

**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 ： 图灵2301**

**学 号 ： U202315064**

**姓 名 ： 刘铖**

**指导教师 ：**

**2024 年 9 月 26 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**

算法思想：

pack\_student\_bytebybyte 函数通过一个 char 类型指针扫描要压缩的数组，从而读取数据进行压缩。压缩过程中跳过字符数组的尾部未使用空间以及 short 对齐浪费的两字节空间，将数据紧密压缩在一起。

pack\_student\_whole 函数使用同样的思路，只是使用的是 strcpy 和 memcpy 进行压缩。

restore\_student 函数使用了一个子函数restore\_once，用以解压缩单个结构体单元。restore\_once 返回解压完单个结构体单元后的新指针，指向下一个要解压的数据，以便解压继续进行。

运行结果：

输入数据：

Liuch 19 64 L\_Star\_Plus's\_Classmate

nwlrb 20 78.600000 qbhcdarzow

kkyhi 20 62.900000 qscdxrjmow

frxsj 19 82.900000 ldbefsarcb

ynecd 19 97.000000 gxxpklorel

输出数据：  
4c 69 75 63 68 00 13 00 00 00 80 42 4c 5f 53 74 61 72 5f 50

name = Liuch, age = 19, score = 64.000000, remark = L\_Star\_Plus's\_Classmate

name = nwlrb, age = 20, score = 78.599998, remark = qbhcdarzow

name = kkyhi, age = 20, score = 62.900002, remark = qscdxrjmow

name = frxsj, age = 19, score = 82.900002, remark = ldbefsarcb

name = ynecd, age = 19, score = 97.000000, remark = gxxpklorel

（5）输出的值与调试时在内存窗口观察到的message的前20个字节一致。

（6）在内存中，score 是00 00 80 42。Score 的值是64。64 = 2 ^ (6)。故其对应的编码为 0 1000 0101 0000…… 转化为 16 进制为 42800000。由于 x86 平台是小端序，在内存中其排列顺序为 00 00 80 42，符合我们的观察。

（7）结构体的内存排布规律：编译器会把内存空间按照结构体中使用字节数最多的类型进行对齐(align），因此short 类型实际占用了 4 字节的空间，其中两字节不会进行使用。由于其余两个字符数组所占的字节数恰是 4 的倍数，故不会多占用空间。

所以结构体 student 的内存排布规律为 8（字符数组 name) + 4(2 个字节为 age 使用的空间 + 2 个字节用于对其) + 4（float 类型的 score） + 200（字符数组 remark）。

**（2）任务 2 的算法思想、运行结果等记录**

negate(int x) 使用位运算来实现取负。(~x) + 1 通过按位取反后加1得到负数，符合二进制补码表示。

absVal(int x) 判断 x 的符号位。如果是负数，则调用 negate(x) 返回其绝对值；否则直接返回 x。

bitAnd(int x, int y) 通过德摩根定律实现。x & y 可以用 ~((~x) | (~y)) 表示，使用按位取反和按位或操作。

bitOr(int x, int y) 同样使用德摩根定律，x | y 可用 ~((~x) & (~y)) 表示，通过取反和按位与操作实现。

bitXor(int x, int y) 先计算了 x，y都为1的位，再计算了其都为0的位，这些位异或结果为 0 ，其他位异或结果为 1，故把它们取 or 再取反即可。

isTmax(int x) 首先检查 x 是否为全1（即 ~x 非零），然后检查 x + 1 是否等于 -x，通过这些条件结合异或操作来判断 x 是否为最大正整数，可以证明不存在其他情况。

bitCount(int x) 通过倍缩的方式得到答案，每次分别得到相邻2位的1的个数，相邻 4 位的 1 的个数，直至这 32 位的 1 的个数，共用 20 次运算。

bitMask(int highbit, int lowbit) 首先判断 highbit 是否为31（即全1的情况），若是返回全1。否则，使用左移和减法生成高位的掩码，然后通过异或操作去掉低位的掩码。

addOK(int x, int y) 通过检查符号位来判断，若两个数同符号而结果符号不同，则说明产生溢出。计算中记录符号位并判断合并后的结果是否合理。

byteSwap(int x, int n, int m) 通过右移和按位与提取两个字节，然后使用异或操作交换位置，最后返回结果。

运算结果：

自测输出：

neg(1828504314) = -1828504314

abs(1828504314) = 1828504314

and(1828504314, -651724519) = 1210335256

or(1828504314, -651724519) = -33555461

xor(1828504314, -651724519) = -1243890717

isTmax(2147483647) = 1

isTmax(1828504314) = 0

bitCount(1828504314) = 19

bitMask(31, 9) = 11111111111111111111111000000000

addOK(1828504314, 651724519) = 1

byteSwap(6cfcc2fa, 0, 0) = 6cfcc2fa

neg(-23834866) = 23834866

abs(-23834866) = 23834866

and(-23834866, 1269981074) = 1250971394

or(-23834866, 1269981074) = -4825186

xor(-23834866, 1269981074) = -1255796580

isTmax(2147483647) = 1

isTmax(-23834866) = 0

bitCount(-23834866) = 18

bitMask(22, 1) = 00000000011111111111111111111110

addOK(23834866, 1269981074) = 0

byteSwap(fe944f0e, 1, 0) = fe940e4f

**四、体会**

在这次实验中，我学到了很多关于数据压缩与解压、位运算的知识。首先，在任务1中，通过编写数据压缩与解压函数，我更加深入理解了内存中的数据存储方式，尤其是在涉及结构体的情况下。不同数据类型在内存中的对齐方式给数据压缩带来了挑战，但通过实验，我理解了如何通过逐字节、整体拷贝等方式有效处理这些问题。同时，任务中的浮点数编码验证以及调试过程中观察内存内容的操作，帮助我加深了对计算机系统底层数据表示的理解。

任务2中，我通过使用位运算实现了一系列常见功能，如取绝对值、按位与、按位或等。特别是只能使用基本的位运算符号，如~、&、|，让我更加体会到位操作的灵活性和强大性。这种限制让我更加注重运算的效率以及算法的设计。同时，通过调试和验证结果，我能够确保这些算法的正确性，进而加深了对底层位运算逻辑的理解。

总的来说，这次实验不仅提升了我的编程技能，还让我更加了解了计算机系统的底层数据处理和表示方式。这为以后更深入的系统编程和优化打下了坚实的基础。

**五、源码**

实验任务 1、2 的源程序（单倍行距，5号宋体字）

**实验任务1：**

#include <stdio.h>

#include <string.h>

#include <stdbool.h>

typedef struct student {

char name[8];

short age;

float score;

char remark[200];

} student;

#define N 5

#define N1 2

#define N2 3

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

unsigned char message[1000010];

void input(student \*x)

{

scanf("%s%hd%f%s", x->name, &(x->age), &(x->score), x->remark);

}

void print(student \*x)

{

printf("name = %s, age = %hd, score = %f, remark = %s\n", x->name, x->age, x->score, x->remark);

}

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

{

char \*e = buf;

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

for (int j = 0; (s + i)->name[j]; ++j) {

\*e = (s + i)->name[j];

e += 1;

}

\*e = 0;

e += 1;

char \*tmp = (char\*)(s + i);

tmp += 8;

for (int j = 0; j < 2; ++j) {

\*e = \*tmp;

e += 1;

tmp += 1;

}

tmp += 2;

for (int j = 0; j < 4; ++j) {

\*e = \*tmp;

e += 1;

tmp += 1;

}

for (int j = 0; (s + i)->remark[j]; ++j) {

\*e = (s + i)->remark[j];

e += 1;

}

\*e = 0;

e += 1;

}

return e - buf;

}

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

{

char \*e = buf;

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

strcpy(e, (s + i)->name);

e += strlen((s + i)->name) + 1;

memcpy(e, &((s + i)->age), sizeof (short));

e += sizeof (short);

memcpy(e, &((s + i)->score), sizeof (float));

e += sizeof (float);

strcpy(e, (s + i)->remark);

e += strlen((s + i)->remark) + 1;

}

return e - buf;

}

char \*restore\_once(char \*buf, student \*s)

{

char \*e = buf;

int i = 0;

do {

s->name[i] = \*e;

e += 1;

i += 1;

} while(\*(e - 1));

char \*tmp = (char\*)s;

tmp += 8;

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

\*tmp = \*e;

e += 1;

tmp += 1;

}

tmp += 2;

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

\*tmp = \*e;

e += 1;

tmp += 1;

}

i = 0;

do {

s->remark[i] = \*e;

e += 1;

i += 1;

} while (\*(e - 1));

return e;

}

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

{

char \*end = buf + len;

int n = 0;

while (buf != end) {

buf = restore\_once(buf, s + n);

n += 1;

}

return n;

}

int main(void)

{

freopen("task1.in", "r", stdin);

for (int i = 0; i < 5; ++i) {

input(old\_s + i);

}

int n1 = pack\_student\_bytebybyte(old\_s, N1, message);

int n2 = pack\_student\_whole(old\_s + N1, N2, message + n1);

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

printf("%02x ", \*(message+ i));

}

putchar('\n');

restore\_student(message, n1, new\_s);

restore\_student(message + n1, n2, new\_s + N1);

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

print(new\_s + i);

}

return 0;

}

**实验任务2：**

#include <time.h>

#include <stdio.h>

#include <limits.h>

#include <assert.h>

#include <stdlib.h>

int negate(int x)

{

assert((~x) + 1 == -x);

return (~x) + 1;

}

int absVal(int x)

{

if (x >> 31 & 1)

return negate(x);

return x;

}

int bitAnd(int x, int y)

{

assert((x & y) == ~((~x) | (~y)));

return ~((~x) | (~y));

}

int bitOr(int x, int y)

{

assert((x | y) == ~((~x) & (~y)));

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

}

int bitXor(int x, int y)

{

int a = x & y;

int b = (~x) & (~y);

assert((x ^ y) == ~bitOr(a, b));

return ~bitOr(a, b);

}

int isTmax(int x)

{

int ret = (!!(~x))^(!!((x + 1) ^ (~x)));

assert(ret == (x == INT\_MAX));

return ret;

}

int bitCount(int x)

{

int res = 0;

for (int i = 0; i < 32; ++i) {

if (x >> i & 1) res += 1;

}

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

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

x = ((x >> 4) & 0x0f0f0f0f) + (x & 0x0f0f0f0f);

x = ((x >> 8) & 0x00ff00ff) + (x & 0x00ff00ff);

x = ((x >> 16) & 0x0000ffff) + (x & 0x0000ffff);

assert(res == x);

return x;

}

int bitMask(int highbit, int lowbit)

{

int full = !(highbit ^ 31) ? -1 : ((1 << (highbit + 1)) - 1);

return full ^ ((1 << lowbit) - 1);

}

int addOK(int x, int y)

{

int tot = 0;

tot += (x >> 31 & 1);

tot += (y >> 31 & 1);

if (!(tot ^ 1)) return 0;

int sum = x + y;

tot += (sum >> 31 & 1);

if (!(tot ^ 2) | !(tot ^ 1)) return 1;

return 0;

}

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

{

int a = (x >> (n << 3)) & ((1 << 8) - 1);

int b = (x >> (m << 3)) & ((1 << 8) - 1);

x ^= a << (n << 3);

x ^= b << (n << 3);

x ^= a << (m << 3);

x ^= b << (m << 3);

return x;

}

unsigned int myrand(void)

{

unsigned int ret = rand();

ret ^= (rand() & 1) << 31;

return ret;

}

void print2(int x)

{

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

putchar('0' + (x >> i & 1));

}

putchar('\n');

}

int main(void)

{

srand(time(NULL));

int T = 1;

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

int x = myrand();

int y = myrand();

printf("neg(%d) = %d\n", x, negate(x));

printf("abs(%d) = %d\n", x, absVal(x));

printf("and(%d, %d) = %d\n", x, y, bitAnd(x, y));

printf("or(%d, %d) = %d\n", x, y, bitOr(x, y));

printf("xor(%d, %d) = %d\n", x, y, bitXor(x, y));

printf("isTmax(%d) = %d\n", INT\_MAX, isTmax(INT\_MAX));

printf("isTmax(%d) = %d\n", x, isTmax(x));

printf("bitCount(%d) = %d\n", x, bitCount(x));

int l = rand() % 32, h = rand() % 32;

if (l > h) {

l ^= h;

h ^= l;

l ^= h;

}

printf("bitMask(%d, %d) = ", h, l);

print2(bitMask(h, l));

printf("addOK(%d, %d) = %d\n", abs(x), abs(y), addOK(abs(x), abs(y)));

int a = rand() % 4, b = rand() % 4;

printf("byteSwap(%x, %d, %d) = %x\n", x, a, b, byteSwap(x, a, b));

}

return 0;

}