栈数据结构

```
#include <stdio.h>
#include <malloc.h>

typedef struct ListStack{
    int NodeData;
    struct ListStack *NextNode;//指针域指向链表的下一个节点
}ListStack,*LinkStack;
void ShowList(LinkStack LS);
LinkStack InitStack(LinkStack LS){//初始化栈队列
    LinkStack LN; int x;
    LS = (LinkStack) malloc(sizeof(ListStack));
    LS = NULL; //初始化
    scanf("%d",&x);
    while(x!=-1){
        LN = (LinkStack) malloc(sizeof(ListStack));
        LN->NodeData = x;
        LN->NextNode = LS;
        LS = LN;
        scanf("%d",&x);
    }
    return LS;
}
LinkStack StackPush(LinkStack LS,int dat){
    LinkStack LN;
    LN = (LinkStack) malloc(sizeof(ListStack));
    LN->NodeData = dat;
    LN->NextNode = LS;
    LS = LN;
    return LS;
}
LinkStack StackPop(LinkStack LS,int *dat){
    LinkStack LN;
    if(LS == NULL){
        printf("空栈");
        return NULL;//空栈
    }
    *dat = LS->NodeData;
    LN = LS;
    LS = LS->NextNode;
    free(LN);
    return LS;
}
void ShowList(LinkStack LS){ //遍历栈
    LinkStack L =  LS;
    if(L==NULL){
        printf("空栈");
    }
    while(L){
```

```
        printf("栈顶值:%d\n",L->NodeData);
        L = L->NextNode;
    }
    return ;
}
```

数制的转换

要求：对输入的任意一个非负十进制整数，输出与其等值的八进制数

代码

```
void DntoOn(LinkStack LS){
    int n,i=0,oNum;
    printf("输入一个非负十进制整数\n");
    scanf("%d",&n);
    while(n){
        int t = n%8;
        LS=StackPush(LS,t);
        n = n/8;
        i++;
    }
    while(i--){
        int t;
        LS=StackPop(LS,&t);
        printf("%d",t);
    }
}
```

- 仅输出格式 无实际八进制数

分析

根据题目描述，

将每次除的余数入栈

然后除尽之后，依次出栈

运行结果

```
int main(){
    LinkStack ls;int a;
    ls=InitStack(ls);
    DntoOn(ls);
}
//console
-1
输入一个非负十进制整数
108
154
```

流程图

括号匹配的检验

要求：检测括号是否合法

代码

```
int getStrLength(char *str);
//
int checkStr(){
    LinkStack LS;
    LS=InitStack(LS);
    // 检查括号字符串是否合法
    char KhStr[40];int len;
    gets(KhStr);
    len = getStrLength(KhStr);
    printf("检测长度%d",len);
    // 算法思想:
    // 如果有左括号就加入栈中 有相应的右括号匹配弹栈
    // 在少括号或括号不匹配的情况下判断为字符串非法
    for(int i=0;i<len;i++){
        switch(KhStr[i]){
            case '(':
            case '[':
                LS = StackPush(LS,KhStr[i]);//进栈
                break;
            case ')':
                if(LS->NodeData=='('){//栈顶匹配，弹栈
                    char t;
                    LS = StackPop(LS,&t);
                    break;
                }else{
                    return 0;//返回false
                }
            case ']':
                if(LS->NodeData=='['){//栈顶匹配，弹栈
                    char t;
                    LS = StackPop(LS,&t);
                    break;
                }else{
                    return 0;//返回false
                }
        }
        ShowList(LS);
    }
    if(LS!=NULL){return 0;}//最后的检验 判断栈是否为空
    return 1;//通过检验返回 true
}
int getStrLength(char str[]){
```

```
    int l=0;
    while(str[l]!='\0'){
        l++;
    }
    return l;
}
```

分析

根据题目分析，将暂未出错的括号入栈

如果有匹配括号出栈，如果匹配错误返回为false

如果有少括号情况返回false

运行结果

```
int main(){
    int sta=checkStr();
    if(sta==0){
        printf("不合法");
    }else{
        printf("合法");
    }
}
//console
e
([])
检测长度4栈顶值:(
----
栈顶值:[
栈顶值:(
----
栈顶值:(
----
空栈----
合法
```

流程图

修改了数据结构

```
#include <stdio.h>
#include <malloc.h>
#include <string.h>
typedef struct ListStack{
    char NodeData;
    struct ListStack *NextNode;//指针域指向链表的下一个节点
}ListStack,*LinkStack;
```

```c
void ShowList(LinkStack LS);
LinkStack InitStack(LinkStack LS){//初始化栈队列
    LinkStack LN; char x;
    LS = (LinkStack) malloc(sizeof(ListStack));
    LS = NULL; //初始化
    scanf("%c",&x);
    fflush(stdin);//防止回车输入
    while(x!='e'){
        LN = (LinkStack) malloc(sizeof(ListStack));
        LN->NodeData = x;
        LN->NextNode = LS;
        LS = LN;
        scanf("%c",&x);
        fflush(stdin);
    }
    return LS;
}
LinkStack StackPush(LinkStack LS,char dat){
    LinkStack LN;
    LN = (LinkStack) malloc(sizeof(ListStack));
    LN->NodeData = dat;
    LN->NextNode = LS;
    LS = LN;
    return LS;
}
LinkStack StackPop(LinkStack LS,char *dat){
    LinkStack LN;
    if(LS == NULL){
        printf("空栈");
        return NULL;//空栈
    }
    *dat = LS->NodeData;
    LN = LS;
    LS = LS->NextNode;
    free(LN);
    return LS;
}
void ShowList(LinkStack LS){ //遍历栈
    LinkStack L =  LS;
    if(L==NULL){
        printf("空栈");
    }
    while(L){
        printf("栈顶值:%c\n",L->NodeData);
        L = L->NextNode;
    }
    printf("----\n");
    return ;
}
```