**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра математического обеспечения и применения ЭВМ**

отчет

**по лабораторной работе №2**

**по дисциплине «Программирование»**

Тема: ДИНАМИЧЕСКИЕ СТРУКТУРЫ ДАННЫХ

|  |  |  |
| --- | --- | --- |
| Студент гр. 6303 |  | Матюшина М.Е. |
| Преподаватель |  | Берленко Т.А. |

Санкт-Петербург

2017

Цель работы: реализация стековой машины на базе списка.

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**"

**Ход работы.**

1. **Описание стуктуры.**

* typedef struct Stack {

int data;

struct Stack \* prev;

} Stack;

Здесь int data – целое число, помещаемое в стек, а struct Stack \* prev – указатель на предыдущий элемент в стека.

1. **Функции для работы со стеком.**

* **Добавление элемента в стек.**

Функция принимает 2 аргумента:

1)указатель на указатель вершины (крайнего элемента) стека

2)целое число, которое будет добавлено в стек

void push (Stack\*\* top, int element) {

Stack\* ptr=(Stack\*)malloc(sizeof(Stack)); //*выделение памяти для элемента*

ptr->prev= \*top;

ptr->data=element;

\*top=ptr;

}

* **Удаление вершины стека.**

Функция принимает в качестве аргумента указатель на указатель вершины стека.

void pop( Stack\*\* top) {

\*top=(\*top)->prev;

}

* **Возвращение элемента, находящегося на вершине стека.**

int top\_element(Stack\* top) {

return top->data;

}

* **Подсчет элементов стека.**

Функция принимает в качестве аргумента указатель на вершину стека.

int count(Stack\* top) {

int n=0;

while (top!=NULL) {

top=top->prev;

++n;

}

return n;

}

* **Функция для выполнения арифметических операций над двумя крайними элементами стека, с последующим удалением этих элементов и добавлением результата в стек**.

Принимает в качестве аргументов:

1)символ – вид арифметической операции

2)указатель на указатель вершины стека

void math(char c, Stack\*\* top){

int a, b;

a=top\_element( \*top);

remove\_element(top);

b=top\_element( \*top);

remove\_element(top);

int result;

switch(c) {

case '+':

result=a+b;

break;

case '-':

result=b-a;

break;

case '\*':

result=a\*b;

break;

case '/':

result=a/b;

break;

}

push(top, result);

}

**3.Функция main()**

В функции main происходят считывание строки, разбиение ее на лексемы функцией strtok (по пробелам), далее строки длиной 1 сравниваются со знаками арифметических действий, в случае совпадения, вызывается функция math. Если длина лексемы равна 2, то смотрим, не является ли она последним элементом исходной строки, если является, то вызывается функция math. Символы остальных лексем интерпретируются как целые числа и помещаются в стек.

int main() {

Stack\* top=NULL; //*указатель на вершину стека*

char\* str=malloc(100\*sizeof(char));

fgets(str, 100, stdin);

int element;

int control; //*счетчик количества элементов в стеке*

char\* part=strtok(str, " "); //*разделение исходной строки str на лексемы part, по пробелам*

while (part!=NULL){

if ( (strlen(part)==1) || ((strlen(part)==2)&&(part[1]=='\n')) ) {

if ((part[0]=='+')||(part[0]=='-')||(part[0]=='\*')||(part[0]=='/')){

control=count(top);

if (control>1) {

math(part[0], &top);

}

else {

printf("error\n");

return 0;

}

}

//atoi - интерпретирует содержания строки, как целое цисло

else {

element=atoi(part);

push(&top, element);

}

}

else if ((strlen(part)!=1)&&(part[strlen(part)-1]!='\n')) {

element=atoi(part);

push(&top, element);

}

part = strtok(NULL, " ");

}

control=count(top);

if (control==1) {

printf("%d\n", top->data);

}

else {

printf("error\n");

}

return 0;

}

1. **Полный код программы.**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

typedef struct Stack {

int data;

struct Stack \* prev;

} Stack;

void push (Stack\*\* top, int element) {

Stack\* ptr=(Stack\*)malloc(sizeof(Stack));

ptr->prev= \*top;

ptr->data=element;

\*top=ptr;

}

void pop( Stack\*\* top) {

\*top=(\*top)->prev;

}

int count(Stack\* top) {

int n=0;

while (top!=NULL) {

top=top->prev;

++n;

}

return n;

}

void math(char c, Stack\*\* top){

int a, b;

a=(\*top)->data;

Stack \* ptr=(\*top)->prev;

b=(ptr)->data;

int result;

switch(c) {

case '+':

result=a+b;

break;

case '-':

result=b-a;

break;

case '\*':

result=a\*b;

break;

case '/':

result=a/b;

break;

}

pop(top);

pop(top);

push(top, result);

}

int main() {

Stack\* top=NULL;

char\* str=malloc(100\*sizeof(char));

fgets(str, 100, stdin);

int element;

int control;

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

while (part!=NULL){

if ( (strlen(part)==1) || ((strlen(part)==2)&&(part[1]=='\n')) ) {

if ((part[0]=='+')||(part[0]=='-')||(part[0]=='\*')||(part[0]=='/')){

control=count(top);

if (control>1) {

math(part[0], &top);

}

else {

printf("error\n");

return 0;

}

}

else {

element=atoi(part);

push(&top, element);

}

}

else if ((strlen(part)!=1)&&(part[strlen(part)-1]!='\n')) {

element=atoi(part);

push(&top, element);

}

part = strtok(NULL, " ");

}

control=count(top);

if (control==1) {

printf("%d\n", top->data);

}

else {

printf("error\n");

}

return 0;

}

1. **Загрузка файла на GitHub.**

* В локальном репозитории была создана новая ветка matyushina\_2sem\_lr2

git checkout -b matyushina\_2sem\_lr2

* Создана директория matyushina\_2sem\_lr2

mkdir matyushina\_2sem\_lr2

* В вышеуказанную папку был перемещен файл LAB2.c
* Исходный файл добавлен в локальный репозиторий

git add LAB2.c

* Командой <git commit –m> были зафиксированы изменения в проекте

git commit –m «matyushina\_2sem\_lr2»

* Командой <git push> последний commit был отправлен на удаленный репозиторий

git push origin matyushina\_2sem\_lr2

**Вывод:** в ходе работы был реализована стековая машина на базе списка, написана программа, интерпретирующая строку и выполняющая арифметические операции с ее элементами. Код и отчет по работе были загружены в репозиторий.