程序设计分组训练 实验 2 实验报告

专　　业：计算机科学与技术

班　　级：_____

学生姓名：_____

学　　号：_____

北京交通大学计算机与信息技术学院

2021 年 10 月 01 日

目录

1.相关知识准备

1.1 rand 函数

rand 函数产生的是伪随机数，也就是说它不是一个真实的随机数。伪随机数是怎么实现的原理大概如下：如果约定：a1=f(seed),an+1=f(an)a1=f(seed),an+1=f(an)那你可以行到一个序列：a1,a2,a3...an，那么要制作一个伪随机函数 rand，只需要让它每调用一次就返回序列的下一个元素就行。就相当于第 1 次调用 rand 返回 a1，第 2 次返回 a2，…，第 n 次返回 an，这样每次调 rand 都能拿到一个不同的数，只要整个序列的规律不明显，整个函数看起来就是随机的。现在计算机上的 rand 函数都是用这样的原理实现的，这里的 seed 被称为“随机数种子”。但这里有一个问题，如果 seed 不变，那我们每次调用 rand 函数获取的序列都是相同的。这就会造成有的程序跑一遍退出后，再重新跑一遍，两次的输出结果是相同的。所以我们还需要一个接口去设置 seed 值，这个接口就是 srand 函数。

计算机启动时就设定好了种子值，如果不调用 srand 函数重新设定的话，在同一次程序运行中，产生的随机数是不同的，但不同次运行中，结果完全一样。因此，需要使用 srand 函数来重设种子。

1.2 srand 函数

srand()用来设置 rand()产生随机数时的随机数种子。在调用

rand()函数产生随机数前，必须先利用 srand()设好随机数种子（seed），如果未设随机数种子，rand()在调用时会自动设随机数种子为 1。srand()函数定义 ： void srand (unsigned int seed);通常可以利用 geypid()或 time(0)的返回值来当做 seed；如果你用 time(0)的话,要加入头文件#include<time.h>。即 srand(time(0))。

1.3 命令行参数

main 函数参数可以由命令行输入，这包括两个参数：int argc 和 char *argv[]。其中，前者代表程序启动时，命令行参数的个数，包括程序名本身，则 argc 至少为 1。后者是指针数组，每一个指针指向一个字符串。

在使用命令行的时候可以用 change disk 命令改变盘符位置再输入相应的程序名和相关参数进行操作。

相关命令行参数博客称熟练使用命令行有助于提升效率和可移植性。

2.实验操作

2.1 atoi 函数

在命令行参数中的均为字符串形式,则需要将其中代表数组数的字符串改成整形。

而 int atoi（const char *str）可以把字符串转换为一个整数，其机制是遍历字符串把每个元素转为相应的数再结合权位最后得出相应的整数。

2.2 判断命令行参数个数并把参数默认初始化

在操作中，命令行可能得到的是只有三元组数目或文件名的参数，这是需要对参数进行判定,然后对命令行没有指定的部分进行默认化。这里对 argv[1]

判定，思路是遍历其中每一个元素，判断其 ASCII 码是否在‘0’和‘9’之间。代码如下：

```
01:            length = strlen(argv[1]);
02:            for(k=1;k<=length;k++){
03:            //判断是否是随机数个数
04:                if(argv[1][k]>='0'&&argv[1][k]<='9') continue;
05:                else break;
06:            }
07:            if(k==length){
08:                number = atoi(argv[1]);
09:                strcpy(txtaddress, "test2.txt");
10:            }
11:            else{
12:                strcpy(txtaddress, argv[1]);
13:                number = rand();
14:            }
```

2.3 写入 txt 文本文件并生成随机数

设置 FLIE*pf 以‘w’模式打开一个 txt 文件，遍历结构体写入文件中，然后关闭。随机数以 rand 函数生成。代码如下。

```
15:void Fileopreate(int number, char txtaddress[]){
16:// 生成文件指针 以w模式打开
17:    FILE*pf = fopen(txtaddress, "w");
18:    int(*array)[3] = (int(*)[3])malloc(sizeof(int)*number*3);
19:
20:
21:    for(int i=0; i<number; i++ ){
22:                for(int j=0; j<3; j++){
23:                    array[i][j] =rand()%100;
24:                }
25:    }
26://空指针的操作
27:    if(pf==NULL){
28:        printf("open error\n");
29:        exit(0);
30:    }
```

```
31:    fprintf(pf, "%d\n", number);
32:    for(int i=0; i<number; i++){
33:        for(int j=0; j<3; j++){
34:            fprintf(pf,"%d, ", array[i][j]);
35:        }
36:        fprintf(pf, "\n");
37:    }
38:    if(fclose(pf)!=0){
39:        printf("close error\n");
40:    }
41:    //标记性语句，便于调试
42:    printf("opreate success!\n");
43:    free(array);
44:}
```

2.4 三种数据结构的实现

（1） 指针数组

利用 malloc 函数动态申请内存。

```
   01:int(*array)[3] = (int(*)[3])malloc(sizeof(int)*number*3);
02:……
03:free(array);
```

（2） 结构体

```
04:void Fileopreate2(int number, char txtaddress[]){
05:    FILE*pf = fopen(txtaddress, "w");
06:    struct array2* array[] = (array2*)malloc(sizeof(array2)*number);
07:
08:    for(int i=0; i<number; i++ ){
09:        array[i]->a =rand()%100;
10:        array[i]->b =rand()%100;
11:        array[i]->c =rand()%100;
12:    }
13:    if(pf==NULL){
14:        printf("open error\n");
15:        exit(0);
16:    }
17:    fprintf(pf, "%d\n", number);
```

```
18:    for(int i=0; i<number; i++){
19:        fprintf(pf,"%d, %d, %d\n", array[i]->a,array[i]->b,array[i]->c);
20:        }
21:        fprintf(pf, "\n");
22:
23:    if(fclose(pf)!=0){
24:        printf("close error\n");
25:    }
26:
27:    printf("opreate success!\n");
28:    free(array);
       }
```

（3） 一维数组模拟二维数组

```
01:void Fileopreate3(int number, char txtaddress[]){
02:    FILE*pf = fopen(txtaddress, "w");
03:    int *array=(int*)malloc(sizeof(int)*number*3);
04:    for(int i=0; i<number; i++ ){
05:        for(int j=0;j<3; j++){
06:            array[i*3+j]= rand%100;
07:        }
08:
09:    if(pf==NULL){
10:        printf("open error\n");
11:        exit(0);
12:    }
13:    fprintf(pf, "%d\n", number);
14:    for(int i=0; i<number; i++){
15:        for(int j=0; j<3; j++){
16:            fprintf(pf,"%d, ", array[i*3+j);
17:        }
18:        fprintf(pf, "\n");
19:    }
20:    if(fclose(pf)!=0){
21:        printf("close error\n");
22:    }
23:
24:    printf("opreate success!\n");
25:    free(array);
26:}
27:
28:}
```

3.程序健壮性

3.1 健壮性的概念

健壮性是指软件对于规范要求以外的输入情况的处理能力。

所谓健壮的系统是指对于规范要求以外的输入能够判断出这个输入不符合规范要求，并能有合理的处理方式。另外健壮性有时也和容错性，可移植性，正确性有交叉的地方。比如，一个软件可以从错误的输入推断出正确合理的输入，这属于容错性量度标准，但是也可以认为这个软件是健壮的。一个软件可以正确地运行在不同环境下，则认为软件可移植性高，也可以叫，软件在不同平台下是健壮的。一个软件能够检测自己内部的设计或者编码错误，并得到正确的执行结果，这是软件的正确性标准，但是也可以说，软件有内部的保护机制，是模块级健壮的。软件健壮性是一个比较模糊的概念，但是却是非常重要的软件外部量度标准。软件设计的健壮与否直接反应了分析设计和编码人员的水平。即所谓的高手写的程序不容易死。

3.2 本程序健壮性分析

本程序只考虑到用户输入三个参数的情况，对于其他情况没有考虑。其实可以再 `switch-case` 中加入 `default` 进行处理。再比如，除第一个参数是程序名之外，后两个参数的位置可以考虑乱序任有效，即无论用户先输入数组数还是文件名都可以通过内部的判断得到正确处理。

之前对健壮性接触比较少，现在逐步有了用户的思维，用户的可能性是无穷无尽的，程序编写人员要提前把相应、可能的不符合要求的数据想到，进而返回错误类型，帮助用户修改相应数据，甚至判断用户真实意——这些都是应有之意。

4.心得体会

4.1 自身收获

本次实验中，对于 rand 函数、srand 函数，malloc 和 free 函数有了更深的理解。对于数组做函数的参数和函数返回多个数值的情况有了更深的理解，在这种情况下，通过指针或指针数组显然是事半功倍的。

最令我感到高兴的是，这次对于实验报告的编写有了更多的体会。采用论文相关格式制定好的样式有利于排版和格式统一，一举改变了此前先码字再排版的情况。

4.2 小组互评

张开元同学的程序完成的很好，特别是“程序健壮性”部分着实令我大开眼界。之前本人的程序仅仅是考虑到自己使用的情况，对极端情况处理比较薄弱，现在逐步建立起用户思维，开始剔除不合法的用户输入参数，这是一种进步。

王麒越同学的实验报告版式十分优秀，据悉，他也使用了“样式”进行快速编辑，这相比之前的报告是一种长足的进步。但和本人一样，其“程序健壮性”部分的论述也有待加强。

在小组的合作学习中进步是一种奇妙的体验，特别是和比自己优秀的人接触更是进步的终南捷径。