#include<stdlib.h>

#include<stdio.h>

#include<iostream>

using namespace std;

#define STACK\_INIT\_SIZE 100

#define STACKINCREMENT 10

typedef char SElemType;

typedef struct {

SElemType \*base;

SElemType \*top;

int stacksize;

}SqStack;

void InitStack(SqStack &S) {

S.base = new SElemType[STACK\_INIT\_SIZE];

S.top = S.base;

S.stacksize = STACK\_INIT\_SIZE;

}

void DestoyStack(SqStack &S) {

if (S.base)delete[] S.base;

}

bool StackEmpty(SqStack S) {

if (S.base == S.top)return true;

else return false;

}

SElemType GetTop(SqStack S) {

if (!StackEmpty(S)) {

return \*(S.top - 1);

}

}

void Push(SqStack &S, SElemType e) {

if (S.top - S.base >= S.stacksize) {

S.base = (SElemType\*)realloc(S.base, S.stacksize + STACKINCREMENT);

S.top = S.base + S.stacksize;

S.stacksize += STACKINCREMENT;

}

\*S.top++ = e;

}

void Pop(SqStack &S, SElemType &e) {

e = \*--S.top;

}

bool In(char c) {

switch (c) {

case'+':

case'-':

case'\*':

case'/':

case'(':

case')':

case'#':

return true;

default:

return false;

}

}

char Precede(char a, char b) {

switch (a) {

case '+':

switch (b) {

case '+':return '>';

case '-':return '>';

case '\*':return '<';

case '/':return'<';

case '(':return'<';

case ')':return'>';

case '#':return'>';

}

case '-':

switch (b) {

case '+':return '>';

case '-':return '>';

case '\*':return '<';

case '/':return'<';

case '(':return'<';

case ')':return'>';

case '#':return'>';

}

case '\*':

switch (b) {

case '+':return '>';

case '-':return '>';

case '\*':return '>';

case '/':return'>';

case '(':return'<';

case ')':return'>';

case '#':return'>';

}

case '/':

switch (b) {

case '+':return '>';

case '-':return '>';

case '\*':return '>';

case '/':return'>';

case '(':return'<';

case ')':return'>';

case '#':return'>';

}

case '(':

switch (b) {

case '+':return '<';

case '-':return '<';

case '\*':return '<';

case '/':return'<';

case '(':return'<';

case ')':return'=';

}

case ')':

switch (b) {

case '+':return '>';

case '-':return '>';

case '\*':return '>';

case '/':return'>';

case ')':return'>';

case '#':return'>';

}

case '#':

switch (b) {

case '+':return '<';

case '-':return '<';

case '\*':return '<';

case '/':return'<';

case '(':return'<';

case '#':return'=';

}

}

}

int Operate(SElemType a, SElemType theta, SElemType b) {

int result = 0;

switch (theta) {

case'+':result = a + b; break;

case'-':result = a - b; break;

case'\*':result = a \* b; break;

case'/':result = a / b; break;

}

return result;

}

int EvaluateExpression() {

SqStack OPTR, OPND;

InitStack(OPTR);

Push(OPTR, '#');

InitStack(OPND);

SElemType c = getchar();

while (c != '#' || GetTop(OPTR) != '#') {

if (!In(c)) {

char temp[STACK\_INIT\_SIZE] = {0};

for (int i = 0; !In(c); i++) {

temp[i] = c;

c = getchar();

}

int d = atoi(temp);

Push(OPND, d);

}

else {

switch (Precede(GetTop(OPTR), c)) {

case '<':

Push(OPTR, c);

c = getchar();

break;

case'=':

Pop(OPTR, c);

c = getchar();

break;

case'>':

SElemType theta, a, b;

Pop(OPTR, theta);

Pop(OPND, b);

Pop(OPND, a);

Push(OPND, Operate(a, theta, b));

break;

}

}

}

int result = GetTop(OPND);

DestoyStack(OPND);

DestoyStack(OPTR);

return result;

}

void main() {

cout << "Please enter an operand(end with #):" << endl;

cout << EvaluateExpression() << endl;

}

心得：

本次作业采用了栈的方法，将“滞后”的思想（即先进后出）体现出来。总体而言算法难度不大，但细节较多，如字符串的分类读取等等。

通过本次作业，对栈有了更加深刻的理解，也对编译器的工作原理有了更深的理解。