**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №6**

**по дисциплине «Программирование»**

Тема: Стек-Машина

|  |  |  |
| --- | --- | --- |
| Студент гр. 6304 |  | Корытов П.В. |
| Преподаватель |  | Кринкин К.В. |

Санкт-Петербург

2017

**Цель работы**

Реализовать на языке С работу со стеком – добавление элементов, их считывание и удаление.

**Задание**

Создать программу, на вход которой подается последовательность (не более 100 элементов) из чисел и арифметических операций (+, -, \*, / (деление нацело)) разделенных пробелом, которые программа должна интерпретировать и выполнить по следующим правилам:

* Если очередной элемент входной последовательности - число, то положить его в стек
* Если очередной элемент - знак операции, то применить эту операцию над двумя верхними элементами стека, а результат положить обратно в стек (следует считать, что левый операнд выражения лежит в стеке глубже)
* Если входная последовательность закончилась, то вывести результат (число в стеке)
* Если в процессе вычисления возникает ошибка (для операции в стеке не хватает аргументов или по завершении работы программы в стеке более одного элемента), то вместо результата следует вывести "error"

**Ход работы**

В данной программе стек представляет собой список структур. Каждая структура содержит указатель на предыдущую и на следующую.

typedef struct num{

int a;

struct num \*next;

struct num \*prev;

}num;

Функция push добавляет элемент в стек следующим образом: сначала записывает элемент в текущую структуру, выделяет память на следующую и связывает указатели в необходимом порядке

num\* push(num\* stack, int a){ //Pushing element to the stack

stack->a = a;

stack->next = (num\*)malloc(sizeof(num));

stack->next->prev=stack;

stack = stack->next;

stack->next = NULL;

return stack;

}

Программе необходимо вытаскивать элементы из стека. Это делает функция pop.

num\* pop(num\* stack, int\* a){

if (stack!=NULL){

if (stack->prev!=NULL){

\*a=stack->prev->a;

stack = stack->prev;

free(stack->next);

return stack;

}

}

return NULL;

}

Для работы программы необходимо доставать по два элемента за раз. Функция, предназначенная для этой задачи, возвращает 0, если достать элементы получилось, а иначе – 1.

int getLastTwo(num\*\* stack, int \*a, int \*b){ //Getting last two elements

\*stack = pop(\*stack, a);

if (\*stack!= NULL){

\*stack = pop(\*stack, b);

if (\*stack!=NULL){

return 0;

}

}

return 1;

}

В главной части сначала считывается строка, затем она делится на «токены» с помощью strtok, после чего обрабатывает каждый токен отдельно. Если токен – знак «+», «\*» или «\», то к последним двум элементам применяется полученная операция и результат замещается. Если же токен начинается на «-», то отдельно проверяется, является ли токен просто знаком «-» или отрицательным числом.

Если токен – число, то он записывается в стек.

В программе переменная err содержит информацию, были ли допущены ошибки в ходе обработки стека – т.е. не пытались ли достать несуществующий элемент и т.п. Если err равен 0 и если стек корректно завершается, то выводится единственный оставшийся элемент. Иначе выводится сообщение «error».

int main()

{

int a; int b; int err = 0;

char\* str = (char\*)malloc(N\*sizeof(char));

fgets(str, N, stdin); //Input. I couldn't figure out the ending symbol of input, so fgets instead of getchar cycle

num\* stack = NULL;

char \*s = strtok(str, " ");

while ((s!=NULL) && (err == 0)){ //There is no point of going through the cycle if an error was found

switch (s[0]){

case '+':

err = getLastTwo(&stack, &a, &b);

stack = push(stack, a+b);

break;

case '\*':

err = getLastTwo(&stack, &a, &b);

stack = push(stack, a\*b);

break;

case '/':

err = getLastTwo(&stack, &a, &b);

stack = push(stack, b/a);

break;

default:

if ((s[0]=='-') && ((s[1] == '\0') || (s[1] == '\n'))){ //Both "-" and negative number start with '-'

err = getLastTwo(&stack, &a, &b); //So i check this case separately

stack = push(stack, b-a);

}

else{

if (stack == NULL) //If there were no pushes, init stack.

stack = stackInit(stack); //It's easier to understand that way if there were no "oush" calls.

stack = push(stack, atoi(s));

}

break;

}

s = strtok(NULL, " ");

}

if ((err == 1) || (stack->prev->prev!=NULL)) //Checking also if there is more than one element in stack

printf("error\n");

else

printf("%d", stack->prev->a);

while (stack!=NULL){

stack = pop(stack, &a);

}

return 0;

}

Код скопирован в среду разработки и скомпилирован. Файлы запушены на git через командную строку. Сделан отчет.

**Вывод.**

Язык С позволяет создавать и обрабатывать стеки – добавлять, считывать и заменять элементы.

Приложение А. Исходный код

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define N 600

typedef struct num{

int a;

struct num \*next;

struct num \*prev;

}num;

num\* stackInit(num\* stack){ //Initializing stack

stack = (num\*)malloc(sizeof(num));

stack->prev = NULL;

return stack;

}

num\* push(num\* stack, int a){ //Pushing element to the stack

if (stack!=NULL){

stack->a = a;

stack->next = (num\*)malloc(sizeof(num));

stack->next->prev=stack;

stack = stack->next;

return stack;

}

return NULL;

}

num\* pop(num\* stack, int\* a){

if (stack!=NULL){

if (stack->prev!=NULL){

\*a=stack->prev->a;

stack = stack->prev;

free(stack->next);

return stack;

}

}

return NULL;

}

int getLastTwo(num\*\* stack, int \*a, int \*b){ //Getting last two elements

\*stack = pop(\*stack, a);

if (\*stack!= NULL){

\*stack = pop(\*stack, b);

if (\*stack!=NULL){

return 0;

}

}

return 1;

}

int main()

{

int a; int b; int err = 0;

char\* str = (char\*)malloc(N\*sizeof(char));

fgets(str, N, stdin); //Input. I couldn't figure out the ending symbol of input, so fgets instead of getchar cycle

num\* stack = NULL;

char \*s = strtok(str, " ");

while ((s!=NULL) && (err == 0)){ //There is no point of going through the cycle if an error was found

switch (s[0]){

case '+':

err = getLastTwo(&stack, &a, &b);

stack = push(stack, a+b);

break;

case '\*':

err = getLastTwo(&stack, &a, &b);

stack = push(stack, a\*b);

break;

case '/':

err = getLastTwo(&stack, &a, &b);

stack = push(stack, b/a);

break;

default:

if ((s[0]=='-') && ((s[1] == '\0') || (s[1] == '\n'))){ //Both "-" and negative number start with '-'

err = getLastTwo(&stack, &a, &b); //So i check this case separately

stack = push(stack, b-a);

}

else{

if (stack == NULL) //If there were no pushes, init stack.

stack = stackInit(stack); //It's easier to understand that way if there were no "oush" calls.

stack = push(stack, atoi(s));

}

break;

}

s = strtok(NULL, " ");

}

if ((err == 1) || (stack->prev->prev!=NULL)) //Checking also if there is more than one element in stack

printf("error\n");

else

printf("%d", stack->prev->a);

while (stack!=NULL){

stack = pop(stack, &a);

}

return 0;

}