

**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 ： U202311239**

**姓 名 ： 刘星佳**

**指导教师 ： 李海波**

**年 月 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**

**（2）任务 2 的算法思想、运行结果等记录**

**四、体会**

**五、源码**

实验任务 1、2 的源程序（单倍行距，5号宋体字）

**// task1.c**

**# include <stdio.h>**

**# include <string.h>**

**typedef struct student{**

**char name[8];**

**short age;**

**float score;**

**char remark[200];**

**}student;**

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

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

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

**# define N 5**

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

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

**char \*bufptr = buf;**

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

**for(int j = 0; s[i].name[j] != '\0'; j++) {**

**\*bufptr = s[i].name[j];**

**bufptr++;**

**}**

**\*bufptr = '\0'; bufptr++;**

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

**for(int j = 0; j < sizeof(s[i].age); j++) {**

**\*bufptr = \*temptr;**

**bufptr++; temptr++;**

**}**

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

**for(int j = 0; j < sizeof(s[i].score); j++) {**

**\*bufptr = \*temptr;**

**bufptr++; temptr++;**

**}**

**for(int j = 0; s[i].remark[j] != '\0'; j++) {**

**\*bufptr = s[i].remark[j];**

**bufptr++;**

**}**

**\*bufptr = '\0'; bufptr++;**

**}**

**return bufptr - buf;**

**}**

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

**char \*bufptr = buf;**

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

**strcpy(bufptr, s[i].name);**

**bufptr += strlen(s[i].name);**

**\*bufptr = '\0'; bufptr++;**

**memcpy(bufptr, (char \*)&s[i].age, sizeof(s[i].age));**

**bufptr += sizeof(s[i].age);**

**memcpy(bufptr, (char \*)&s[i].score, sizeof(s[i].score));**

**bufptr += sizeof(s[i].score);**

**strcpy(bufptr, s[i].remark);**

**bufptr += strlen(s[i].remark);**

**\*bufptr = '\0'; bufptr++;**

**}**

**return bufptr - buf;**

**}**

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

**char \*bufptr = buf;**

**student \*stuptr = s;**

**while(bufptr - buf < len) {**

**char \*temptr = stuptr->name;**

**while(\*bufptr != '\0') {**

**\*temptr = \*bufptr;**

**temptr++; bufptr++;**

**}**

**\*temptr = \*bufptr; bufptr++;**

**temptr = (char \*)&stuptr->age;**

**for(int i = 0; i < sizeof(short); i++) {**

**\*temptr = \*bufptr;**

**temptr++; bufptr++;**

**}**

**temptr = (char \*)&stuptr->score;**

**for(int i = 0; i < sizeof(float); i++) {**

**\*temptr = \*bufptr;**

**temptr++; bufptr++;**

**}**

**temptr = stuptr->remark;**

**while(\*bufptr != '\0') {**

**\*temptr = \*bufptr;**

**temptr++; bufptr++;**

**}**

**\*temptr = \*bufptr; bufptr++;**

**stuptr++;**

**}**

**return stuptr - s;**

**}**

**int main() {**

**char buf[1000];**

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

**scanf("%s%hd%f%s", old\_s[i].name, &old\_s[i].age, &old\_s[i].score, old\_s[i].remark);**

**}**

**printf("old\_s:\n");**

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

**printf("name: %s\t\tage: %hd\t\tscore: %f\t\tremark:%s\n", old\_s[i].name, old\_s[i].age, old\_s[i].score, old\_s[i].remark);**

**}**

**int buflen1 = pack\_student\_bytebybyte(old\_s, 2, buf);**

**int buflen2 = pack\_student\_whole(old\_s + 2, 3, buf + buflen1);**

**int stunum = restore\_student(buf, buflen1 + buflen2, new\_s);**

**printf("new\_s:\n");**

**for(int i = 0; i < stunum; i++) {**

**printf("name: %s\t\tage: %hd\t\tscore: %f\t\tremark:%s\n", new\_s[i].name, new\_s[i].age, new\_s[i].score, new\_s[i].remark);**

**}**

**return 0;**

**}**

**//task2.c**

**# include <stdio.h>**

**# include <stdlib.h>**

**# include <time.h>**

**# include <assert.h>**

**int absVal(int x) {**

**if(x >> 31) {**

**return (~x) + 1;**

**} else {**

**return x;**

**}**

**}**

**int negate(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) {**

**int v1 = bitAnd(~x, y), v2 = bitAnd(x, ~y);**

**return bitOr(v1, v2);**

**}**

**int isTmax(int x) {**

**return bitAnd(!((x + 1) ^ (~x)), !!(x + 1));**

**}**

**int bitCount(int xx) {**

**unsigned int x = xx;**

**unsigned int t1 = x & 0x55555555, t2 = x ^ t1;**

**x = t1 + (t2 >> 1);**

**t1 = x & 0x33333333; t2 = x ^ t1;**

**x = t1 + (t2 >> 2);**

**t1 = x & 0x0f0f0f0f; t2 = x ^ t1;**

**x = t1 + (t2 >> 4);**

**t1 = x & 0x00ff00ff; t2 = x ^ t1;**

**x = t1 + (t2 >> 8);**

**t1 = x & 0x0000ffff; t2 = x ^ t1;**

**x = t1 + (t2 >> 16);**

**return x;**

**}**

**int bitMask(int highbit, int lowbit) {**

**if(!(highbit ^ 31)) {**

**return 0xffffffff >> lowbit << lowbit;**

**} else {**

**return ((1u << highbit + 1) + 0xffffffff) >> lowbit << lowbit;**

**}**

**}**

**int addOK(int x, int y) {**

**int xh = (x >> 31) & 1, yh = (y >> 31) & 1, xo = x & 0x7fffffff, yo = y & 0x7fffffff, ad = ((xo + yo) >> 31) & 1;**

**if(xh & yh & (!ad) | (!xh) & (!yh) & ad) return 1;**

**else return 0;**

**}**

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

**int u = n << 3, v = m << 3;**

**int x1 = (x >> u) & 255, x2 = (x >> v) & 255;**

**return x ^ (x1 << u) ^ (x2 << v) ^ (x1 << v) ^ (x2 << u);**

**}**

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

**int netgate\_standard(int x) { return -x;}**

**int bitAnd\_standard(int x, int y) {return x & y;}**

**int bitOr\_standard(int x, int y) {return x | y;}**

**int bitXor\_standard(int x, int y) {return x ^ y;}**

**int isTmax\_standard(int x) {return x == 0x7fffffff;}**

**int bitCount\_standard(int x) {**

**int ans = 0;**

**for(int i = 0; i < 32; i++) ans += (x & 1), x >>= 1;**

**return ans;**

**}**

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

**unsigned int ans = 0;**

**for(int i = lowbit; i <= highbit; i++) ans |= 1u << i;**

**return ans;**

**}**

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

**long long xl = x, yl = y, sum = xl + yl;**

**if(sum > \_\_INT\_MAX\_\_ || sum < (int)0x80000000) return 1;**

**else return 0;**

**}**

**int byteSwap\_standard(int x, int n, int m) {**

**int tmp = x;**

**char \*p = (char \*)&tmp;**

**int a = \*(p + n);**

**int b = \*(p + m);**

**\*(p + n) = b;**

**\*(p + m) = a;**

**return tmp;**

**}**

**int myrand() {**

**int x = rand();**

**if(rand() % 2) x = -x;**

**return x;**

**}**

**int main() {**

**srand(time(0));**

**puts("abs\_test...");**

**for(int i = 0; i < 10; i++) {**

**int x = myrand();**

**assert(absVal(x) == absVal\_standard(x));**

**printf(" %d-th passed\n", i);**

**}**

**puts("abs\_test passed");**

**puts("negate\_test...");**

**for(int i = 0; i < 10; i++) {**

**int x = myrand();**

**assert(negate(x) == netgate\_standard(x));**

**printf(" %d-th passed\n", i);**

**}**

**puts("negate\_test passed");**

**puts("and\_test...");**

**for(int i = 0; i < 10; i++) {**

**int x = myrand(), y = myrand();**

**assert(bitAnd(x, y) == bitAnd\_standard(x, y));**

**printf(" %d-th passed\n", i);**

**}**

**puts("and\_test passed");**

**puts("or\_test...");**

**for(int i = 0; i < 10; i++) {**

**int x = myrand(), y = myrand();**

**assert(bitOr(x, y) == bitOr\_standard(x, y));**

**printf(" %d-th passed\n", i);**

**}**

**puts("or\_test passed");**

**puts("xor\_test...");**

**for(int i = 0; i < 10; i++) {**

**int x = myrand(), y = myrand();**

**assert(bitXor(x, y) == bitXor\_standard(x, y));**

**printf(" %d-th passed\n", i);**

**}**

**puts("xor\_test passed");**

**puts("tmax\_test...");**

**for(int i = 0; i < 10; i++) {**

**int x;**

**if(i == 0) x = 0x7fffffff;**

**else if(i == 1) x = -1;**

**else x = myrand();**

**assert(isTmax(x) == isTmax\_standard(x));**

**printf(" %d-th passed\n", i);**

**}**

**puts("tmax\_test passed");**

**puts("bitcount\_test...");**

**for(int i = 0; i < 10; i++) {**

**int x = myrand();**

**assert(bitCount(x) == bitCount\_standard(x));**

**printf(" %d-th passed\n", i);**

**}**

**puts("bitcount\_test passed");**

**puts("bitmask\_test...");**

**for(int i = 0; i < 10; i++) {**

**int x = rand() % 32, y = rand() % 32;**

**if(x < y) {**

**int t;**

**t = x; x = y; y = t;**

**}**

**if(i < 3) x = 31;**

**assert(bitMask(x, y) == bitMask\_standard(x, y));**

**printf(" %d-th passed\n", i);**

**}**

**puts("bitmask\_test passed");**

**puts("addok\_test...");**

**for(int i = 0; i < 10; i++) {**

**int x = myrand(), y = myrand();**

**assert(addOK(x, y) == addOK\_standard(x, y));**

**printf(" %d-th passed\n", i);**

**}**

**puts("addok\_test passed");**

**puts("byteswap\_test...");**

**for(int i = 0; i < 10; i++) {**

**int x = myrand(), n = rand() % 4, m = rand() % 4;**

**assert(byteSwap(x, n, m) == byteSwap\_standard(x, n, m));**

**printf(" %d-th passed\n", i);**

**}**

**puts("byteswap\_test passed");**

**return 0;**

**}**