**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 |  | Иванов Д.В. |
| Преподаватель |  | Берленко Т.А. |

Санкт-Петербург

2017

**Цель работы:** провести обзор стандартной библиотеки, реализовать стек на базе односвязного списка.

**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 списка.

**Ход работы:**

1. Подключены необходимые для работы заголовочные файлы

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

1. Объявлена структура – односвязный список, на базе которого будет реализован стек.

typedef struct StackL

{

int number;

struct StackL \*next;

} StackL;

1. Объявлены и определены все функции, необходимые для работы стековой машины.

* Функция для подсчёта кол-ва элементов в стеке

int getSize(StackL \*head);

…

int getSize(StackL \*head)

{

int size = 0;

while (head) {

size++;

head = head->next;

}

return size;

}

* Функция для помещения элемента в стек

void push(StackL \*\*head, int number);

…

void push(StackL \*\*head, int number)

{

StackL \*tmp = (StackL\*)malloc(sizeof(StackL));

tmp->next = \*head;

tmp->number = number;

\*head = tmp;

}

* Функция для удаление элемента из стека

void pop(StackL \*\*head);

…

void pop(StackL \*\*head)

{

\*head = (\*head)->next;

}

* Функция для выбора арифметической операции

void operation(char buff, StackL \*\*head);

…

void operation(char buff, StackL \*\*head)

{

switch (buff)

{

case '+':

plus(head);

break;

case '-':

minus(head);

break;

case '\*':

times(head);

break;

case '/':

divid(head);

break;

default:

break;

}

}

* Функция для нахождения суммы двух элементов

void plus(StackL \*\*head);

…

void plus(StackL \*\*head)

{

int a, b;

a = (\*(head))->number;

b = (\*(head))->next->number;

pop(head);

pop(head);

push(head, b + a);

}

* Функция для нахождения разности двух элементов

void minus(StackL \*\*head);

…

void minus(StackL \*\*head)

{

int a, b;

a = (\*(head))->number;

b = (\*(head))->next->number;

pop(head);

pop(head);

push(head, b - a);

}

* Функция для нахождения произведения двух элементов

void times(StackL \*\*head);

…

void times(StackL \*\*head)

{

int a, b;

a = (\*(head))->number;

b = (\*(head))->next->number;

pop(head);

pop(head);

push(head, b\*a);

}

* Функция для нахождения разности двух элементов

void divid(StackL \*\*head);

…

void divid(StackL \*\*head)

{

int a, b;

a = (\*(head))->number;

b = (\*(head))->next->number;

pop(head);

pop(head);

push(head, b / a);

}

1. В main()

* Объявлены указатель head на верхний элемент стека, массив char str[100], в которую будет записана поданная строка.

StackL \*head = NULL;

char str[100];

fgets(str, 100, stdin);

* Поданная строка делится на лексемы, которые обрабатываются в цикле while.

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

while (lexem != NULL)

{...}

* + В цикле лексема приводится в числовой вид посредством функции стандартной библиотеки atoi().

Если поданная строка не является числом (символ/слово), то функция atoi() возвращает 0.

Поскольку строка может содержать ‘0’, после обработки которого atoi() вернет 0, требуется проверить является ли лексема ‘0’, если atoi() вернула 0.

int numb=atoi(lexem);

if ((numb == 0) && (\*lexem != '0'))

{...}

* + - Если это не так, то лексема содержит символ (один из символов арифметических операций). В этом случае происходит проверка – достаточно ли в стеке элементов для проведения арифметической операции.

Если элементов недостаточно программа выводит «error» и завершает работу.

В противном случае, программа вызывает функцию operation(), передавая ей лексему и указатель на первый элемент стека, которая в свою очередь вызывает функцию, проводящую необходимую арифметическую операцию.

if (getSize(head) == 1)

{

printf("error");

return 0;

}

else

operation(\*lexem, &head);

* + - Если же возвращенное значение отлично от 0, программа кладёт данное число в стек.

else

push(&head, numb);

* + Затем происходит поиск следующей лексемы

lexem = strtok(NULL, " ");

* При удачном выходе из цикла производится проверка – если в стеке осталось больше одного числа, программа выводит «error», в противном случае элемент, оставшийся в стеке, и завершает работу.

if (getSize(head) > 1)

printf("error");

else

printf("%d", head->number);

return 0;

1. Затем файл main.c с кодом написанной программы и отчёт о проведенной работе был загружен на github c помощью консоли:

* Создана новая ветка Ivanov\_2\_lr1

git checkout –b Ivanov\_2\_lr1

* Создана новая директория Ivanov\_2\_lr1 и файл main.c в ней, куда помещен код написанной программы

mkdir Ivanov\_2\_lr1

cd Ivanov\_2\_lr1

nano main.c

* Созданные файлы добавлены для загрузки

git add Ivanov\_2\_lr1/

* Добавлен комментарий

git commit –m “2 semestr. Lab 1”

* Файлы загружены на репозиторий Git

git push origin Ivanov\_2\_lr1

**Вывод:** в ходе работы получены навыки работы с функцией быстрой сортировки массивов qsort, проведен подсчёт времени работы данной функции, выполнен обзор стандартной библиотеки.