Министерство науки и высшего образования Российской Ф
Федеральное государственное бюджетное образовательное вы
образования
«Московский государственный технический университ
имени Н.Э. Баумана
(национальный исследовательский университет)» (МГ
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 Назиров Илхомджон Вохидович

Группа ИУ7 – 35Б

Преподаватель:

2021 г.

Описание условия задачи

Создать программу работы со стеком, выполняющую операции добавление, 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.

При тестировании программы необходимо:

- проверить правильность ввода и вывода данных (в том числе, отследить попытки ввода данных, неверных по типу);
- обеспечить вывод сообщений при отсутствии входных данных («пустой ввод»);
- проверить правильность выполнения операций;
- обеспечить вывод соответствующих сообщений при попытке удаления элемента из пустого стека;
- отследить переполнение стека.

При реализации стека в виде списка необходимо:

- ограничить доступный объем оперативной памяти путем указания: максимального количества элементов в стеке; максимального адреса памяти, превышение которого будет свидетельствовать о переполнении стека;
- следить за освобождением памяти при удалении элемента из стека.

Входные данные:

1. Пункт Меню choice {0.,..,12}

```
        Menu

-- Array Stack --
        - 1 - Add element in stack
        - 2 - Remove element from stack
        - 3 - Generate the data
        - 4 - Solve operation from stack
        - 5 - Print stack

-- List Stack --
        - 6 - Add element in stack
        - 7 - Remove element from stack
        - 8 - Generate the data
        - 9 - Solve operation from stack
        - 10 - Print stack

-- Compare the efficient of array and list stack --
        - 11 - Compare

-- Exit Program --
        - 0 - Exit

        - 12 - Print the address of deleted element
```

2.**Выражение в виде целое число + знак**

Выходные данные:

4,9 - Результат Вычислить значение выражения (сложение и вычитание).

5,10 - Вывод текущего состояния стеков

11 - Вывод время и память, затраченные на сортировку стеков в разных реализациях (массив и список)

12 - Вывод массив свободных областей

Функции программы:

1,6 - Добавить элементы в стеки (количество)

2,7 - Удалить элементы из стеков

3,8 - Заполнение стека рандомными данными

4,9 - Вычислить значение выражения.

5,10 - Вывод текущего состояния стеков

11 - Вывести время и память, затраченные на сортировку стеков в разных реализациях (массив и список)

0 – Выйти

Обращение к программе:

Запускается через терминал с помощью команды ./app.exe.

Аварийные ситуации:

1. Некорректный ввод номера команды.

На входе: число, большее чем 12 или меньшее, чем 0.

На выходе: сообщение «Ошибка: неверно введен номер пункта меню»

2. Совершение вычислить значение выражения при пустом стеке. На входе: целое число 4 (номер команды). На выходе: сообщение «Ошибка: пустой стек»

3. Добавление элемента в стек, когда он заполнен полностью. На входе: элемент стека. На выходе: сообщение «Ошибка: произошло переполнение стека»

Описание структуры данных

Реализация стека с помощью линейного односвязного списка

```
typedef struct node_el {
    int number; // число
    int obelus; // знак числа
    struct node_el *prev; // ссылка на следующий список

} node;

typedef struct {
    node *ptr; // head списка
} stack_list;
```

Реализация стека с помощью массива

```
typedef struct {
    int **arr_stack; // массив элементов
    int top_size; // количество элементов в стеке
    int size; // Максимальный размер стека
} stack_array;
```

Описание алгоритма

Разделяю входную строку на число и знак после записываю в стек далее после того как записал в стек при каждом проходе я беру число и вычитаю или делаю сложение с предыдущим числом и так пока стек не станет пустым.

НАБОР ТЕСТОВ

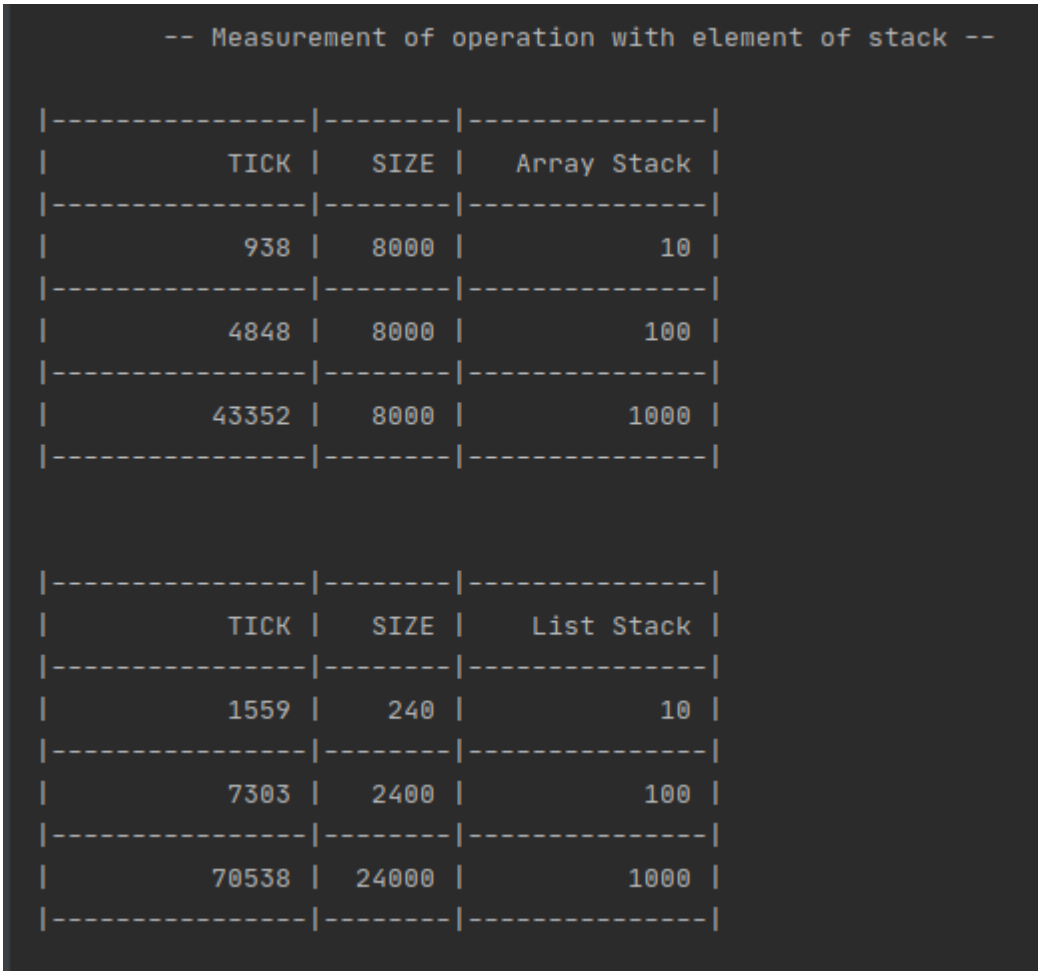
		Название теста **Пользовательский** **Результат**	

1	Добавление элемента в заполненный стек	Добавление элемента в стек (1 - стек как массив, 2 - стек как список)	Ошибка: произошло переполнение стека
2	вычислить значение выражения при пустом стеке	Команда 4 или 9	Ошибка: пустой стек
3	Добавление элементов в стек (команда 1)	1) 1 ,6- стек как массив / список 2) Ввести количество элементов 3) Ввести элементы	Элементы успешно добавлены в стек
4	Удаление элементов из стека (команда 2)	1)2,7- Выбрать стек 2) Ввести количество удаляемых элементов	Элементы успешно удалены из стека

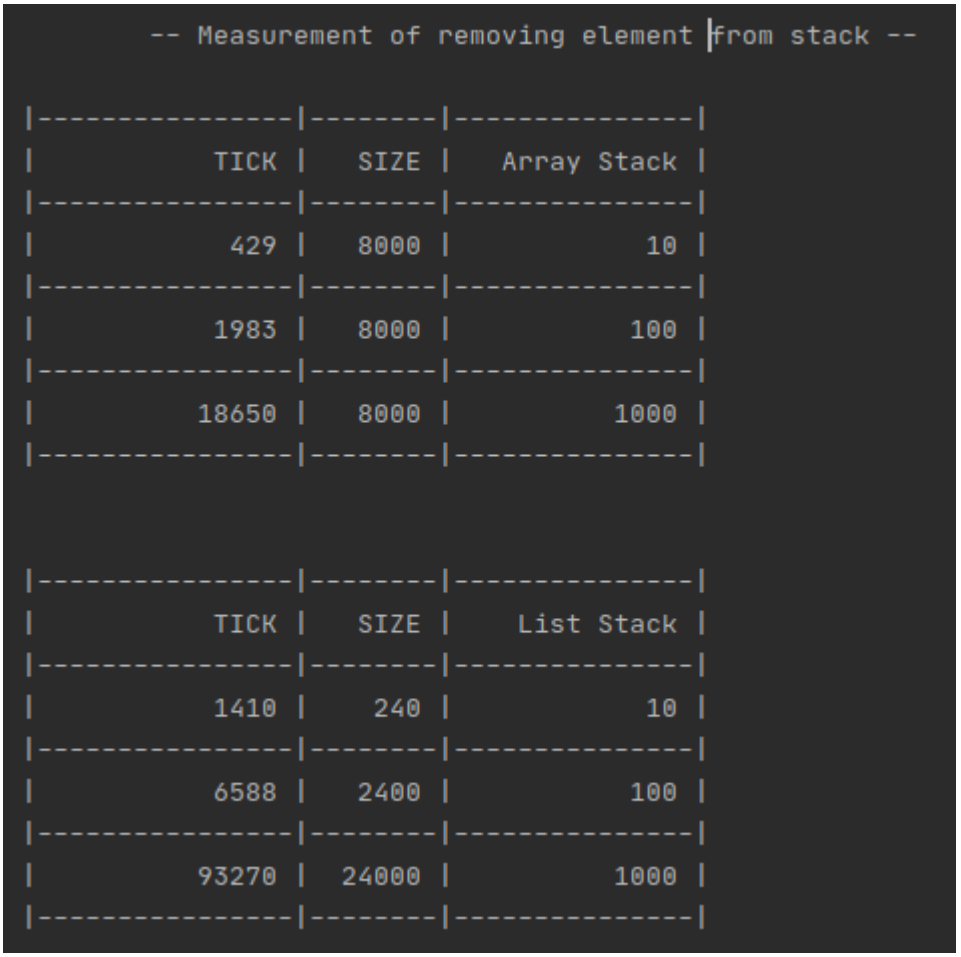
5	Вывод текущего состояния стеков на экран (команда 5)	Вызов команды (5).	Все стеки выведены на экран + массив свободных указателей
6	вычислить значение выражения (команда 4)	Вызов команды (4).	Вычисление прошла успешно
7	Вывод сравнение стеков на экран (команда 11)	Вызов команды (11).	Таблица

Оценка эффективности по времени и памяти

Measurement of pushing element from stack



Measurement of removing element from stack



Measurement of operation with element of stack

```
        -- Measurement of operation with element of stack --

|----------------|--------|---------------|
|           TICK |   SIZE |   Array Stack |
|----------------|--------|---------------|
|            938 |   8000 |            10 |
|----------------|--------|---------------|
|           4848 |   8000 |           100 |
|----------------|--------|---------------|
|          43352 |   8000 |          1000 |
|----------------|--------|---------------|


|----------------|--------|---------------|
|           TICK |   SIZE |    List Stack |
|----------------|--------|---------------|
|           1559 |    240 |            10 |
|----------------|--------|---------------|
|           7303 |   2400 |           100 |
|----------------|--------|---------------|
|          70538 |  24000 |          1000 |
|----------------|--------|---------------|
```

Время:

По времени добавление элемента массив эффективнее чем список на 45-60% .
По времени удаление элемента массив эффективнее чем список на 70-75%.
По времени вычислить значение выражения элемента массив эффективнее чем список на 40-46%.

По памяти:

По памяти реализация стека с помощью списка эффективнее в случае когда массива заполнен менее чем на ~35% в моем случае при размер измерение на 1000 элементов начинаю с 350 элемента массив побеждает по памяти списка .

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. Стек функционирует по принципу: LIFO - последним пришел – первым ушел,

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

Если хранить стек как список, то память выделяется в куче. Если хранить как массив — либо в куче, либо на стеке (зависит от того, динамический или статический массив используется). Для каждого элемента стека, который хранится как список, выделяется на 4 или 8 байт (если брать современные ПК) 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) .

Для массива выделяется память одной кучей в начале программы а для списка каждый раз во время добавление элемента выделяются память.

3. Каким образом освобождается память при удалении элемента стека при различной реализации 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.
При хранении стека как массив, память освобождается при завершении программы.

4. Что происходит с элементами стека при его просмотре?

Элементы стека удаляются, так как каждый раз достается верхний элемент стека, чтобы посмотреть следующий.

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

Если в задаче заранее известно максимальный размер вводимых данных и массив заполнен больше чем 35% во-время работы программы лучше использовать массив иначе лучше использовать список.

Вывод

При реализация стека в виде массива требуется меньше времени на обработку. Это связано с тем, что при реализации массивом доступ к нужному элементу получить проще, требуется лишь передвинуть указатель. Если реализовать в виде списка, то требуется время для удаления верхнего элемента (верхушки стека) и перестановки указателя.

Можно сделать вывод, что для хранения стека если в задаче заранее известно максимальный размер вводимых данных и массив заполнен больше чем 35% во-время работы программы лучше использовать массив иначе лучше использовать список.