2016 程序设计方法课程综合训练 实验报告

目录

正文

实验一：实现插入排序算法

所谓插入排序算法是指：以任意顺序读入20个数据。将读入的第一数据放入数组的第一个元素位置，以后读入的数据与已存入数组中的数据比较，确定它在从小到大的排列中的位置，将该位置及其后的元素向后移动一个位置，将新读入的数据填入空出的位置中。这样在数组中的数据总是从小到大的顺序排列，20个数据处理完后输出数组中数据。

> zccz14: 什么？插入排序还要求数据量的嘛？(突然笑死.jpg)

> zccz14: 题意：实现与输入耦合的插入排序算法，升序，数据规模$n = 20$。
>
> 什么鬼，这还要跟输入(`stdin`)耦合。

数据结构

线性表

算法描述

> zccz14: ……一时语塞.jpg

1. 如果还有数没读入，读入一个数，捏在手上(newComing)，否则结束算法。
2. 从已排好序的数组中从后往前扫描……
3. 如果手上的数比数组里那个数小，将其往后挪一位；
4. 否则，表明已经找到要插入的位置（并且之后的元素已经被腾挪妥当），插入之，跳到1。

实现

```c
#include <stdio.h>
#define N 20
int data[N];
#define R(x, y) ((x) < (y))
int main() {
  scanf("%d", data);
  for (int i = 1; i < N; i++) {
    int newComing;
    scanf("%d", &newComing);
    int j;
    for (j = i - 1; j >= 0 && R(newComing, data[j]); j--) {
      data[j + 1] = data[j];
    }
    data[j + 1] = newComing;
  }
  for (int i = 0; i < N; i++) {
    printf("%d ", data[i]);
  }
  return 0;
}
```

运行结果

对于样例：

```
6
3
2
3
4
45
-1
-23
0
3
1
23336
666
-23336666
233
1234567
2233
1111
1121
0408
```

程序输入：

```
-23336666 -23 -1 0 1 2 3 3 3 4 6 45 233 408 666 1111 1121 2233 23336 1234567
```

> zccz14: 第一个实验就这样吧。

实验二：实现数字旋转方阵

根据输入数据$N(4 \leq N \leq 15)$，编程输出如所下图示的螺旋方阵。要求：

1. 根据输入数据，编程输出螺旋方阵。
2. 根据用户的选择，演示螺旋方阵的生成过程。

> zccz14: 你居然还要选择？？？生成过程？？？给你看单步调试可以嘛？"I am angry.jpg"

$$\begin{pmatrix} 1 & 20 & 19 & 18 & 17 & 16 \\ 2 & 21 & 32 & 31 & 30 & 15 \\ 3 & 22 & 33 & 36 & 29 & 14 \\ 4 & 23 & 34 & 35 & 28 & 13 \\ 5 & 24 & 25 & 26 & 27 & 12 \\ 6 & 7 & 8 & 9 & 10 & 11 \end{pmatrix}$$

> 这个题其实简洁明快的，用一个方向、越界判断加计步器就OK。
>
> 等下我抢个红包……看到口令不想抢了

数据结构

二维线性表

算法描述

1. 从一个起点开始(行 r := -1, 列c := 0), 初始方向d := 0(向下)，初始计步器 i := 0。
2. 计算下个点的位置 (newR := r + dr[d], newC := c + dc[d])。
3. 如果 (newR, newC) 没有越界，并且这个点之前没有访问过，标记步数 i + 1，并且步数自增1。
4. 如果步数达到矩阵规模 n 的平方，结束算法。
5. 否则，换下一个方向，跳到步骤2。

实现

```c
#include <stdio.h>
#include <string.h>
#define MAXN 15
// the result matrix (allocate only)
int Mat[MAXN][MAXN];
int n;
// constant direction array
// 'dr' means delta row
// 'dc' means delta column
// here is 'down', 'right', 'up', 'left'
int dr[] = {1, 0, -1, 0};
int dc[] = {0, 1, 0, -1};

char inputBuffer[10];

int isInRange(int r, int c) { return r >= 0 && c >= 0 && r < n && c < n; }

void print() {
  for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
      printf("%3d ", Mat[i][j]);
    }
    printf("\n");
  }
  fflush(stdout);
}

int main() {
  while (scanf("%d", &n) == 1) {
    if (!(4 <= n && n <= 15)) {
      printf("illegal input!\n");
      break;
    }
    int isInDetail = 0;
    printf("Show Detail? (Y for Yes, N for No) (default no): \n");
    fflush(stdout);
    scanf("%s", inputBuffer);
    isInDetail = inputBuffer[0] == 'Y';
    int r = -1, c = 0;     // start point
    int d = 0;             // initialize direction
    int totalStep = n * n; // total step counter
    int i = 0;
    while (i < totalStep) {
      int newR = r + dr[d], newC = c + dc[d];
      if (isInRange(newR, newC) && !Mat[newR][newC]) {
        // if (newR, newC) is in range, and there has been not visited yet,
        // mark counter here.
        Mat[newR][newC] = ++i;
        // update position
        r = newR;
```

```
        c = newC;
      } else {
        // next direction
        d = (d + 1) % 4;
      }
      if (isInDetail) {
        printf("step #%d:\n", i);
        print();
        printf("\n");
      }
    }
    // print the final matrix
    if (!isInDetail) {
      print();
    }
    memset(Mat, 0, sizeof(Mat));
  }
  return 0;
}
```

运行结果

对于输入：

```
6
N
```

程序输出：

```
Show Detail? (Y for Yes, N for No) (default no):
  1  20  19  18  17  16
  2  21  32  31  30  15
  3  22  33  36  29  14
  4  23  34  35  28  13
  5  24  25  26  27  12
  6   7   8   9  10  11
```

> 好了，吃饭去吧...

实验三：关于建立链表的实验

编写程序。要求：

1. 建立两个有序整数链表；
2. 寻找在两个有序整数链表中均出现的第一个相同整数，并将其输出。

> zccz14: 思路与插排略像，要保持链表中的元素有序。不同的是，线性表里将一堆元素向后平移所需的时间是 O(n)，而链表里则只需要O(1)常数时间即可完成，但这并没有什么卵用，因为遍历链表需要 O(n)的时间。

数据结构

有序单链表 (Ordered Single Linked List)

链表是递归定义的，即链表的某个节点也是一个链表。

只要是递归定义的数据结构，在其声明时，就可以同时声明节点(Node)与容器(Container)。

如：

```
struct List {
  int val;
  struct List *next;
};
```

如果这个数据结构不是递归定义的，如一个红黑树：

> 红黑树的子树并不是一颗红黑树。

```
struct RBTreeNode {
  int val;
  int color;
  struct RBTreeNode *left, *right;
};
struct RBTree {
  struct RBTreeNode *root;
};
```

那么分离节点(`RBTreeNode`)与容器(`RBTree`)的声明就是一个比较好的做法。

算法描述

插入节点

1. 构造新的节点 newNode
2. 如果这是一个空链表，则将链表的头指向 newNode，退出。
3. 如果链表的第一个元素的值大于等于 newNode 的值，将 newNode 插到链表头，退出。
4. 遍历链表，找到第一个节点cur，节点的值小于newNode的值，但下一个节点的值大于等于 newNode 的值，或者没有下一个节点了
5. 将newNode 插入到 cur 之后，退出。

查找第一个相同的元素

1. 设置两个指针a, b指向两个链表的头部。
2. 如果 a指向的值 大于 b指向的值，则将 b 移到下一个节点；
3. 如果 a指向的值 小于 b指向的值，则将 a 移到下一个节点；
4. 如果 a指向的值 等于 b指向的值，返回这个值，结束。
5. 跳到步骤2。

实现

```c
#include <stdio.h>
#include <stdlib.h>


struct List{
  int val;
  struct List *next;
};

// insertion
void insert(struct List **list, int value) {
  // construct the new node
  struct List *newNode = (struct List*)malloc(sizeof(struct List));
  newNode->val = value;
  newNode->next = 0;

  if (*list == 0) {
    // if the list is a empty list
    // insert directly
    *list = newNode;
    return;
  }
  if ((*list)->val >= value) {
    // insert before head
    newNode->next = *list;
    *list = newNode;
    return;
  }
  // travel the list
  struct List *cur; // mark current node
  for (cur = *list; cur; cur = cur->next) {
    if (cur->next) {
      // if cur has next
      if (cur->val < value && cur->next->val >= value) {
        // insert newNode after cur
        newNode->next = cur->next;
        cur->next = newNode;
        return;
      }
    } else {
      // if cur has not next node
      // let newNode to be cur's next
      cur->next = newNode;
      return;
    }
  }
}

int findIntersectionFirst(struct List *a, struct List *b, int *ret) {
  // just run alternatively
```

```
  while (a && b) {
    if (a->val > b->val) {
      b = b->next;
    } else if (a->val < b->val) {
      a = a->next;
    } else {
      *ret = a->val;
      return 1; // find, the result stored in ret
    }
  }
  return 0; // not found
}


struct List *list1, *list2;
int main() {
  insert(&list1, 5);
  insert(&list1, 5);
  insert(&list1, 3);
  // list1: 3 -> 5 -> 5
  insert(&list2, 2);
  insert(&list2, 4);
  insert(&list2, 5);
  insert(&list2, 3);
  // list2: 2 -> 3 -> 4 -> 5
  int ret;
  if (findIntersectionFirst(list1, list2, &ret) == 1) {
    printf("the first same element is %d\n", ret);
  } else {
    printf("not found same element\n");
  }
  return 0;
}
```

运行结果

```
the first same element is 3
```

> 贼强，让我们继续写第四个吧。

实验四：完成某学期的学生的成绩管理

编写程序，完成本班学生学系成绩管理，具体要求如下：

1. 输入一个班 n 名学生 ($n \leq 30$)，m 门课程 ($m \leq 8$) 的成绩，做成文件 `score.in`， 要求有良好的输入界面(Excuse Me.jpg)；
2. `score.in` 中读入该班成绩，计算出每位学生总成绩、平均成绩；
3. 按总成绩降序排列；
4. 根据学号或姓名查找某学生成绩；

5. 在屏幕上设计表格，显示该班成绩单（包括单科成绩、总成绩、平均成绩）；
6. 将经过排序的成绩及总成绩、平均成绩输出到文件 `t.out` 。

> zccz14: 区区一个C程大作业，竟然有这么麻烦的题目。
>
> 吃惊.jpg 啊不想写……

> 良好的输入界面请读者自行解决www

数据结构

二维线性表 & 结构体

算法描述

大暴力、大模拟

实现

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAXN 30
#define MAXM 8
#define STUIDLEN 10
#define STUNAMELEN 20

struct Student {
  char id[STUIDLEN + 1];
  char name[STUNAMELEN + 1];
  float grade[MAXM];
  int gradeCnt;
  //
  float totGrade;
  float avgGrade;
};

struct Class {
  struct Student students[MAXN];
  int studentCnt;
};

void loadFromFile(FILE *file, struct Class *class) {
  fscanf(file, "%d", &(class->studentCnt));
  int i, j;
  for (i = 0; i < class->studentCnt; i++) {
    struct Student *stu = &(class->students[i]);
    stu->totGrade = 0;
    fscanf(file, "%s%s%d", stu->id, stu->name, &(stu->gradeCnt));
    for (j = 0; j < stu->gradeCnt; j++) {
      fscanf(file, "%f", &(stu->grade[j]));
      stu->totGrade += stu->grade[j];
    }
    stu->avgGrade = stu->totGrade / stu->gradeCnt;
  }
}

void saveToFile(FILE *file, struct Class *class) {
  fprintf(file, "%d\n", class->studentCnt);
  int i, j;
  for (i = 0; i < class->studentCnt; i++) {
    struct Student *stu = &(class->students[i]);
    fprintf(file, "%s\t%s\t%d\t", stu->id, stu->name, stu->gradeCnt);
    for (j = 0; j < stu->gradeCnt; j++) {
      fprintf(file, "%.2f\t", stu->grade[j]);
    }
    fprintf(file, "\n");
  }
```

```
}

int cmp(const void *a, const void *b) {
  return (((struct Student*)b)->totGrade - ((struct Student*)a)->totGrade);
}

void sort(struct Class *class) {
  qsort(class->students, class->studentCnt, sizeof(struct Student), cmp);
}

void printStu(FILE *file, struct Student *stu) {
  if (stu) {
    fprintf(file, "%s\t%s\t%.2f\t%.2f\t", stu->id, stu->name, stu->totGrade, stu-
>avgGrade);
    int j;
    for (j = 0; j < stu->gradeCnt; j++) {
      fprintf(file, "%.2f\t", stu->grade[j]);
    }
    fprintf(file, "\n");
  } else {
    fprintf(file, "Student Not Available\n");
  }
}

void print(FILE *file, struct Class *class, char *displayName) {
  fprintf(file, "class %s has %d student(s)\n", displayName, class->studentCnt);
  int i;
  fprintf(file, "Student ID\tName\tTotal\tAverage\tGrades\n");
  for (i = 0; i < class->studentCnt; i++) {
    printStu(file, &(class->students[i]));
  }
}

struct Student *search(struct Class *class, char *keyword) {
  for (int i = 0; i < class->studentCnt; i++) {
    struct Student *stu = &(class->students[i]);
    if (strcmp(keyword, stu->id) == 0 || strcmp(keyword, stu->name) == 0) {
      return stu;
    }
  }
  return 0;
}

struct Class A;

int main() {
  FILE *file;
  file = fopen("score.in", "r");
  loadFromFile(file, &A);

  fclose(file);
```

```
    // sort in desending order by total grade
    sort(&A);
    // print class
    print(stdout, &A, "SE 42");
    // search student by Stu.ID or Name
    printStu(stdout, search(&A, "CZ"));
    printStu(stdout, search(&A, "2141601025"));
    printStu(stdout, search(&A, "2161601001"));
    // save to t.out
    file = fopen("t.out", "w");
    print(file, &A, "SE 42");
    fclose(file);

    return 0;
}
```

运行结果

在环境中具有 `score.in` 如下时：

```
4
2141601025      GZP     1       60.00
2141601026      CZ      6       59.00   59.00   59.00   59.00   59.00   59.00
2141601027      GJL     8       100.00  100.00  100.00  100.00  100.00  100.00  100.00
100.00
2141601030      OYPC    1       100.00
```

程序输出：

```
class SE 42 has 4 student(s)
Student ID      Name    Total   Average Grades
2141601027      GJL     800.00  100.00  100.00  100.00  100.00  100.00  100.00  100.00
100.00  100.00
2141601026      CZ      354.00  59.00   59.00   59.00   59.00   59.00   59.00   59.00
2141601030      OYPC    100.00  100.00  100.00
2141601025      GZP     60.00   60.00   60.00
2141601026      CZ      354.00  59.00   59.00   59.00   59.00   59.00   59.00   59.00
2141601025      GZP     60.00   60.00   60.00
Student Not Available
```

并产生文件：

```
class SE 42 has 4 student(s)
Student ID      Name    Total   Average Grades
2141601027      GJL     800.00  100.00  100.00  100.00  100.00  100.00  100.00  100.00
100.00  100.00
2141601026      CZ      354.00  59.00   59.00   59.00   59.00   59.00   59.00   59.00
2141601030      OYPC    100.00  100.00  100.00
2141601025      GZP     60.00   60.00   60.00
```

> 室友叫我去打乒乓球了。下次再写最后一题。

实验五：完成字符串压缩功能

编写程序，要求如下：

1. 输入一串字符串；
2. 将不连续相同的单个字符复制到压缩字符串；
3. 将一组连续相同的字符（不超过10个）变换成一个用于表示重复次数的数字字符和一个重复出现的字符到压缩字符串；
4. 将压缩了的字符串输出。

> zccz14: 这个就是游程编码(run-time code)，是一个编码变换而已。
>
> 什么都不说[1]，听说能获得更长的寿命，+1s。

数据结构

线性表

算法描述

1. 读入一个字符串S
2. 初始化 最后一个字符 lastCh := 0, 计数器 cnt := 0, 当前压缩字符串的长度 curC := 0，扫描字符串S的指针 i := 0。
3. 如果S[i] 与上次的字符 lastCh 相同，则计数器自增1；
4. 否则开始结算上个连续字符串
    1. 如果 计数器 大于 1，向压缩字符串中推送计数器的值；
    2. 如果 计数器 大于 0，向压缩字符串中推送上个字符的值；
    3. 更新 lastCh := S[i]，并置计数器 cnt := 1。
5. 如果S[i] 是字符串结束符 '\0'，跳出循环，跳到步骤7
6. 计数器自增1，到下一个字符。
7. 将压缩字符串的结尾置为 '\0'，结束。

实现

```c
#include <stdio.h>
#define MAXS 10
char S[MAXS + 1];
char C[MAXS + 1];

int main() {
  // input string
  while (scanf("%s", S) == 1) {
    // status
    int i = 0, cnt = 0;
    int curC = 0;
    char lastCh = 0;
    // travel the string
    do {
      if (S[i] == lastCh) {
        cnt++;
      } else {
        if (cnt > 0) {
          if (cnt > 1) {
            C[curC++] = cnt + '0';
          }
          // copy the character to C
          C[curC++] = lastCh;
        }
        // update last character and set counter = 1
        lastCh = S[i];
        cnt = 1;
      }
    } while (S[i++]);
    C[curC++] = 0; // end the string
    printf("%s\n", C);
  }
  return 0;
}
```

运行结果

对于输入：

```
AAABBB
AABBC
aaaaaaaabb
abcdeABCDE
```

程序输出：

```
3A3B
2A2BC
8a2b
abcdeABCDE
```

实验总结

嗨呀，好气啊。

代码参照 https://github.com/zccz14/CProgramming2016

意见：语塞.jpg

建议：你们这是自寻死路.jpg

集中实践环节的实施办法：想办法干它一炮.jpg

致谢词

感谢耐心看完直播的朋友们。

参考文献

- 《蛤三篇·怒斥记者》

---

1. 出自《怒斥记者》↩