Отчёт лабораторной работы №4

по курсу «Типы и структуры данных»

Отчёт выполнил:

Вариант 23

г. Москва 2019

Отчет

### Цель работы:

Реализовать операции работы со стеком, который представлен в виде массива (статического или динамического) и в виде односвязного линейного списка; оценить преимущества и недостатки каждой реализации: получить представление о механизмах выделения и освобождения памяти при работе со стеком.

**Условие**:

Создать программу работы со стеком, выполняющую операции добавления, удаления элементов и вывод текущего состояния стека. Реализовать стек:

а) массивом;

б) списком.

Все стандартные операции со стеком должны быть оформлены подпрограммами. При реализации стека списком в вывод текущего состояния стека добавить просмотр адресов элементов стека и создать свой список или массив свободных областей (адресов освобождаемых элементов) с выводом его на экран.

Ввести арифметическое выражение типа: число|знак| ... число|знак| число, Вычислить значение выражения.

**Описание ТЗ:**

**1.Введение:**

**1.1 Наименование программы:**

Работа со стеком.

### 1.2. Срок выполнения работы:

2 недели с момента выдачи задачи.

**1.3 Краткая характеристика области применения:**

Программа позволяет вычислить значение введенного выражения, также позволяет продемонстрировать работу стека (стек-массив и стек-список).

**2. Основания для разработки:**

### 2.1. Заказчик

Силантьева Александра Васильевна

### 2.2. Исполнитель

Исполнитель – Савинов Егор Дмитриевич, студент группы ИУ7-34б.

### 2.3. Основания для разработки

Учебная программа 3 семестра программы ИУ7.

### 3. Назначение разработки

### 3.1 Общая концепция системы:

Программа позволяет вычислить значение введенного выражения, также позволяет продемонстрировать работу стека (стек-массив и стек-список).

### 3.2 Описание функциональности системы:

Программа позволяет вычислить значение введенного выражения, также позволяет продемонстрировать работу стека (стек-массив и стек-список).

### 4. Требования к программе:

### 4.1 Требования к информационным структурам:

Программа должна уметь отрабатывать любой ввод пользователя. При корректных данных следует выводить меню, таблицу эффективности. Если пользователь переполнит стек или попытается прочесть пустой стек, то нужно выдавать сообщение об ошибке.

### 4.2. Требования к функциональным характеристикам:

Программа должна выполнять:

1. Корректно отрабатывать команды меню.
2. Контролировать переполнение стека.
3. Позволить пользователю работать со стеком списком или массивом.
4. Произвести замер эффективности реализации стека массивом или списком.
5. При реализации стека списком выводить таблицу освобожденных адресов.

### 4.3. К надежности:

Программа должна уметь обрабатывать любой ввод, также при введении некорректных данных не должна завершаться аварийно.

### 4.4 Требования к составу аппаратного обеспечения:

У клиента интегрированная среда разработки должна поддерживать файлы с расширением .с. Программа не требует поддержки со стороны пользователя.

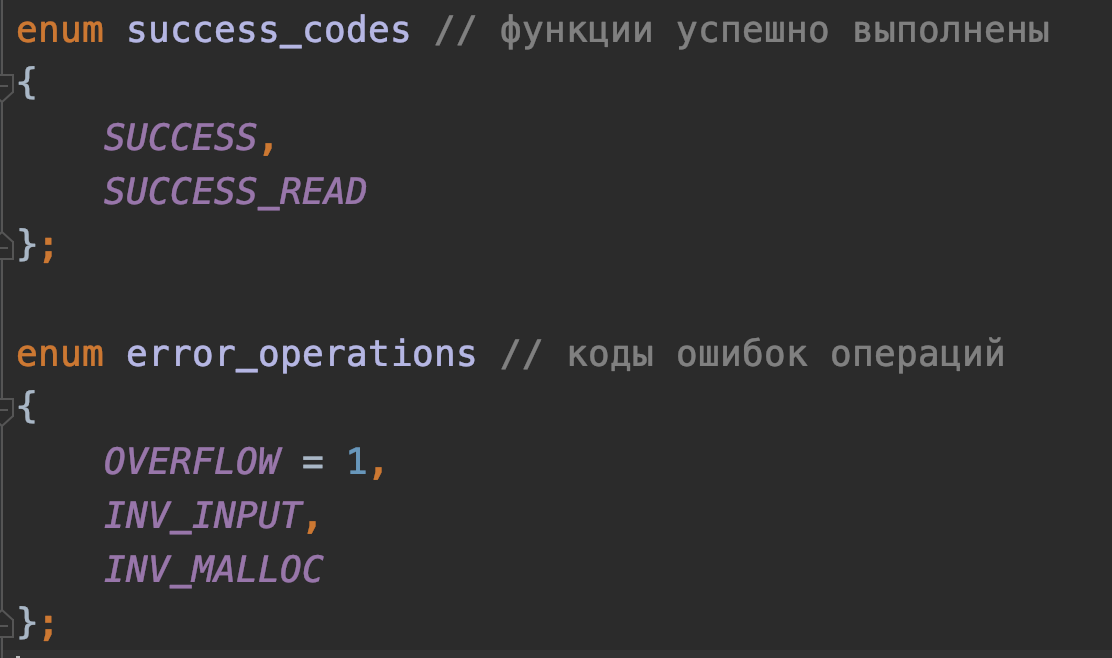
**Алгоритм работы программы:**

1. Заполнение стека.
2. Вывод результата введенного выражения.
3. Вывод таблицы эффективности реализации стека массивом или списком.

**Константы:**

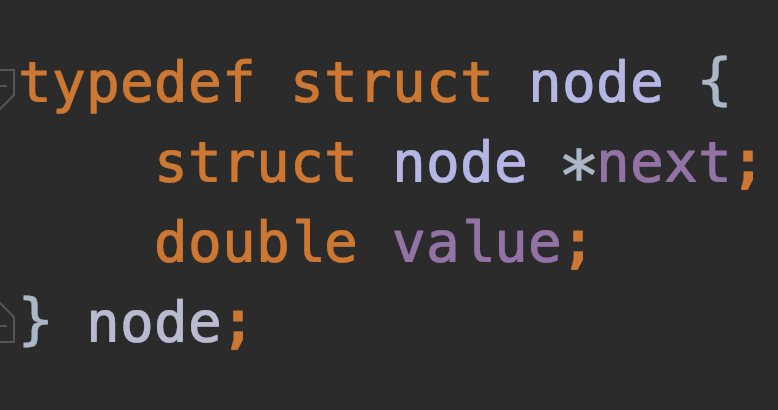


**Коды выполнения программы:**

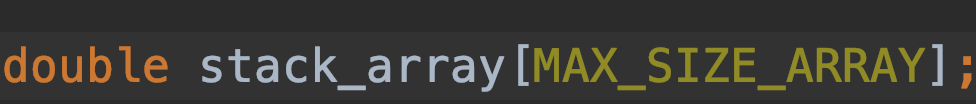
****

**Структура данных:**

**Структура узла списка**



**Реализация стека массивом**

****

**Способ обращения к программе**:

Программа состоит из шести исходных файлов main.c utills.c operation\_stack\_list.c operation\_stack\_array.c menu.c. Для запуска программы необходим исполняемый файл с расширением .exe. Запуск возможен с консоли или из любого другого проводника ОС.

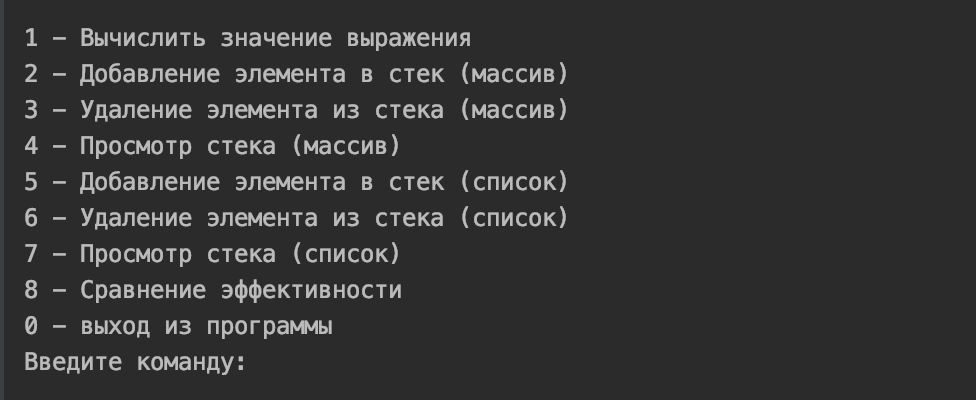
**Входные данные:**

Число, знак.

**Выходные данные:**

Элементы стека, список освобождённых адресов, значение введенного выражения.

Для пользователя предусмотрено меню действий:



**Функции:**

void print\_menu();

void get\_choice(int, int, int \*);

int push\_array\_menu(double \*\*, double \*, char \*, int \*);

Вход: адрес на текущий элемент, адрес последнего элемента, массив знаков, его длина.

Выход: код выполнения работы функции.

int push\_array(double \*\*, double \*, double);

Вход: адрес на текущий элемент, адрес последнего элемента, значения, которое нужно поместить в стек.

Выход: код выполнения работы функции.

int pop\_array(double \*\*, double const \*);

Вход: адрес на текущий элемент, адрес последнего элемента.

Выход: код выполнения работы функции.

void print\_array(double \*\*, double \*, char \*, int);

Вход: адрес на текущий элемент, адрес последнего элемента, массив знаков, его длина.

int push\_list\_menu(node \*\*, char \*, int \*, node \*\*, int \*, int \*);

Вход: голова списка, массив знаков, его длина, массив освобожденных адресов, его длина, переменная, которая отвечает за переполнение.

Выход: код выполнения работы функции.

int push\_list(node \*\*, double, double \*\*, int \*);

Вход: голова списка, элемент для вставки, массив освобожденных адресов, его длина.

Выход: код выполнения работы функции.

void free\_node(node \*\*);

Вход: голова списка.

int pop\_list(node \*\*, double \*\*, int \*, int \*);

Вход: голова списка, массив освобожденных адресов, его длина, переменная, которая отвечает за переполнение.

void print\_list(node \*\*, char \*, int, double \*\*, int, int \*);

Вход: голова списка, массив знаков, его длина, массив освобожденных адресов, его длина, переменная, которая отвечает за переполнение.

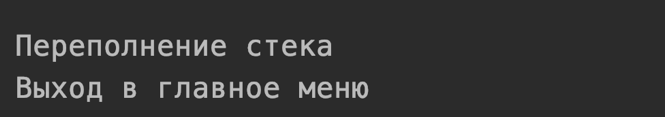
int calculation(node \*\*, double \*\*, int \*, int \*, char \*, int);

Вход: голова списка, массив знаков, его длина, массив освобожденных адресов, его длина, переменная, которая отвечает за переполнение.

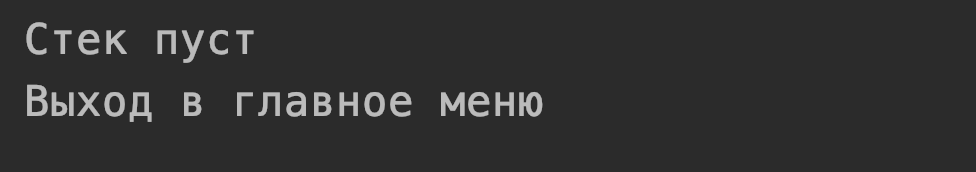
void time\_calc();

**Аварийные ситуации:**

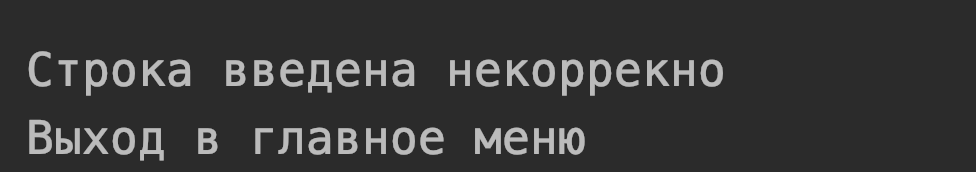
1. Переполнение стека:



1. Попытка прочитать пустой стек:



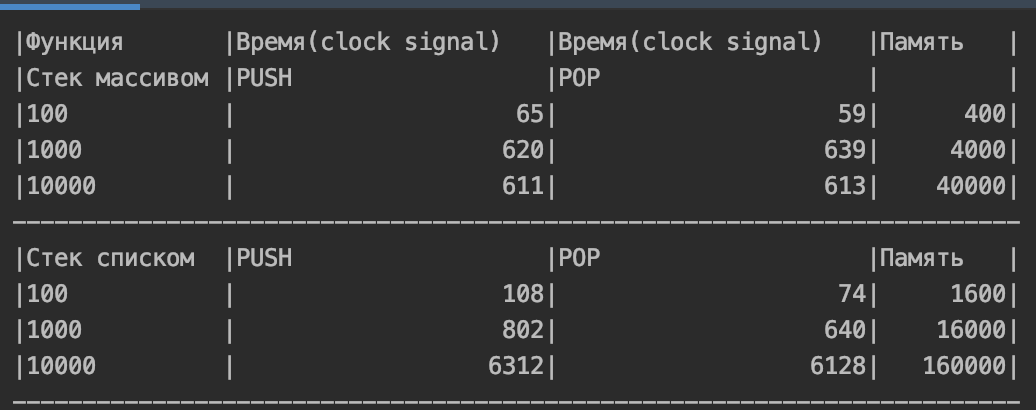
1. Некорректно введена строка выражения:



**Тесты**

|  |  |
| --- | --- |
| выражение | Результат выражения |
| 2+2\*2 | 6.000000 |
| 2+0/2 | Деление на ноль запрещено  Выход в главное меню |
| 2\*2+3\*4 | 14.000000 |
| Попытка подсчитать результа, после того, как результат был подсчитан | Стек пуст  Выход в главное меню |
| 2+2 | | Значение |адрес  |+2.000000|0x7ff398500228  |2.000000 |0x7ff398500218 |
| Удаление элемента | Освобожденные адреса:  0x7ff398500228  Вывод стека (список):  |Значение |адрес  |2.000000 |0x7ff398500218 |

**Оценка эффективности**

****

Работа со стеком массивом быстрее в 10 раз, чем со стеком списком. Стек массив в 4 раза занимает меньше памяти, чем стек список.

**Вывод:**

Стек в виде массива, выигрывает по памяти и времени обработки. Его выгодно использовать, если длина стека заранее известна. На стеки неопределенной длины лучше использовать стеки в виде списков.

**Ответы на вопросы.**

1. Что такое стек?

Стек – это последовательный список с переменной длиной, в котором включение и исключение элементов происходит только с одной стороны – с его вершины. Стек функционирует по принципу: последним пришел – первым ушел, Last In – First Out (LIFO).

2. Каким образом и сколько памяти выделяется под хранение стека при различной его реализации?

Стек списком:

Память выделяется динамически, при включении нового элемента

Стек массивом:

Выделяется непрерывная область памяти ограниченного размера, имеющая нижнюю и верхнюю границу. Добавление нового элемента происходит путём смещения указателя на последний элемент.

3. Каким образом освобождается память при удалении элемента стека при различной реализации стека?

Стек списком:

При реализации списком память из-под элемента освобождается при его удалении.

Стек массивом:

Память не освобождается, только смещается указатель.

4. Что происходит с элементами стека при его просмотре?

Указатель на элемент стека связывает предыдущий элемент. При просмотре стека доступны операции только с последним элементом стека поэтому, чтобы пройтись по стеку нужно обойти все элементы стека по указателям на предыдущий элемент.

5. Каким образом эффективнее реализовывать стек? От чего это зависит?

Маленькие стеки – массив, так как он выигрывает по времени обработки.

Стеки неопределенной длины – список.