = x\_plus\_1 >> 31;

    // 如果 x + 1 为 0 且 x 是正数，返回 1；否则返回 0

    return !(~(signX ^ signSum) & signX) & !!(x\_plus\_1);

}

int isTmax\_standard(int x){

    return x == 0x7FFFFFFF;

}

/\*统计x的二进制表示中 1 的个数

只能使用,! ~ & ^ | + << >> ,运算次数不超过 40次 \*/

int bitCount(int x){

    int mask1 = 0x55 | (0x55 << 8);  // 01010101... 逐位统计

    mask1 = mask1 | (mask1 << 16);

    int mask2 = 0x33 | (0x33 << 8);  // 00110011... 逐位统计

    mask2 = mask2 | (mask2 << 16);

    int mask4 = 0x0F | (0x0F << 8);  // 00001111... 逐位统计

    mask4 = mask4 | (mask4 << 16);

    int mask8 = 0xFF | (0xFF << 16); // 前8位全1

    int mask16 = 0xFF | (0xFF << 8); // 前16位全1

    x = (x & mask1) + ((x >> 1) & mask1);  // 统计每两位内的1

    x = (x & mask2) + ((x >> 2) & mask2);  // 统计每四位内的1

    x = (x & mask4) + ((x >> 4) & mask4);  // 统计每八位内的1

    x = (x & mask8) + ((x >> 8) & mask8);  // 统计每16位内的1

    x = (x & mask16) + ((x >> 16) & mask16);  // 最终得出1的个数

    return x;

}

int bitCount\_standard(int x){

    // Brian Kernighan算法

    int cnt = 0;

    while(x){

        x &= x - 1; // 每次清除最低位的1

        cnt++;

    }

    return cnt;

}

/\* 产生从lowbit 到 highbit 全为1，其他位为0的数

只使用 ! ~ & ^ | + << >> ;运算次数不超过 16次 \*/

int bitMask(int highbit, int lowbit){

    unsigned int highMask = ~0U << lowbit;   // 从lowbit起全为1

    unsigned int lowMask = ~0U << (highbit + 1);  // 从highbit之后全为0

    return highMask & ~lowMask;   // 截取中间部分

}

int bitMask\_standard(int highbit, int lowbit){

    int mask = 0;

    for(int i = lowbit; i <= highbit; ++i){

        mask |= 1 << i;

    }

    return mask;

}

/\* 当x+y 会产生溢出时返回1,否则返回 0

仅使用 !、 ~、 &、 ^、 |、 +、 <<、 >>,运算次数不超过 20次 \*/

int addOK(int x, int y){

    // 符号位相同且结果的符号位与x的符号位不同

    int sum = x + y;

    int signX = (x >> 31)&1;

    int signY = (y >> 31)&1;

    int signSum = (sum >> 31)&1;

    return !(~(signX ^ signY)) & (signX ^ signSum);

}

int addOK\_standard(int x, int y){

    return (x > 0 && y > 0 && x + y < 0) || (x < 0 && y < 0 && x + y > 0);

}

/\* 将x的第n个字节与第m个字节交换,返回交换后的结果

n、m的取值在 0~3之间

仅使用 !、 ~、 &、 ^、 |、 +、 <<、 >>,运算次数不超过 25次 \*/

int byteSwap(int x, int n, int m){

    int n\_shift = n << 3;  // 将字节编号转化为位编号

    int m\_shift = m << 3;

    int n\_byte = (x >> n\_shift) & 0xFF;  // 提取n字节

    int m\_byte = (x >> m\_shift) & 0xFF;  // 提取m字节

    int mask = (0xFF << n\_shift) | (0xFF << m\_shift);  // 构造掩码去除两个字节

    x = x & ~mask;  // 清空n和m字节

    return x | (n\_byte << m\_shift) | (m\_byte << n\_shift);  // 交换字节后重新赋值

}

void printMenu(){

    cout << "|============|" << endl;

    cout << "| 1. absVal  |" << endl;

    cout << "| 2. negate  |" << endl;

    cout << "| 3. bitAnd  |" << endl;

    cout << "| 4. bitOr   |" << endl;

    cout << "| 5. bitXor  |" << endl;

    cout << "| 6. isTmax  |" << endl;

    cout << "| 7. bitCount|" << endl;

    cout << "| 8. bitMask |" << endl;

    cout << "| 9. addOK   |" << endl;

    cout << "| 10.byteSwap|" << endl;

    cout << "| 0. Exit    |" << endl;

    cout << "|============|" << endl << endl;

}

int main()

{

    int op=1;

    printMenu();

    while(op){

        cout<<"Input your option: ";

        cin>>op;

        // system("cls");

        // printMenu();

        switch(op){

            case 1:

            {

                int x;

                cout << "Input x: ";

                cin >> x;

                cout << "absVal: " << absVal(x) << " " << absVal\_standard(x) << endl;

                break;

            }

            case 2:

            {

                int x;

                cout << "Input x: ";

                cin >> x;

                cout << "negate: " << my\_negate(x) << " " << negate\_standard(x) << endl;

                break;

            }

            case 3:

            {

                int x,y;

                cout << "Input x, y: ";

                cin >> x >> y;

                cout << "bitAnd: " << bitAnd(x, y) << " " << bitAnd\_standard(x, y) << endl;

                break;

            }

            case 4:

            {

                int x,y;

                cout << "Input x, y: ";

                cin >> x >> y;

                cout << "bitOr: " << bitOr(x, y) << " " << bitOr\_standard(x, y) << endl;

                break;

            }

            case 5:

            {

                int x,y;

                cout << "Input x, y: ";

                cin >> x >> y;

                cout << "bitXor: " << bitXor(x, y) << " " << bitXor\_standard(x, y) << endl;

                break;

            }

            case 6:

            {

                int x;

                cout << "Input x: ";

                cin >> x;

                cout << "isTmax: " << isTmax(x) << " " << isTmax\_standard(x) << endl;

                break;

            }

            case 7:

            {

                int x;

                cout << "Input x: ";

                cin >> x;

                cout << "bitCount: " << bitCount(x) << " " << bitCount\_standard(x) << endl;

                break;

            }

            case 8:

            {

                int n,m;

                cout << "Input highbit, lowbit: ";

                cin >> n >> m;

                cout << "bitMask: "<< bitMask(n, m) << " " << bitMask\_standard(n, m) << endl;

                break;

            }

            case 9:

            {

                int x,y;

                cout << "Input x, y: ";

                cin >> x >> y;

                cout << "addOK: " << addOK(x, y) << " " << addOK\_standard(x, y) << endl;

                break;

            }

            case 10:

            {

                int x,n,m;

                cout << "Input x, n, m: ";

                cin >> x ;

                cin >> n >> m;

                cout << "byteSwap: " << byteSwap(x, n, m) << endl;

                break;

            }

            case 0:

                break;

            default:

                cout << "Invalid input\n";

        }

    }

    return 0;

}