

# 北京交通大学

## 《数据结构（A）》第3章基本作业

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 20 日



## 《数据结构 (A)》第 3 章作业<sup>①</sup>

提醒同学：如果使用 C++ 环境，则应该创建一个 project 来存放你的（这次作业）程序。如果数据结构定义与已实现的 project 相同，则一定的在已有的 project 之下实现，并要求：

- （1）单独书写新的一个 .h 文件；
- （2）一个题目程序书写成一个新的 .cpp 文件；
- （3）修改（也可以认为是）main 文件。

如果没有定义好数据结构，则要求重新建立一个新的 project，以上 3 个文件都需要自己写。

学生作业仅提交 .docx 文件：包括题目、思路、代码与注释、以及测试情况及其分析等（格式按模板要求）。

### 1 基本作业题目

3.1 （《数据结构题集（C 语言版）》，第 3 章，第 3.9 题，难度系数为 3）  
试将下列非递归过程改写为递归过程。

```
void test(int n){  
    int i;  
    i=n;  
    while (i>1)  
        Printf(i--);  
}
```

3.2 （《数据结构题集（C 语言版）》，第 3 章，第 24 页，第 3.19 题，难

---

<sup>①</sup> 这是《数据结构 (A)》第 3 章的基本作业，学生提交的截止日期是 2021 年 10 月 24 日。

度系数为 4) 假设一个算术表达式中可以包含 3 种括号：园括号“(”和“)”、方括号“[”和“]”、花括号“{”和“}”，且这三种括号可按任意的次序嵌套使用（如： $\cdots[\cdots\{\cdots\}\cdots[\cdots]\cdots]\cdots[\cdots]\cdots(\cdots)\cdots$ ）。编写判别给定表达式中所含括号是否正确配对出现的算法（已知表达式已存入数据元素为字符的顺序表中）。

## 2 基本作业题目解答

**【3.1 题解答】：**题集（C 语言版）》，第 3 章，第 3.9 题，难度系数为 3）  
试将下列递归过程改写

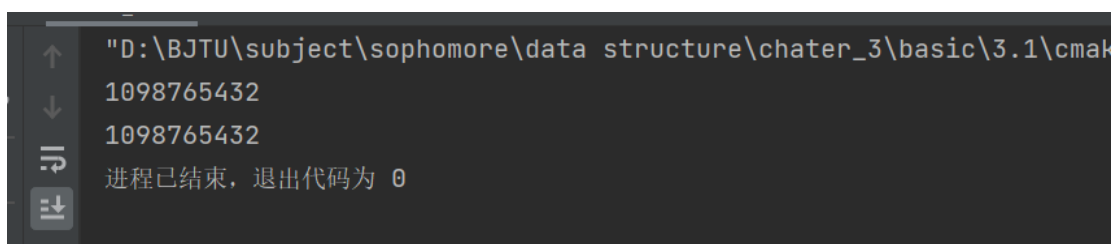
思路：

正常改写，`recursion` 写法则改为函数调用自身，并且，调用时修改参数为  $(n-1)$ 。

代码：

```
01:#include <stdio.h>
02:// 两个函数同时出现 方便对比
03:void test(int n){
04:    int i;
05:    i=n;
06:    while (i>1)
07:        printf("%d",i--);
08:}
09:void test_recursive(int n){
10:    if(n>1){
11:        printf("%d", n);
12:        test_recursive(n-1);
13:    }
14:}
15:int main() {
16:    test(10);
17:    printf("\n");
18:    test_recursive(10);
19:
20:}
```

调试:



```
"D:\BJTU\subject\sophomore\data structure\chater_3\basic\3.1\cmak
1098765432
1098765432
进程已结束, 退出代码为 0
```

调试正常

注: 原题中为 c++ 写法, 采用 c 编译环境直接报错, 故加入输出格式符。

**【3.2 题解答】:** 题集 (C 语言版)》, 第 3 章, 第 3.19 题, 难度系数为 3)

思路:

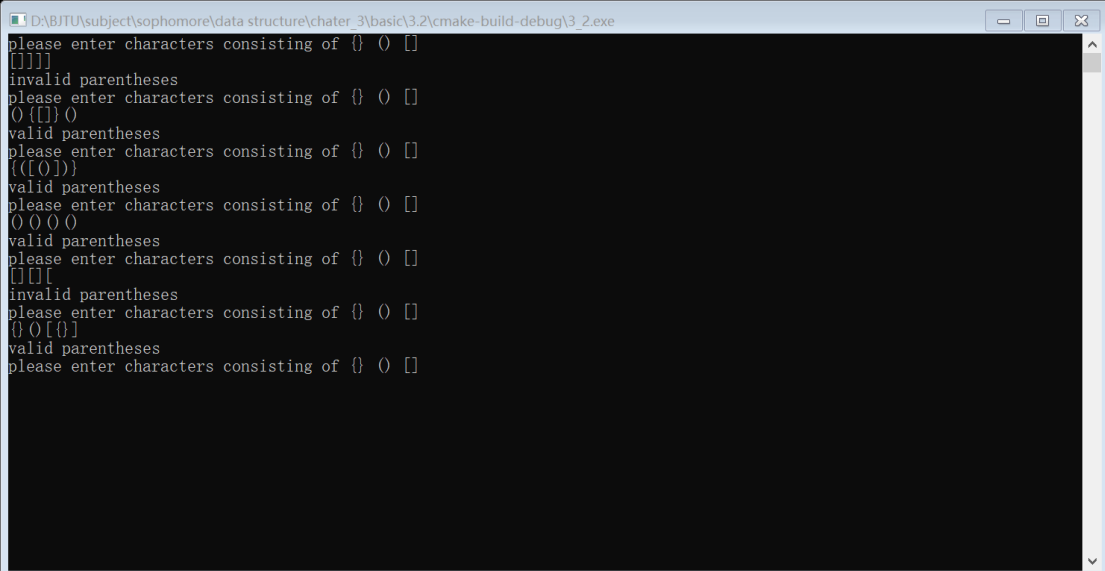
构建栈来进行回溯保存。由控制台输入一个只含 `[](){}`  的字符串 `s`, 这里以宏变量 `MAXSIZE` 进行调整, 方便调试。从右向左遍历字符串 `s`, 为 `]])` 则入栈, 为 `[{(` 则进行判断。与栈顶元素进行配对, 成功则下一步, 失败退出循环, 输出 `invalid parentheses`。最后的输出 `valid parentheses` 条件是遍历到 `s` 的最左端且空栈。

我们可以在开始循环前由奇偶判断是否为 `invalid`, 由字符串最右边元素为左括号判断 `invalid`, 从而快速判断。

```
01:#include <stdio.h>
02:#include <stdlib.h>
03:#include <string.h>
04:#define MAXSIZE 10
05:struct Stack{
06:    char *string;
07:    int count;
08:};
09:
10:
```

```
11:
12:void push(Stack&s, char c){
13:    s.string[s.count]=c;
14:    s.string[s.count+1]='\0';
15:    s.count ++;
16:}
17:void pop(Stack&s){
18:    s.string[s.count-1]='\0';
19:    s.count --;
20:}
21:int main() {
22:    Stack *s = (Stack *) malloc(sizeof(Stack));
23:    s->count = 0;
24:    s->string = (char *) malloc(sizeof(char) * MAXSIZE);
25:
26:    char tem[10];
27:    int i;
28:    printf("please enter characters consisting of {} () []\n");
29:    scanf("%s", &tem);
30:    while (strlen(tem)>0) {
31:// 想通过奇偶性判断
32:        if (strlen(tem) % 2 == 1) {
33:            printf("invalid parentheses\n");
34:        }
35:        else{
36://从右向左判断
37:            for (i = strlen(tem); i > 0; i--) {
38:                char j = tem[i - 1];
39:                if (j == '}' || j == ')' || j == ']') {
40:                    push(*s, j);
41:                } else {
42:// 括号相合判断时 先判断栈空与否
43:                    if (s->count == 0) {
44:                        printf("invalid parentheses\n");
45:                        break;
46:                    }
47:                    else if (j == '{' && s->string[s->count - 1] == '}')
48:                        pop(*s);
49:                    else if (j == '(' && s->string[s->count - 1] == ')')
50:                        pop(*s);
51:                    else if (j == '[' && s->string[s->count - 1] == ']')
52:                        pop(*s);
53:                    else break;
```

```
54:         }
55:
56:     }
57:     if (i == 0&& s->count==0) printf("valid parentheses\n");
58:     else printf("invalid parentheses\n");
59: }
60:
61:
62:     printf("please enter characters consisting of {} () []\n");
63:     scanf("%s", &tem);
64: }
65:
66: }
```



```
D:\BJTU\subject\sophomore\data structure\chater_3\basic\3.2\cmake-build-debug\3_2.exe
please enter characters consisting of {} () []
[]
invalid parentheses
please enter characters consisting of {} () []
(){}
valid parentheses
please enter characters consisting of {} () []
({()})
valid parentheses
please enter characters consisting of {} () []
(){}()
valid parentheses
please enter characters consisting of {} () []
[]{}
invalid parentheses
please enter characters consisting of {} () []
(){}[]
valid parentheses
please enter characters consisting of {} () []
```

调试正常