实验3 堆栈的应用

[ 发布人： [易法令](https://www.scholat.com/flyi)   截止时间：2024-10-15 23:59:59  ]

一、实验目的:

 1．掌握堆栈的存储方式和基本操作

2．掌握堆栈后进先出运算原则在解决实际问题中的应用

 二、实验内容:

 1.利用栈结构，编写程序将十进制数转换成N制数（N可以为2、4、8、16等）。

 说明:十进制数值转换成二进制使用辗转相除法将一个十进制数值转换成二进制数值。即用该十进制数值除以2，并保留其余数；重复此操作，直到该十进制数值为0为止。最后将所有的余数反向输出就是所对应的二进制数值。十进制数值转换成八进制算法类似。转换算法要求用一个函数完成。

 2.假设算术表达式中允许包含两种括号:圆括号和方括号，其嵌套的顺序随意，即（[][]）

 或[（[]()）]等为正确格式，而[(]或()))或 [())均为不正确的格式。请使用栈结构，写一算法检验某表达式中的括号是否匹配，并测试你的算法是否正确。测试表达式为:

 (1)[(1+2)\*3-1]+[((1+2]\*3)-1]

(2) [(1+2)\*3-1]+[(1+2)\*3-1]

三、实验源代码

头文件SequenceStack源码：

typedef struct

{

    ElemType stack[MaxStackSize];

    int top;

}SequenceStack;

/\*初始化顺序栈S\*/

void StackInitiate(SequenceStack \*S)

{

    S->top = 0; /\*定义初始栈顶的下标值\*/

}

/\*判断顺序栈S是否为空\*/

int StackNotEmpty(SequenceStack S)

{

    if (S.top <= 0)

        return 0;

    else

        return 1;

}

/\*进栈\*/

int StackPush(SequenceStack \*S, ElemType x)

{

    if (S->top >= MaxStackSize)

    {

        printf("Stack is full and cannot be inserted!\n");

        return 0;   /\*操作失败返回0\*/

    }

    else

    {

        S->stack[S->top] = x;

        S->top++;

        return 1;   /\*入栈成功返回1\*/

    }

}

/\*出栈\*/

int StackPop(SequenceStack \*S, ElemType \*d)

{

    if (S->top <= 0)

    {

        printf("Stack is empty and no data out!\n");

        return 0;   /\*操作失败返回0\*/

    }

    else

    {

        S->top--;

        \*d = S->stack[S->top];  /\*将栈顶元素值赋予参数d\*/

        return 1;   /\*出栈成功返回1\*/

    }

}

/\*取栈顶元素\*/

int StackTop(SequenceStack S, ElemType \*d)

{

    if (S.top <= 0)

    {

        printf("Stack is full!\n");

        return 0;

    }

    else

    {

        \*d = S.stack[S.top - 1];    /\*当前栈顶元素赋予参数d\*/

        return 1;

    }

}

第一题源码：

#include <stdio.h>

typedef int ElemType;

#define MaxStackSize 100

#include "SequenceStack.h"

void conversion();

int main(void)

{

    conversion();

    return 0;

}

/\*将十进制数转化为二进制数\*/

void conversion()

{

    int decimal;

    SequenceStack myStack;

    StackInitiate(&myStack);

    scanf("%d", &decimal);  /\*输入一个十进制数\*/

    while (decimal)

    {

        StackPush(&myStack, decimal % 2);   /\*每次将余数压进栈中\*/

        decimal /= 2;

    }

    while (StackNotEmpty(myStack))

    {

        StackPop(&myStack, &decimal);   /\*依次弹出栈中的元素就是相应的二进制的值\*/

        printf("%d", decimal);

    }

}

第二题源码：

#include <stdio.h>

typedef char ElemType;

#define MaxStackSize 100

#include "SequenceStack.h"

int bracketMatching(char \*expression);

int main(void)

{

    char \*expr1 = "[(1+2)\*3-1]+[((1+2]\*3)-1]";

    char \*expr2 = "[(1+2)\*3-1]+[(1+2)\*3-1]";

    if (bracketMatching(expr1))

        printf("[(1+2)\*3-1]+[((1+2]\*3)-1] is True\n");

    else

        printf("[(1+2)\*3-1]+[((1+2]\*3)-1] is False\n");

    if (bracketMatching(expr2))

        printf("[(1+2)\*3-1]+[(1+2)\*3-1] is True\n");

    else

        printf("[(1+2)\*3-1]+[(1+2)\*3-1] is False\n");

    return 0;

}

int bracketMatching(char \*expression)

{

    SequenceStack tempStack;

    StackInitiate(&tempStack);  /\*初始化栈\*/

    char popchar;

    for (int i = 0; expression[i] != '\0'; i++) /\*遍历并匹配括号，失败返回0\*/

    {

        char expr = expression[i];

        switch (expr)

        {

        case '(':

        case '[':

        case '{':

            StackPush(&tempStack, expr);

            break;

        case ')':

            StackPop(&tempStack, &popchar);

            if (popchar != '(')

            {

                return 0;

            }

            break;

        case ']':

            StackPop(&tempStack, &popchar);

            if (popchar != '[')

            {

                return 0;

            }

            break;

        case '}':

            StackPop(&tempStack, &popchar);

            if (popchar != '{')

            {

                return 0;

            }

            break;

        default:

            break;

        }

    }

    /\*判断栈是否还存在左括号\*/

    if (StackNotEmpty(tempStack))

    {

        return 0;

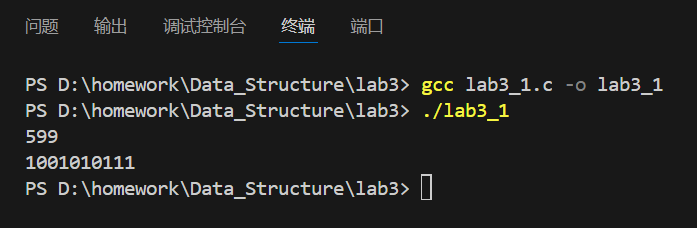
    }

    return 1;

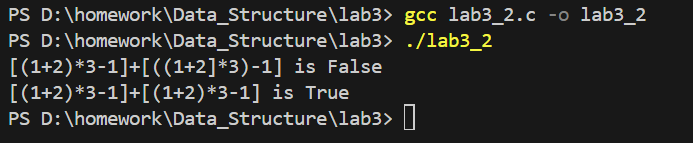
}

 四、实验结果

第一题结果：



第二题结果：



五、实验分析与总结

答：本次实验中，我掌握了堆栈的存储方式和基本操作，掌握了堆栈后进先出运算原则在解决实际问题中的应用。通过堆栈实现十进制数到N进制数的转换，利用了堆栈的后进先出特性，将余数依次入栈，最后出栈时即为所需的N进制数。这种方法不仅适用于二进制，还适用于其他进制（如八进制、十六进制等），只需改变基数即可。使用堆栈检验括号匹配问题，当遇到左括号时入栈，遇到右括号时检查栈顶元素是否匹配，若匹配则出栈，否则表示括号不匹配。该方法简单有效，能够处理任意嵌套顺序的括号，包括圆括号和方括号。