**实验二 栈的存储表示、实现及应用**

**实验目的**

1. 掌握栈的存储（尤其是顺序存储）的表示、实现；
2. 掌握栈的简单应用。

**实验内容**

1. 编写程序，利用栈，把任意非负十进制正整数转换为n（n可以为2、8、16等等）进制数输出。对每种进制转换，给出至少5组测试数据及结果。
2. 编写程序，利用栈，实现表达式求值。至少给出3组测试数据及结果。

**实验步骤**

对每一项实验内容：

（1）审清题意，分析并理出解决问题的基本思路。

（2）根据基本思路，设计好程序的算法。

（3）根据算法编写源程序。

（4）在计算机上编译程序，检验程序的可运行性。

（5）给出几组测试数据，测试程序的输出结果是否正确。

(6) 撰写实验报告。

**实验报告要求**

对每一项实验内容：

1. 自然语言描述基本思路；
2. 附关键代码

#include<iostream>  
using namespace std;  
  
typedef struct {  
    int \*base;    //连续内存，存储栈元素  
    int  \*top;    //指向栈顶元素的下一个位置  
    int  stacksize;           //当前实际可用最大容量  
} SqStack;  
  
void InitStack (SqStack &S)  
{// 构造一个空栈S  
    S.base=(int\*)malloc(100\*  
        sizeof(int));  
    if (!S.base) exit (EOVERFLOW);     //内存分配失败  
    S.top = S.base;  
    S.stacksize = 100;  //一个常量  
}  
   
void print(SqStack S){  
    while(S.top!=S.base){  
        cout << \*--S.top <<endl;  
    }  
}  
  
void Push (SqStack &S, int e) {  
    if (S.top - S.base >= S.stacksize) {//栈满，追加存储空间  
        S.base = (int \*) realloc ( S.base,(S.stacksize + 1) \* sizeof (int));  
        if (!S.base) exit (EOVERFLOW); //存储分配失败  
        S.top = S.base + S.stacksize;  
        S.stacksize += 1;  
    }  
    \*S.top++ = e;    // \*S.top = e; S.top++;  
}  
  
  
void Pop (SqStack &S) {  
    if (S.top == S.base) exit(-1);  
    S.top--;  
}  
  
string decimalToBaseN(int decimal, int base) {  
    if (decimal == 0) {  
        return "0";  
    }  
      
    const string digits = "0123456789ABCDEF";  // 用于表示16进制的字符  
    SqStack S;  
    InitStack(S);  
      
    while (decimal > 0) {  
        int remainder = decimal % base;  
        Push(S,remainder);  
        decimal /= base;  
    }  
      
    string result;  
    while (S.top!=S.base) {  
        result += digits[\*(S.top-1)];  
        Pop(S);  
    }  
    free(S.base);  
    return result;  
}  
  
int main(){  
    string result = decimalToBaseN(29,15);  
    cout << result <<endl;  
}

#include **<**iostream**>**#include **<**stack**>  
using** **namespace** **std;  
  
int** Get\_Priority**(char** c**)** **{**    **switch** **(**c**)** **{**    **case** '+'**:**    **case** '-'**:**        **return** 1**;**    **case** '\*'**:**    **case** '/'**:**        **return** 2**;**    **default:**        **return** 0**;** *//* *默认情况下返回0*    **}  
}  
  
char** Precede**(char** Top**,** **char** c**)** **{**    **if** **(**Get\_Priority**(**Top**)** **<** Get\_Priority**(**c**))** **{**        **return** '<'**;**    **}** **else** **{**        **return** '>'**;**    **}  
}  
  
int** CalculationOfTwo**(int** a**,** **int** b**,** **char** c**)** **{**    **switch** **(**c**)** **{**    **case** '+'**:**        **return** a **+** b**;**    **case** '-'**:**        **return** a **-** b**;**    **case** '\*'**:**        **return** a **\*** b**;**    **case** '/'**:**        **return** a **/** b**;**    **}**    **return** 0**;** *//* *返回一个默认值***}  
  
int** EvaluateExp**()** **{**    stack**<char>** OPTR**;**    stack**<int>** OPND**;**    OPTR**.**push**(**'#'**);**      
    **while** **(char** c **=** getchar**())** **{**        **if** **(**isdigit**(**c**))** **{**            OPND**.**push**(**c **-** '0'**);** *//* *将字符转换为整数*        **}** **else** **if** **(**c **==** '('**)** **{**            OPTR**.**push**(**c**);**        **}** **else** **if** **(**c **==** ')'**)** **{**            **while** **(**OPTR**.**top**()** **!=** '('**)** **{**                **int** a **=** OPND**.**top**();**                OPND**.**pop**();**                **int** b **=** OPND**.**top**();**                OPND**.**pop**();**                **char** op **=** OPTR**.**top**();**                OPTR**.**pop**();**                **int** result **=** CalculationOfTwo**(**b**,** a**,** op**);**                OPND**.**push**(**result**);**            **}**            OPTR**.**pop**();** *//* *弹出左括号*        **}** **else** **if** **(**c **==** '#'**)** **{**            **while(**OPTR**.**top**()!=**'#'**){**                **int** a **=** OPND**.**top**();**                OPND**.**pop**();**                **int** b **=** OPND**.**top**();**                OPND**.**pop**();**                **char** op **=** OPTR**.**top**();**                OPTR**.**pop**();**                **int** result **=** CalculationOfTwo**(**b**,** a**,** op**);**                OPND**.**push**(**result**);**            **}**            **return** OPND**.**top**();**        **}** **else** **{**            **if(**OPTR**.**top**()!=**'#'**){**                **switch** **(**Precede**(**OPTR**.**top**(),** c**))** **{**                **case** '<'**:**                    OPTR**.**push**(**c**);**                    **break;** *//* *添加* *break* *语句*                **case** '>'**:**                     **int** a **=** OPND**.**top**();**                    OPND**.**pop**();**                    **int** b **=** OPND**.**top**();**                    OPND**.**pop**();**                    **char** op **=** OPTR**.**top**();**                    OPTR**.**pop**();**                    **int** result **=** CalculationOfTwo**(**b**,** a**,** op**);**                    OPND**.**push**(**result**);**                    OPTR**.**push**(**c**);**                    *//* *不要忘记更新* *c*                    **break;** *//* *添加* *break* *语句*                **}**            **}**            **else{**                OPTR**.**push**(**c**);**            **}**        **}**    **}**    **return** 0**;  
}  
  
int** main**()** **{**    cout **<<** EvaluateExp**()** **<<** endl**;**    **return** 0**;  
}**