**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 |  | Малышенко Ю.И. |
| Преподаватель |  | Берленко Т.А. |

Санкт-Петербург

2017

Цель работы: реализация стековой машины на базе массива.

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**"

**Использованные команды:**

* **push**  -  добавляет число в стек.
* **pop** - удаляет из стека последний элемент.
* **top** - возвращает текущий элемент с вершины.
* **size** – подсчитывает элементы в стеке.
* **input** – выполняет арифметические операции над двумя последними элементами стека, в дальнейшем удаляя эти элементы и добавляя результат в стек.

**Ход работы.**

1. **Описание стуктуры.**

#define STACK\_MAX 100

typedef struct Stack\_tag

{

int data[STACK\_MAX];

int size;

} Stack\_t;

1. **Функции для работы со стеком.**

**А) push**

void push(Stack\_t \*stack, const I value)

{

stack->data[stack->size] = value stack->size++;

}

**Б) push**

int pop(Stack\_t \*stack)

{

stack->size--;

return stack->data[stack->size];

}

**В) Size**

int Size (Stack\_t \*stack)

{

return stack->size;

}

**Г) top**

int top(const Stack\_t \*stack)

{

return stack->data[stack->size - 1];

}

**Д) input**

int input(char n, Stack\_t \*stack)

{

if (stack->size == 0 || stack->size == 1 )

{

return 1;

}

int x;

int y;

if(Size(stack) != 0)

{

x=top(stack);

pop(stack);

y=top(stack);

pop(stack);

}

int result;

switch(n)

{

case '+': { result = y+x; break; }

case '-': { result = y-x; break; }

case '\*': { result = y\*x; break; }

case '/': { result = y/x; break; }

}

push(stack, result);

return 0;

}

**3.Функция main()**

В функции main происходят считывание строки, разбиение ее на токены функцией strtok (по пробелам и \n), далее мы рассматриваем элементы посимвольно функцией isdigit(Возвращает ненулевое значение, если её аргумент является десятичной цифрой, в противном случае возвращается нуль), если элемент является положительным числом, то он записывается в стэк, если элемент отрицательный, его длина должна быть строго больше 1 , если элемент является арифметическим знаком, то его длина должная быть равна 1, тогда вызовется функция input.

int main()

{

Stack\_t stack;

stack.size=0;

char arr[100];

int tmp;

int i;

int control = 1;

fgets(arr, 100, stdin);

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

while(arrPtr != NULL)

{

if(isdigit(\*arrPtr))

push(&stack, atoi(arrPtr));

else

if(strlen(arrPtr) > 1)

{

push(&stack, atoi(arrPtr));

}

else

if(strlen(arrPtr) == 1)

{

if(input(\*arrPtr, &stack) == 1)

control = 1;

}

arrPtr = strtok(NULL, " \n");

}

if(Size(&stack) == 1)

printf("%d\n", top(&stack));

else

{

control = 1;

printf("error");

return 0;

}

return 0;

}

1. **Исходный код.**

#include <stdio.h>

#include <stdlib.h>

#include <ctype.h>

#include <string.h>

#define STACK\_MAX 100

typedef int I;

typedef struct Stack\_tag

{

I data[STACK\_MAX];

I size

} Stack\_t;

void push(Stack\_t \*stack, const I value)

{

stack->data[stack->size] = value; //указатель на структуру

stack->size++;

}

int pop(Stack\_t \*stack)

{

stack->size--;

return stack->data[stack->size];

}

int Size (Stack\_t \*stack)

{

return stack->size;

}

int top(const Stack\_t \*stack)

{

return stack->data[stack->size - 1];

}

int input(char n, Stack\_t \*stack)

{

if (stack->size == 0 || stack->size == 1 )

{

return 1;

}

int x;

int y;

if(Size(stack) != 0)

{

x=top(stack);

pop(stack);

y=top(stack);

pop(stack);

}

int result;

switch(n)

{

case '+': { result = y+x; break; }

case '-': { result = y-x; break; }

case '\*': { result = y\*x; break; }

case '/': { result = y/x; break; }

}

push(stack, result);

return 0;

}

int main()

{

Stack\_t stack;

stack.size=0;

char arr[100];

int tmp;

int i;

int control = 1;

fgets(arr, 100, stdin);

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

while(arrPtr != NULL)

{

if(isdigit(\*arrPtr))

push(&stack, atoi(arrPtr));

else

if(strlen(arrPtr) > 1)

{

push(&stack, atoi(arrPtr));

}

else

if(strlen(arrPtr) == 1)

{

if(input(\*arrPtr, &stack) == 1)

control = 1;

}

arrPtr = strtok(NULL, " \n");

}

if(Size(&stack) == 1)

printf("%d\n", top(&stack));

else

{

control = 1;

printf("error");

return 0;

}

return 0;

}

**Вывод**: в ходе лабораторной работы был реализован стэк на базе массива. Написана программа, интерпретирующая строку и выполняющая арифметические операции с ее элементами.