**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

отчет

**по лабораторной работе №2**

**по дисциплине «Программирование»**

Тема: **Динамические структуры данных. Тестирование.**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 6304 |  | Иванкова В.М. |
| Преподаватель |  | Берленко Т.А. |

Санкт-Петербург

2017

Оглавление

[Цель: 3](#_Toc484110814)

[Задание: 3](#_Toc484110815)

[Содержание: 3](#_Toc484110816)

[Инициализация 3](#_Toc484110817)

[Разбиение на лексемы 3](#_Toc484110818)

[Работа стековой машины 3](#_Toc484110819)

[Вывод 5](#_Toc484110820)

[Приложение 6](#_Toc484110821)

Цель:

Написать программу, на вход которой подается последовательность из чисел и арифметических операций, разделенных пробелом, которые программа должна интерпретировать и выполнить по определенным правилам. Стек, который требуется реализовать самостоятельно, требуется выполнить на базе массива.

Задание:

**Стековая машина.**

На вход программе 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".

Стек требуется реализовать самостоятельно на базе массива.

Содержание:

Создание структуры

typedef struct index{

int a;

struct index \*next;

struct index \*prev;

}index;

Инициализация

index\* stackInit(index\* stack){

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

stack->prev = NULL;

return stack;

}

Добавление элемента в стек

index\* push(index\* stack, int a){

if (stack!=NULL){

stack->a = a;

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

stack->next->prev=stack;

stack = stack->next;

return stack;

}

return NULL;

}

Размер стека

int size(index\* stack){

int i=0;

while (stack->prev!=NULL){

i++;

stack = stack->prev;

}

return I;

}

Вынимаем первые два элемента стека

int getLastTwo(index\*\* stack, int \*a, int \*b){

pop(stack, a);

if (\*stack!= NULL){

pop(stack, b);

if (\*stack!=NULL){

return 0;

}

}

return 1;

}

Вынимаем элемент из стека

void pop(index\*\* stack2, int\* a){

index\* stack = \*stack2;

if (stack!=NULL){

if (stack->prev!=NULL){

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

stack = stack->prev;

free(stack->next);

\*stack2 = stack;

}

else {

free(\*stack2);

\*stack2 = NULL;

}

}

}

Разбиение на лексемы

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

}

Работа стековой машины

while ((k!=NULL) && (end == 0)){

switch (k[0]){

case '+':

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

stack = push(stack, a+b);

break;

case '\*':

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

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

break;

case '/':

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

stack = push(stack, b/a);

break;

default:

if ((k[0]=='-') && ((k[1] == '\0') || (k[1] == '\n'))){

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

stack = push(stack, b-a);

}

else{

if (stack == NULL)

stack = stackInit(stack);

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

}

break;

}

k = strtok(NULL, " ");

}

if ((end == 1) || (size(stack)!=1))

printf("error\n");

else

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

while (stack!=NULL){

pop(&stack, &a);

}

return 0;

}

# Вывод

В ходе выполнения лабораторной работы, используя стандартные средства языка С, была самостоятельно смоделирована такая динамическая структура данных как стек (в данном случае на базе массива), которая позволила удобно хранить и обрабатывать данные для решения поставленной задачи.

Приложение

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define N 600

typedef struct index{

int a;

struct index \*next;

struct index \*prev;

}index;

index\* stackInit(index\* stack){

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

stack->prev = NULL;

return stack;

}

index\* push(index\* stack, int a){

if (stack!=NULL){

stack->a = a;

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

stack->next->prev=stack;

stack = stack->next;

return stack;

}

return NULL;

}

void pop(index\*\* stack2, int\* a){

index\* stack = \*stack2;

if (stack!=NULL){

if (stack->prev!=NULL){

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

stack = stack->prev;

free(stack->next);

\*stack2 = stack;

}

else {

free(\*stack2);

\*stack2 = NULL;

}

}

}

int size(index\* stack){

int i=0;

while (stack->prev!=NULL){

i++;

stack = stack->prev;

}

return i;

}

int getLastTwo(index\*\* stack, int \*a, int \*b){

pop(stack, a);

if (\*stack!= NULL){

pop(stack, b);

if (\*stack!=NULL){

return 0;

}

}

return 1;

}

int main()

{

int a; int b; int end= 0;

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

fgets(str, N, stdin);

index\* stack = NULL;

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

while ((k!=NULL) && (end == 0)){

switch (k[0]){

case '+':

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

stack = push(stack, a+b);

break;

case '\*':

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

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

break;

case '/':

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

stack = push(stack, b/a);

break;

default:

if ((k[0]=='-') && ((k[1] == '\0') || (k[1] == '\n'))){

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

stack = push(stack, b-a);

}

else{

if (stack == NULL)

stack = stackInit(stack);

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

}

break;

}

k = strtok(NULL, " ");

}

if ((end == 1) || (size(stack)!=1))

printf("error\n");

else

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

while (stack!=NULL){

pop(&stack, &a);

}

return 0;

}