

**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**

**专业班级 ： 计卓202201**

**学 号 ： U202215322**

**姓 名 ： 濮澍**

**指导教师 ： 许向阳**

**2024年 3月 24 日**

**一、实验目的与要求**

⑴ 熟练掌握程序开发平台(VS2019/VS2022/VS2023) 的基本用法，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**

**·函数思想**

bytebybyte在于一个字节一个字节复制，所以只需定义一个char\*指针，每次放入数据都指向下一个数据就行，对float和short需要分别根据二者字节数定义循环

函数中定义一个偏移量，记录当前操作时偏移量，最终返回时返回了用户输入的压缩后的字节数

whole 在于使用str函数和指针运算进行压缩，strcpy自不用说，而这里的float和short则需要将char\*指针强制类型转换为 float\*和short\* 再解引用赋值。Whole中的偏移量now\_add就不在是一个字节一个字节加，而是根据每次加入的数据量大小进行增加。而且在此函数中，buf+now\_add相当于指针用途

restore\_student相当于whole的相反操作

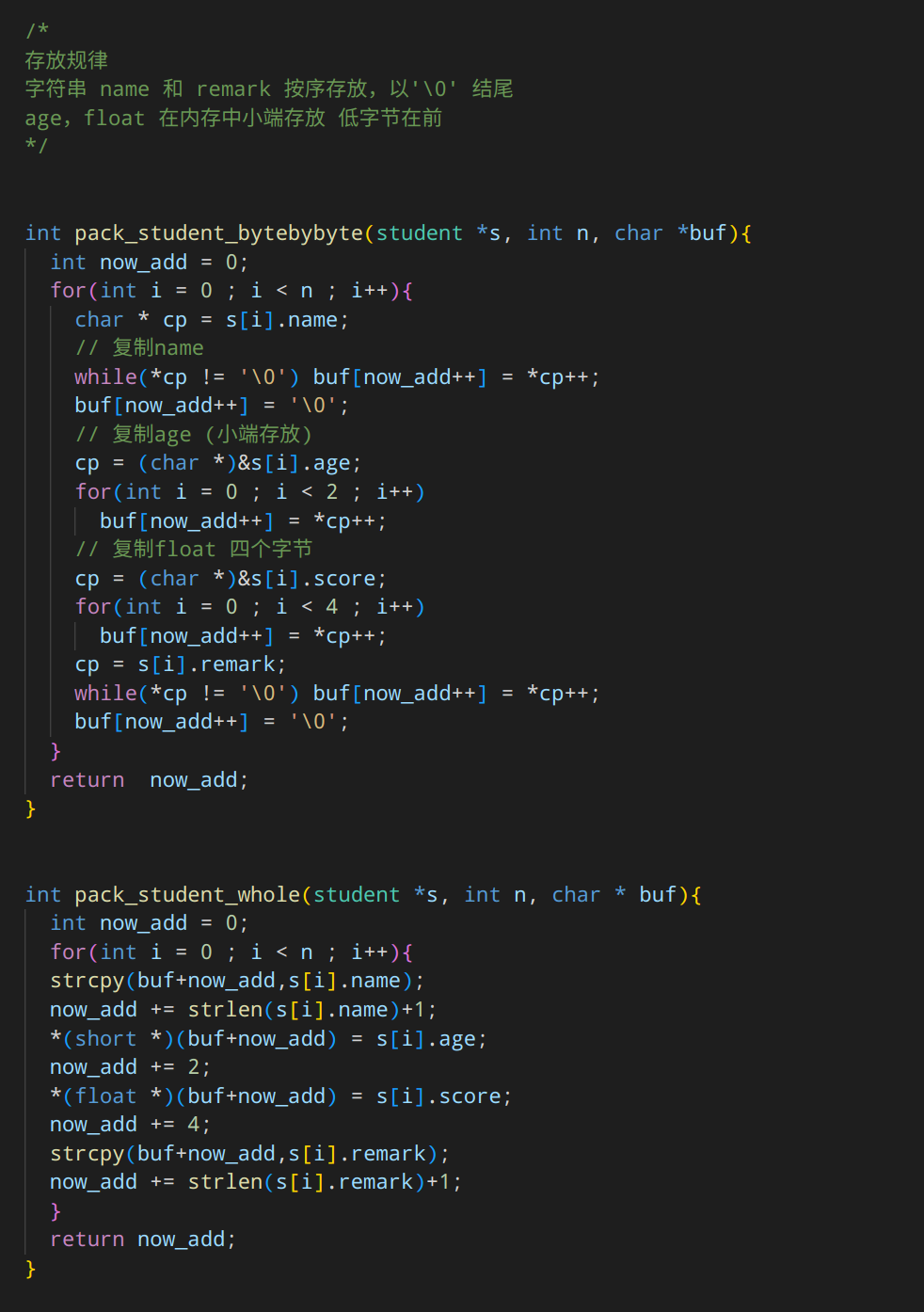


图1 压缩函数

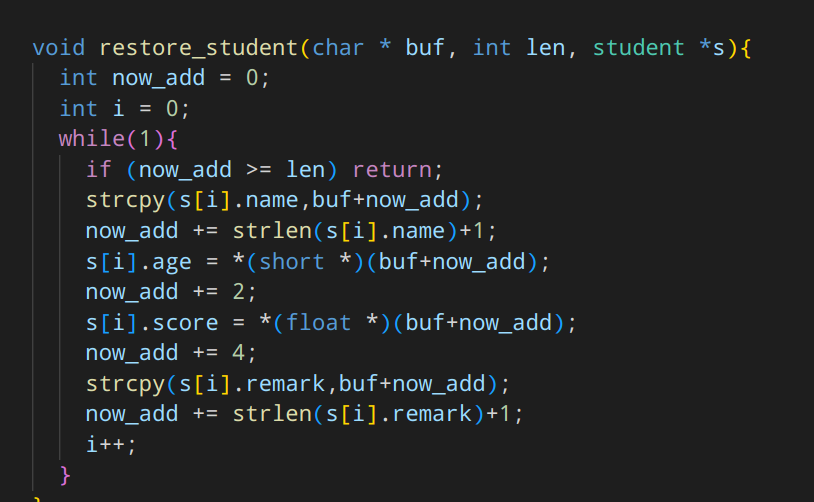


图2 解压缩函数

**·运行结果**

根据任务书，对内存等进行观察

Code总览

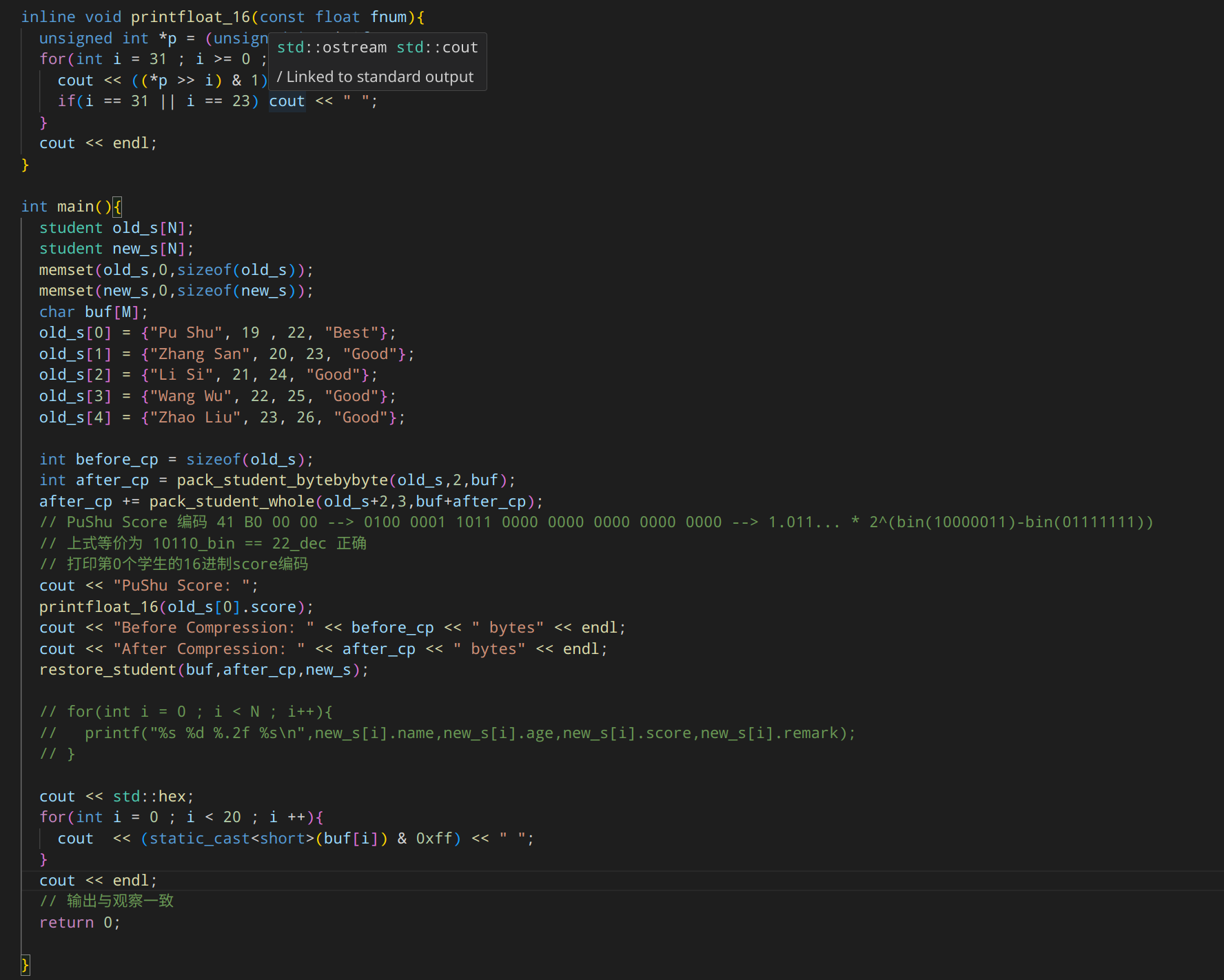


图3 main函数

图3中注释表示出了该学生分数22的float形式，在程序运行时，有下图4运行结果和分析一致

main函数中前两个学生通过bytebybyte压缩，后三个用whole压缩

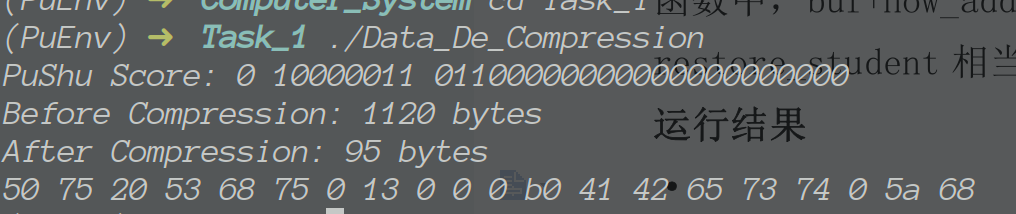


图4 运行结果

根据任务书，我们观察buff原本的前20个字节和输出的20个字节区别 发现二者一致

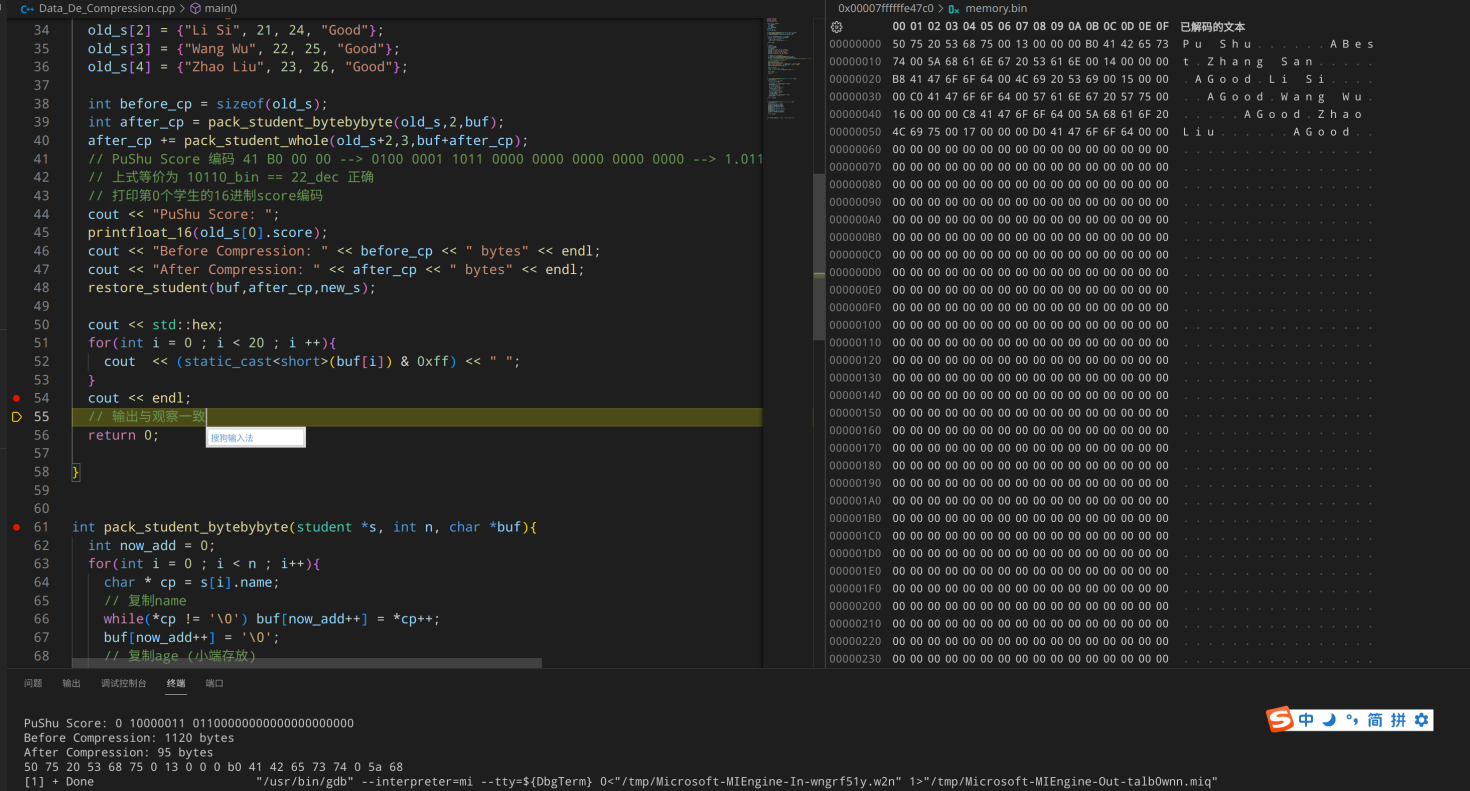


图6 压缩后内存窗口和输出对比

**·数据存放规律**

**存放规律**

**字符串 name 和 remark 按序存放，以'\0' 结尾**

**age，float 在内存中小端存放 低字节在前**

**（2）任务 2 的算法思想、运行结果等记录**

**·算法思想**

absVal与符号位密切相关，所以返回值一定用到’x>>31’ 是1 时-1取反 否则不变

neg 就是简单的取反加一

bitAnd/Or/Xor画出真值表就可求解

isTmax要判断输入是否是0x7FFFFFFF 该数+1即为0x80000000，与Tmax相反，异或为0

又要特判x==-1

bitCount 整体思路还是循环，只是这里通过每次取更多位上的数据后，相加获得结果，比较难 意思是类似合并相邻石子，每次都合并一组，最终合并成最大组

bitMask 需要获取lowbit->highbit 位运算就是取中间部分

addOK 符号位不同 一定不溢出，输出1 ； 符号位相同，相加后符号位相反则溢出

byteswap 通过掩码和异或操作获取n字节和m字节的数据，通过左右移进行交换

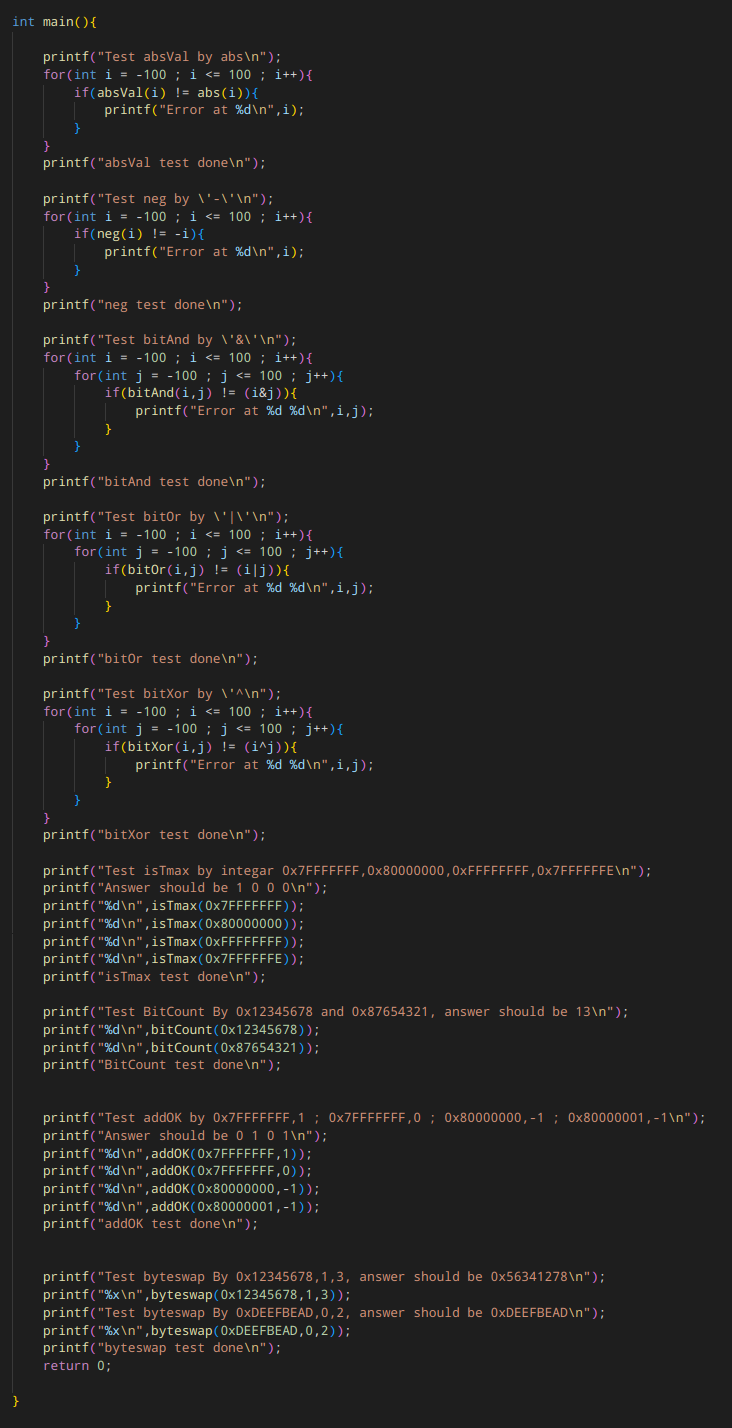


图7 测试函数

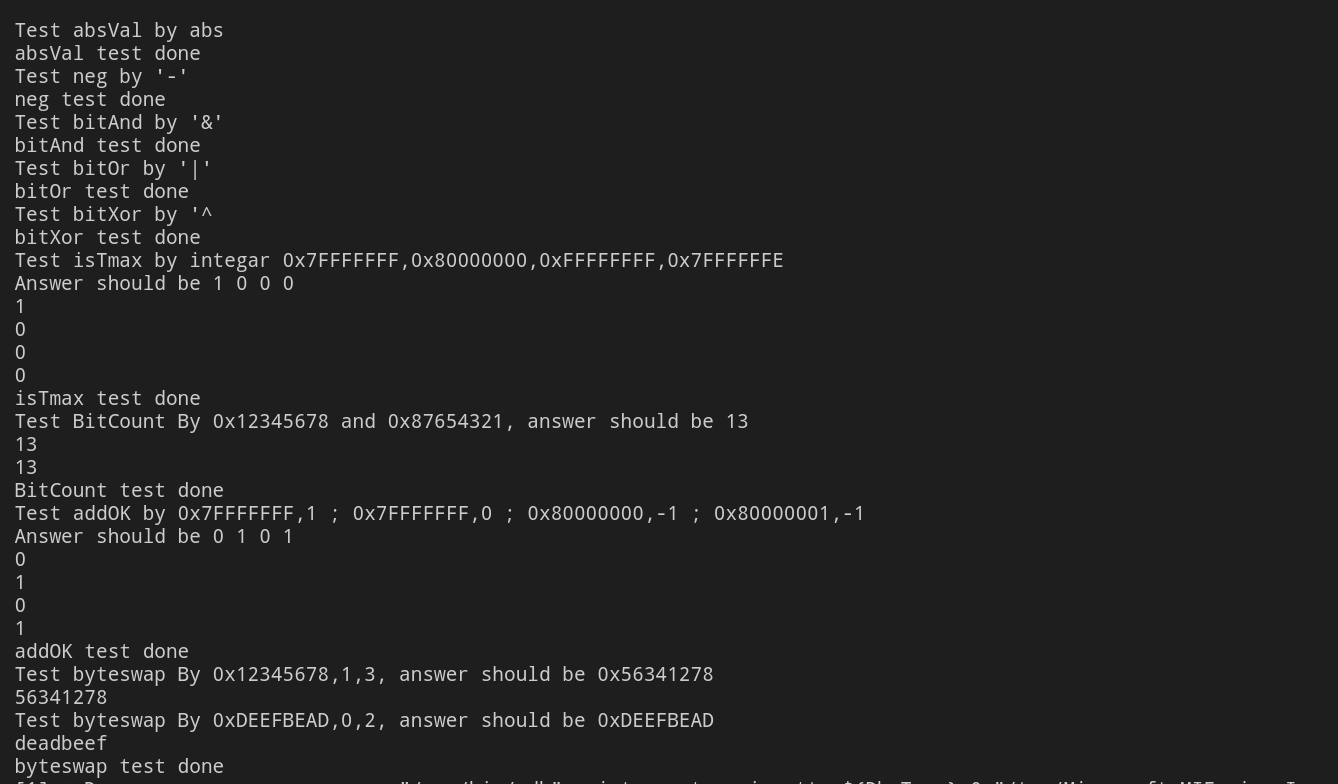


图8 测试结果

1. **体会**

通过任务1熟悉了不同数据类型在操作系统中存放形式，并了解了一系列压缩思想与方法

通过任务2 熟悉并练习了位运算，让系统运行更快捷，并且对思维很有帮助。但有的时候确实很难思考出思路

**五、源码**

实验任务 1、2 的源程序（单倍行距，5号宋体字）

任务1源程序

#include <iostream>

#include <cstring>

#include <iterator>

using namespace std;

typedef struct Student{

char name[15];

short age;

float score;

char remark[200];

} student;

const int N = 5;

const int M = 1e5;

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

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

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

inline void printfloat\_16(const float fnum){

unsigned int \*p = (unsigned int \*)&fnum;

for(int i = 31 ; i >= 0 ; i--){

cout << ((\*p >> i) & 1);

if(i == 31 || i == 23) cout << " ";

}

cout << endl;

}

int main(){

student old\_s[N];

student new\_s[N];

memset(old\_s,0,sizeof(old\_s));

memset(new\_s,0,sizeof(new\_s));

char buf[M];

old\_s[0] = {"Pu Shu", 19 , 22, "Best"};

old\_s[1] = {"Zhang San", 20, 23, "Good"};

old\_s[2] = {"Li Si", 21, 24, "Good"};

old\_s[3] = {"Wang Wu", 22, 25, "Good"};

old\_s[4] = {"Zhao Liu", 23, 26, "Good"};

int before\_cp = sizeof(old\_s);

int after\_cp = pack\_student\_bytebybyte(old\_s,2,buf);

after\_cp += pack\_student\_whole(old\_s+2,3,buf+after\_cp);

// PuShu Score 编码 41 B0 00 00 --> 0100 0001 1011 0000 0000 0000 0000 0000 --> 1.011... \* 2^(bin(10000011)-bin(01111111))

// 上式等价为 10110\_bin == 22\_dec 正确

// 打印第0个学生的16进制score编码

cout << "PuShu Score: ";

printfloat\_16(old\_s[0].score);

cout << "Before Compression: " << before\_cp << " bytes" << endl;

cout << "After Compression: " << after\_cp << " bytes" << endl;

restore\_student(buf,after\_cp,new\_s);

cout << std::hex;

for(int i = 0 ; i < 20 ; i ++){

cout << (static\_cast<short>(buf[i]) & 0xff) << " ";

}

cout << endl;

// 输出与观察一致

return 0;

}

/\*

存放规律

字符串 name 和 remark 按序存放，以'\0' 结尾

age，float 在内存中小端存放 低字节在前

\*/

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

int now\_add = 0;

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

char \* cp = s[i].name;

// 复制name

while(\*cp != '\0') buf[now\_add++] = \*cp++;

buf[now\_add++] = '\0';

// 复制age (小端存放)

cp = (char \*)&s[i].age;

for(int i = 0 ; i < 2 ; i++)

buf[now\_add++] = \*cp++;

// 复制float 四个字节

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

for(int i = 0 ; i < 4 ; i++)

buf[now\_add++] = \*cp++;

cp = s[i].remark;

while(\*cp != '\0') buf[now\_add++] = \*cp++;

buf[now\_add++] = '\0';

}

return now\_add;

}

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

int now\_add = 0;

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

strcpy(buf+now\_add,s[i].name);

now\_add += strlen(s[i].name)+1;

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

now\_add += 2;

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

now\_add += 4;

strcpy(buf+now\_add,s[i].remark);

now\_add += strlen(s[i].remark)+1;

}

return now\_add;

}

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

int now\_add = 0;

int i = 0;

while(1){

if (now\_add >= len) return;

strcpy(s[i].name,buf+now\_add);

now\_add += strlen(s[i].name)+1;

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

now\_add += 2;

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

now\_add += 4;

strcpy(s[i].remark,buf+now\_add);

now\_add += strlen(s[i].remark)+1;

i++;

}

}

任务2源程序

#include <iostream>

#include <cmath>

using namespace std;

int absVal(int x){

return (x+(x>>31)) ^ (x>>31);

}

int neg(int x){

return ~x + 1;

}

int bitAnd(int x,int y){

return ~(~x|~y);

}

int bitOr(int x,int y){

return ~((~x)&(~y));

}

int bitXor(int x, int y){

return ~(~(x&(~y)) & ~((~x)&y));

}

int isTmax(int x){

return !(x^~(x+1)) & !!(x+1); // 特判x == -1

}

int bitCount(int x){

// 将相邻的两位相加，结果存储在这两位中的低位

x = (x & 0x55555555) + ((x >> 1) & 0x55555555); // 0101

// 将相邻的四位相加，结果存储在这四位中的低两位

x = (x & 0x33333333) + ((x >> 2) & 0x33333333); // 0011

// 将相邻的八位相加，结果存储在这八位中的低四位

x = (x & 0x0F0F0F0F) + ((x >> 4) & 0x0F0F0F0F); // 0000 1111

// 将相邻的十六位相加，结果存储在这十六位中的低八位

x = (x & 0x00FF00FF) + ((x >> 8) & 0x00FF00FF); // 00000000 11111111

// 将相邻的三十二位相加，结果存储在这三十二位中的低十六位

x = (x & 0x0000FFFF) + ((x >> 16) & 0x0000FFFF); // 00000000 00000000 11111111 11111111

// 此时，x的低十六位就是原始输入x中1的个数

return x;

}

int bitMask(int highbit, int lowbit){

return ((~0) << lowbit) & ~((~0) << (highbit+1));

}

int addOK(int x,int y){

// return ((x>>31 & 1) ^ (y>>31 & 1)) | (((x>>31 & 1) & (y>>31 & 1)) ^ (x+y) >> 31 & 1;

return (x>>31 & 1) ^ (y>>31 & 1) | !((x>>31&1) & (y>>31&1) ^ (x+y)>>31&1);

}

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

int mask = 0x000000FF;

int get\_n = (mask << (n<<3)) & x;

int get\_m = (mask << (m << 3) ) & x;

return (~((mask << (n<<3)) | (mask << (m<<3))) & x) | (((get\_n >> (n<<3)) << (m<<3)) & (mask << (m<<3)) | ((get\_m >> (m<<3)) << (n<<3)) & (mask << (n<<3)));

}

int main(){

printf("Test absVal by abs\n");

for(int i = -100 ; i <= 100 ; i++){

if(absVal(i) != abs(i)){

printf("Error at %d\n",i);

}

}

printf("absVal test done\n");

printf("Test neg by \'-\'\n");

for(int i = -100 ; i <= 100 ; i++){

if(neg(i) != -i){

printf("Error at %d\n",i);

}

}

printf("neg test done\n");

printf("Test bitAnd by \'&\'\n");

for(int i = -100 ; i <= 100 ; i++){

for(int j = -100 ; j <= 100 ; j++){

if(bitAnd(i,j) != (i&j)){

printf("Error at %d %d\n",i,j);

}

}

}

printf("bitAnd test done\n");

printf("Test bitOr by \'|\'\n");

for(int i = -100 ; i <= 100 ; i++){

for(int j = -100 ; j <= 100 ; j++){

if(bitOr(i,j) != (i|j)){

printf("Error at %d %d\n",i,j);

}

}

}

printf("bitOr test done\n");

printf("Test bitXor by \'^\n");

for(int i = -100 ; i <= 100 ; i++){

for(int j = -100 ; j <= 100 ; j++){

if(bitXor(i,j) != (i^j)){

printf("Error at %d %d\n",i,j);

}

}

}

printf("bitXor test done\n");

printf("Test isTmax by integar 0x7FFFFFFF,0x80000000,0xFFFFFFFF,0x7FFFFFFE\n");

printf("Answer should be 1 0 0 0\n");

printf("%d\n",isTmax(0x7FFFFFFF));

printf("%d\n",isTmax(0x80000000));

printf("%d\n",isTmax(0xFFFFFFFF));

printf("%d\n",isTmax(0x7FFFFFFE));

printf("isTmax test done\n");

printf("Test BitCount By 0x12345678 and 0x87654321, answer should be 13\n");

printf("%d\n",bitCount(0x12345678));

printf("%d\n",bitCount(0x87654321));

printf("BitCount test done\n");

printf("Test addOK by 0x7FFFFFFF,1 ; 0x7FFFFFFF,0 ; 0x80000000,-1 ; 0x80000001,-1\n");

printf("Answer should be 0 1 0 1\n");

printf("%d\n",addOK(0x7FFFFFFF,1));

printf("%d\n",addOK(0x7FFFFFFF,0));

printf("%d\n",addOK(0x80000000,-1));

printf("%d\n",addOK(0x80000001,-1));

printf("addOK test done\n");

printf("Test byteswap By 0x12345678,1,3, answer should be 0x56341278\n");

printf("%x\n",byteswap(0x12345678,1,3));

printf("Test byteswap By 0xDEEFBEAD,0,2, answer should be 0xDEEFBEAD\n");

printf("%x\n",byteswap(0xDEEFBEAD,0,2));

printf("byteswap test done\n");

return 0;

}