**Министерство науки и высшего образования Российской Федерации**
**Федеральное государственное бюджетное образовательное учреждение высшего образования**
**«Московский государственный технический университет имени Н.Э. Баумана**
**(национальный исследовательский университет)»**
**(МГТУ им. Н.Э. Баумана)**

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии

# ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4 «РАБОТА СО СТЕКОМ»

Студент Цветков Иван Алексеевич

Группа ИУ7 – 33Б

*2020 г.*

# Описание условия задачи

Разработать программу работы со стеком, реализующую операции добавления и удаления элементов из стека и отображения текущегосостояния стека. Реализовать стек:
а) массивом;
б) списком.

Все стандартные операции со стеком должны быть оформлены отдельными подпрограммами. В случае реализации стека в виде списка при отображении текущего состояния стека предусмотреть возможность просмотра адресов элементов стека и создания дополнительного собственного списка свободных областей (адресов освобождаемой памяти при удалении элемента, который можно реализовать как списком, так и массивом) с выводом его на экран. Список свободных областей необходим для того, чтобы проследить, каким образом происходит выделение памяти менеджером памяти при запросах на нее и убедиться в возникновении или отсутствии фрагментации памяти.

# Описание технического задания

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

Доступные операции: «+» - сложить
«-» - вычесть
«*» - умножить
«/» - разделить

**Входные данные:**

**1. Целое число, представляющее собой пункт меню:**
целое число в диапазоне от 0 до 13

**3. Дополнительный ввод:** поле типа int или char в зависимости от требования

**Выходные данные:**

1. Результат выполнения команды
2. Сообщение об ошибке (при ее возникновении)

**Функции программы:**

1. Ввести выражение

Стек в виде списка:

2. Записать выражение в стек

3. Добавить элемент в стек

4. Удалить элемент из стека

5. Вывести текущее состояние стека

6. Произвести вычисление и вывести результат на экран

7. Распечатать массив освободившихся адрессов

Стек в виде массива:

8. Записать выражение в стек

9. Добавить элемент в стек

10. Удалить элемент из стека

11. Вывести текущее состояние стека

12. Произвести вычисление и вывести результат на экран

13. Вывести замеры времени и памяти

0. Выйти из программы

## Обращение к программе:

Запускается через терминал командой ./app.exe

## Аварийные ситуации:

1. Неверно введен пункт меню
   (не число или число меньшее 0 или больше 13)
2. Неверно введен символ при вводе выражения
   (не число или не знак)
3. Первый символ в выражении не число
   (нельзя начинать со знака)
4. Последний символ в выражении не число

(нельзя заканчивать знаком)
5. Ошибка выделении памяти
(при динамическом выделении)
6. Максимально возможный размер стека (как для массива, так и для списка) введен неверно
(не число или число меньшее 1 или большее 10000)
7. Переполнение стека
(достигнута граница стека)
8. Неверно введено число при добавлении элемента в стек
(не число)
9. Неверно введено число элементов для добавления в стек
(не число или число меньшее 1 или большее максимальной величины стека)
10. Стек не может быть создан, так как количество элементов при вводе выражения больше максимального размера стека

# Описание структуры данных

*Структура для хранения элемента стека, реализованного в виде списка*

```
typedef struct list
{
    int ind;
    int num;

    struct list *next;
} list_t;
```

Поля структуры:
1. int ind — номер элемента в стеке
2. int num — значение текущего элемента в стеке
3. struct list *next — указатель на следующий элемент стека

*Структура для хранения стека, реализованного в виде массива*

```
typedef struct array
```

```
{
    int *arr;

    int len;
} array_t;
```

Поля структуры:
1. int *arr — указатель на текущий элемент стека
2. int len — количество элементов в стеке

exp_t — структура, которая хранит массив типа char, в который записано все выражение (sym[1001]), len — длина выражения

```
typedef struct
{
    char sym[1001];

    int len;

} exp_t;
```

# Описание алгоритма

1. Выводится меню программы
2. Пользователь вводит номер любой команды, которой соответствует свое назначение
3. Ввод осуществляется, пока не будет совершена ошибка при вводе (аварийная ситуация) или пока не будет введен 0 (означает выход из программы)

# НАБОР ТЕСТОВ

| | **Название теста** | **Пользовательский ввод** | **Результат** |
|---|---|---|---|
| 1 | Некорректный ввод пункта меню | iu | Ошибка: пункты меню это числа от 0 до 13 |
| 2 | Некорректный ввод выражения | iu | Ошибка: неверный символ в выражении |
| 3 | Выражение содержит несколько знаков подряд | 1++2 | Ошибка: не может быть введено несколько знаков подряд |
| 4 | В выражении первый символ не число | +1+2 | Ошибка: первым должно быть введено число |
| 5 | В выражении крайний символ не число | 1+2+ | Ошибка: последним должно быть введено число |
| 6 | В выражении присутствует нецелое число | 1.33+5 | Ошибка: неверный символ в выражении |
| 7 | Превышен максимально возможный размер стека при добавлении элемента | (если максимально возможная величина стека 100)<br>попытка добавить 101 элемент в стек | Ошибка: стек переполнен |

| | | | |
|---|---|---|---|
| 8 | Превышен максимально возможный размер стека при загрузке выражения в стек | (если максимально возможная величина стека 100)<br>попытка добавить выражение с 101 элементом | Ошибка: стек не может быть создан, так как количество элементов превышает размер стека |
| 9 | При вводе максимального размера стека допущена ошибка<br>(не число или число, меньшее 1 и большее 10000) | iu<br>или<br>-1 | Ошибка: размер стека это число, не меньшее 1 и не большее 10000 |
| 10 | Неверно введено число при добавлении элемента в стек<br>(не число) | iu | Ошибка: неверно введен элемент для добавления |
| 11 | Неверно введено количество элементов для добавления в стек<br>(не число или число меньшее 1 или большее максимальной величины стека) | iu | Ошибка: неверно введено количество элементов в стеке |
| 12 | Невозможность загрузить выражение в стек, так как само выражение введено не было | Попытка ввести в стек выражение без его создания | Ошибка: выражение не введено |

| | | | |
|---|---|---|---|
| | | | |
| 13 | Попытка печати или удаления из стека массива, если он пуст | Пустой стек массив | Стек пуст |
| 14 | Попытка печати или удаления из стека списка, если он пуст | Пустой стек список | Стек пуст |
| 15 | Выражение можно ввести лишь 1 раз | Попытка ввести выражение еще раз | Ошибка: выражение уже введено |
| 16 | Выражение в стек массив можно ввести лишь один раз | Попытка ввести выражение в стек массив еще раз | Ошибка: стек уже создан |
| 17 | Выражение в стек список можно ввести лишь один раз | Попытка ввести выражение в стек список еще раз | Ошибка: стек уже создан |
| 18 | Массив освободившихся адресов пуст | Попытка распечатать массив освободившихся адресов, если он пуст | Массив освободившихся адресов пуст |
| 19 | Вычислить результат выражения в стеке в виде списка | Выражение<br>1+2+3<br>Команда<br>6 | Результат вычисления:<br>6 |
| 20 | Вычислить | Выражение | Результат |

| | | | |
|---|---|---|---|
| | результат выражения в стеке в виде массива | 1+2+3<br>Команда<br>12 | вычисления:<br>6 |
| 21 | Количество элементов для замера времени и памяти введено неверно<br>(не число или число меньшее 0 и большее 1000) | iu<br>или<br>-1 | Ошибка: неверно введено количество элементов в стеках |
| 22 | Вывод замеров времени и памяти (все значения введены корректно) | Команда<br>13<br>Количество элементов в стеках<br>100 | Вывод замеров времени для добавления, удаления и выполнения вычисления, а также количество затраченной памяти |
| 23 | Выход из программы | Команда<br>0 | Выход из программы, очистка консоли |

# Оценка эффективности

Время будет измеряться в тактах процессора на процессоре с частотой 35000000 Гц

Добавление элементов (в тиках)

| Размер | Массив | Список |
|---|---|---|
| 10 | 370 | 1341 |
| 100 | 3524 | 15218 |
| 500 | 5906 | 27780 |
| 1000 | 17762 | 85224 |

Удаление элементов (в тиках)

| Размер | Массив | Список |
|---|---|---|
| 10 | 369 | 915 |
| 100 | 3656 | 9953 |
| 500 | 4641 | 14825 |
| 1000 | 9070 | 30872 |

Вычисление выражения (в тиках)

| Размер | Массив | Список |
|---|---|---|
| 10 | 6845 | 7696 |
| 100 | 28728 | 36211 |
| 500 | 90832 | 115558 |
| 1000 | 175316 | 222821 |

Замеры памяти

| Размер | Массив | Список |
|---|---|---|

| | | |
|---|---|---|
| 10 | 48 | 160 |
| 100 | 408 | 1600 |
| 500 | 2008 | 8000 |
| 1000 | 4008 | 16000 |

# Ответы на контрольные 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

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

Если хранить как массив — либо в куче, либо на стеке (зависит от того, динамически или статический массив используется).
Если хранить стек как список, то память выделяется в куче.

Для каждого элемента стека, который хранится как список, выделяется на 4 или 8 байт больше, чем для элемента стека, который хранится как массив.
Данные байты использованы для хранения указателя на следующий элемент списка. (из-за этого либо 4 либо 8 байт)

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

Если хранить стек как массив (статический), то смещается только указатель на начало стека
Если хранить стек как список, то верхний элемент удаляется при помощи операции освобождения памяти для него и смещением указателя, который указывает на начало стека

***4. Что происходит с элементами стека при его просмотре?***

При просмотре стека мы имеем возможность добраться только до верхушки стека. Чтобы получить доступ к следующему элементу, необходимо удалить текущую верхушку

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

Стек эффективнее реализовать с помощью массива, так как он выигрывает в количестве занимаемой памяти (если массив динамический) и во времени обработки стека
Хранение с помощью списка может выигрывать, если только стек реализован с помощью статического массива, так как в данном случае размер памяти под список ограничен размером оперативной памяти (хранится в куче), а для статического массива — ограничена размером стека.

# Вывод

В случае необходимости реализации стека на компьютере, его стоит реализовывать на динамическом массиве, так он имеет преимущества над статическим массивом и списком

Статический массив ограничен по памяти, так как располагается в стеке самого компьютера, в то время как динамический массив ограничен лишь объемом оперативной памяти компьютера

Список, в свою очередь, занимает больше количества памяти (примерно в 4 раза), так как ему, помимо самого элемента, нужно хранить указатель на следующий элемент списка, а также скорость работы с элементами списка занимает больше времени:

при удалении (примерно в 3 раза)

при добавлении (примерно в 4-5 раз)

при подсчете выражения (примерно в 1.3-1.4 раза)