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 Козлова Ирина Васильевна

Группа ИУ7 – 32Б

2020 г.

Оглавление

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.

Указания к выполнению работы

Интерфейс программы должен быть понятен неподготовленному пользователю. При разработке интерфейса программы следует предусмотреть:

- • указание формата и диапазона вводимых данных,
- • блокирование ввода данных, неверных по типу,
- • указание операции, производимой программой:
  - о добавление элемента в стек,
  - о удаление элемента из стека,
  - о вычисление (обработка данных);
- • наличие пояснений при выводе результата.

Кроме того, нужно вывести на экран время выполнения программы при реализации стека списком и массивом, а также указать требуемый объем памяти. Необходимо так же выдать на экран список адресов освобождаемых элементов при удалении элементов стека.

При тестировании программы необходимо:

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

При реализации стека в виде списка необходимо:

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

ОПИСАНИЕ ТЕХНИЧЕСКОГО ЗАДАНИЯ

Проверить правильность расстановки скобок трех типов (круглых, квадратных и фигурных) в выражении.

Входные данные:

1. **Целое число, представляющее собой номер команды**: целое число в диапазоне от 0 до 13.
2. **Командно-зависимые данные:**
   целочисленные значения (количество элемента стека)
   строковые константы (элементы стека)

Выходные данные:

1. Результат выполнения определенной команды.
2. Характеристика сравнения вариантов обработки стека различными способами.

Функции программы:

1. Добавить элемент в стек.
2. Добавить элемент в стек.
3. Удалить элемент стека.
4. Вывод текущего состояние стека.
5. Проверить корректное расположение скобок.
6. Добавить элемент в стек.
7. Удалить элемент стека.
8. Вывод текущего состояние стека.
9. Проверить корректное расположение скобок.
10. Вывод текущего состояние стека.
11. Вывод массива освобожденных адресов.
12. Статистика работы различных функций.
0. Выход из программы.

Обращение к программе:

Запускается через терминал. Так же можно собрать программу используя makefile и запустить ее с помощью команды run.

Аварийные ситуации:

1. Некорректный ввод номера команды.
   На входе: число, большее чем 13 или меньшее, чем 0.
   На выходе: сообщение «ERROR!!! Invalid command entered, please re-enter!!!»
2. Некорректный ввод номера команды.
   На входе: пустой ввод.
   На выходе: сообщение «Invalid command entered, please re-enter!!!»
3. Некорректный ввод размера стека.
   На входе: отрицательное число, нуль, число, превышающее максимально допустимое число для количества элементов стека или буква..
   На выходе: сообщение «ERROR!!! There were problems filling in the table.»
4. Попытка создать новый стек, при имеющемся в программе.
   На входе: команда создания стека.
   На выходе: сообщение «Stack is.»
5. Совершение команды со стеком, при его отсутствия.
   На входе: целое число в диапазоне от 2 до 5 или от 6 до 12 (номер команды).
   На выходе: сообщение «Stack does not exist.»
6. Добавление элемента в стек, когда он заполнен полнстью.

На ввод: элемент стека.
На выход: сообщение о том, что стек заполнен полностью.

Описание структуры данных

func_var именованный типа для работы с функциями и для того, чтобы занимать меньше памяти (удобно будет заменить его в случае смены ТЗ):

```
typedef int func_var;
```

Реализация стека с помощью линейного односвязного списка.

```
typedef struct liststack
{
    char data;
    func_var ind;
    struct liststack *next;
} liststack_r;
```

Поля структуры:

data – элемент стека

ind – индекс узла списка

****next*** – указатель на следующий элемент списка

Реализация стека с помощью массива.

Вспомогательная структуры — массив адресов (свободных).

```
typedef struct
{
    size_t *arr;
    func_var capa;
    func_var ind;
} arr_r;
```

Поля структуры:

****arr*** - указатель на массив свободных адресов

capa **–** *м*аксимальное количество элементов в массиве

ind **–** индекс текущего элемента списка

Структура данных — стек с помощью массива.

```
typedef struct
{
    func_var top;
    func_var capa;
    char *arr;
} arrstack_r;
```

Поля структуры:

top — голова стека

capa — *к*оличество элементов стека

****arr*** – указатель на массив элементов стека

Описание алгоритма

1. Выводится меню данной программы.
2. Пользователь вводит номер команды из предложенного меню.
3. Пока пользователь не введет 0 (выход из программы), ему будет предложено вводить номера команд и выполнять действия по выбору.

Набор тестов

	Название теста	**Пользовательский ввод**	**Результат**
1	Некорректный ввод команды	45	Invalid command entered, please re-enter!!!
2	Пустой ввод	Пустой ввод.	Invalid command entered, please re-

			enter!!!
3	Добавление элемента в стек	Ввод команды на добавление элемента.	Stack is full!
4	Не существует стека (при вводе команд 2-5 и 7-11)	Ввод команды - операции стека, при не созданном стеке до этого (действует как для стека в виде массива так и для стека в виде списка).	Stack does not exist.
5	Создание стека (команда 1)	Ввод неверного размера стека (максимального).	Input invalid number.
6	Создание стека (команда 1)	Ввод неверного размера стека (который, хотим заполнить) меньше 0 или больше заданного максимального.	Input invalid number.
7	Создание стека (команда 1)	Ввод верных размеров	Стек успешно создан.
8	Добавление элемента в стек (команда 2)	Введен элемент.	Элемент добавлен в стек.
9	Удаление элемента из стека (команда 3)	Вызов команды.	Удаленный элемент выведен на экран.
10	Вывод стека на экран (команда 4)	Вызов команды.	Выведен стек на экран.
11	Вызов команды 5. (проверка на скобки)	Стек содержит: qwery	TRUE
		Стек содержит: wef(v)fwe	TRUE
		Стек содержит:	TRUE

		we(0)f[]wef{d}	
		Стек содержит: {}{(0)}	TRUE
		Стек содержит: []dw{(fwe)}	TRUE
		Стек содержит: {[(])}	FALSE
		Стек содержит: wef{wef	FALSE
		Стек содержит: {efw}}	FALSE
		Стек содержит: }fewf{	FALSE
		Стек содержит: efi(0))))	FALSE
		Стек содержит:{}{]	FALSE

Стек представлен в виде списка.

12	Создание стека (команда 6)	Ввод неверного размера стека (максимального).	Input invalid number.
13	Создание стека (команда 6)	Ввод неверного размера стека (который, хотим заполнить) меньше 0 или больше заданного максимального.	Input invalid number.
14	Создание стека (команда 6)	Ввод верных размеров	Стек успешно создан.
15	Добавление элемента в стек (команда 7)	Введен элемент.	Элемент добавлен в стек.
16	Удаление элемента из стека (команда	Вызов команды.	Удаленный элемент выведен на экран.

	8)		
17	Вывод стека на экран (команда 10)	Вызов команды.	Выведен стек на экран.
18	Вызов команды 9. (проверка на скобки)	Стек содержит: qwery	TRUE
		Стек содержит: wef(v)fwe	TRUE
		Стек содержит: we(0)f[]wef{d}	TRUE
		Стек содержит: {}{(0)}	TRUE
		Стек содержит: []dw{(fwe)}	TRUE
		Стек содержит: {[(])}	FALSE
		Стек содержит: wef{wef	FALSE
		Стек содержит: {efw}}	FALSE
		Стек содержит: }fewf{	FALSE
		Стек содержит: efi(0))))	FALSE
		Стек содержит:{}{]	FALSE
19	Вывод адресов пустых элементов (команда 11)	Стек существует, но не было освобождено ни одного элемента	Array is empty.
20	Вывод адресов пустых элементов (команда 11)	Стек существует и был освобожден хоть один элемент	Вывод ажресов.

Оценка эффективности

Измерения эффективности сортировок будут производиться в тактах процессора, с помощью специальной функции будут делаться замеры

количества тактов, поэтому погрешность данного измерения минимальна. Частота процессора 1990000000 Гц.

Команда добавления элемента в стек.

Размер	**Список**	**Массив**
10	443	204
100	492	304
500	638	315
1000	925	379

Команда удаления элемента из стека.

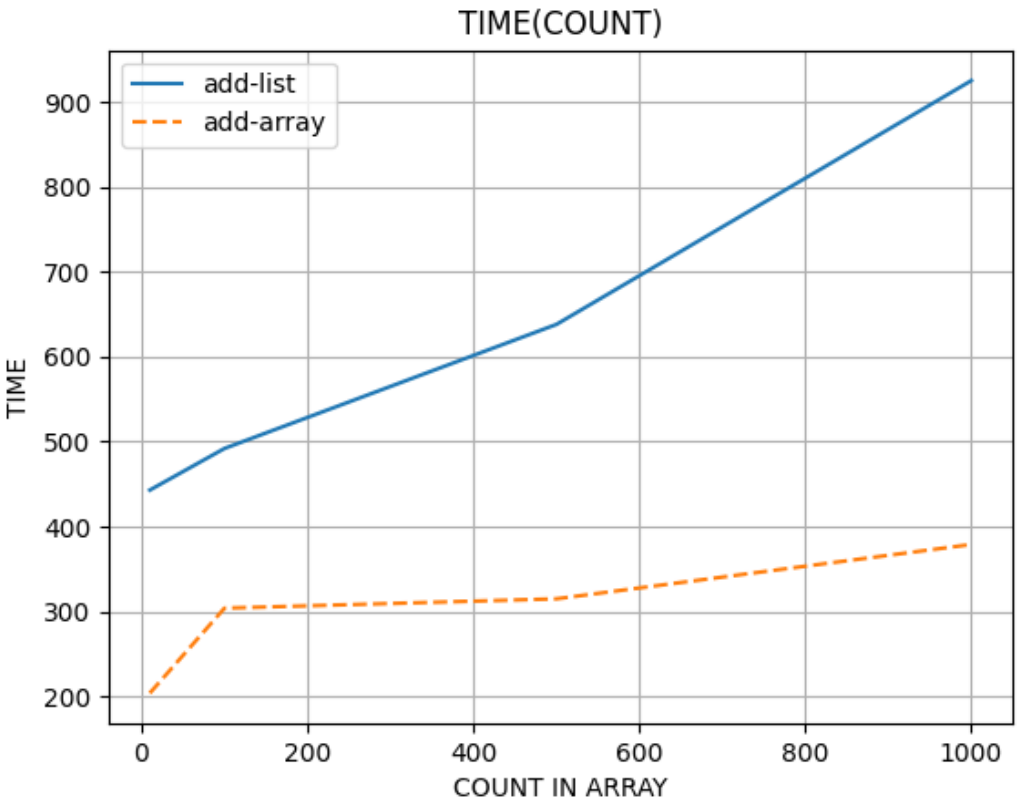
Размер	**Список**	**Массив**
10	1132	503
100	1249	531
500	1902	589
1000	2309	622

Команда проверки на правильность расстановки скобок.

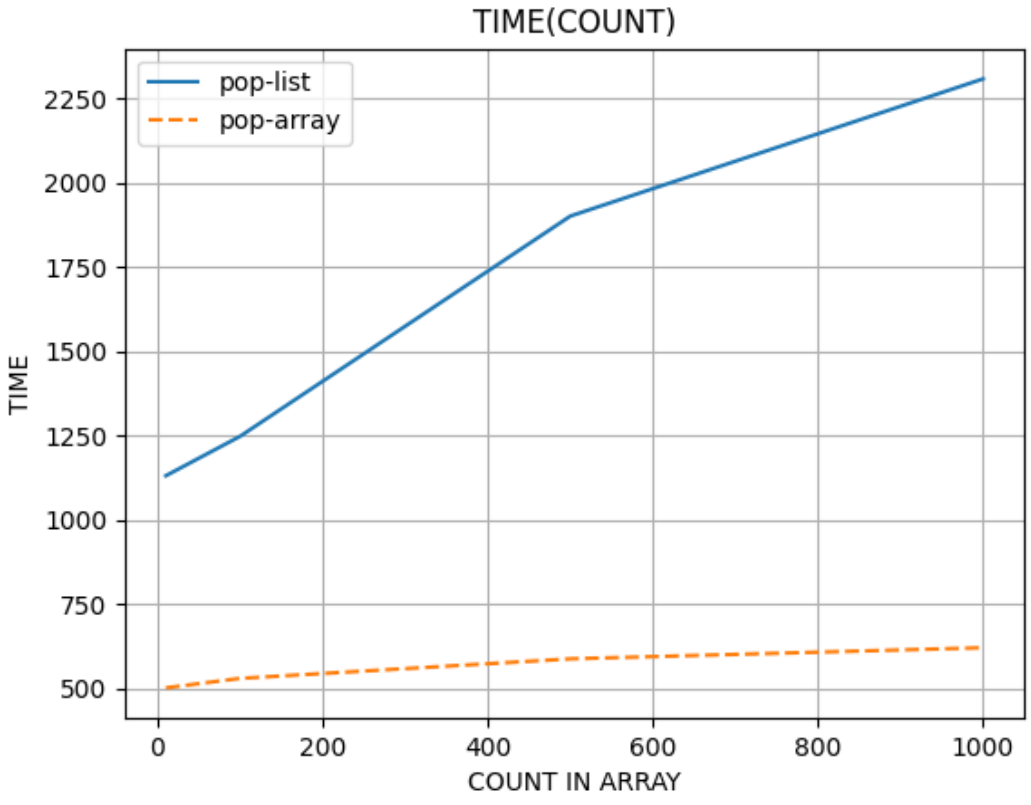
Размер	**Список**	**Массив**
10	4197	1389
100	10305	3435
500	27811	9315
1000	48765	16255

Графики оценки эффективности

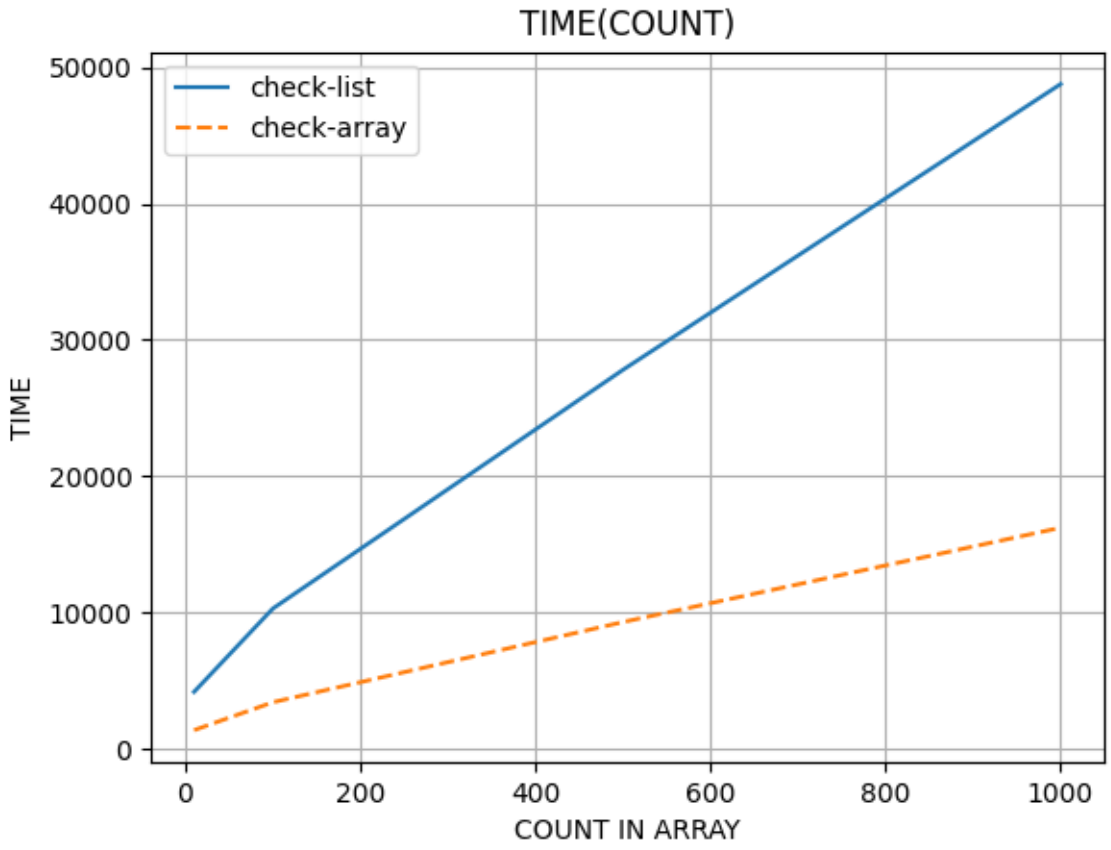
Команды добавления элемента в стек



Команды удаления элемента из стека



Команды проверки скобок в стеке.



Оценка эффективности по памяти

Размер	Список	Массив
10	130	48
100	1300	480
500	7500	2400
1000	13000	4800

Из данной таблицы можно сделать вывод, что при полном заполнении стека, стек в виде списка проигрывает по памяти стеку в виде массива. Но если же стек будет заполнен не на 100%, а до 40%, то стек в виде списка будет выигрывать по памяти у стека в виде массива, что видно из формул и таблице ниже.

Стек в виде массива: (указатель на char (8 байт) + кол-во элементов * (int, 4 байта))

Стек в виде списка: (указатель на следующий элемент (8 байт) + (char, 1 байт) + (int, 4 байт)) * кол-во элементов

Максимальное количество в стеке	Количество в стеке	Список	Массив
100	10	480	480
100	30	390	480
100	40	520	480
100	50	650	480

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 хранение стека 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, динамически 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)

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.
Если хранить стек как массив, то смещается только указатель на начало стека.

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 это зависит?

Стек эффективнее реализовать с помощью массива, так как он выигрывает в количестве занимаемой памяти (если это классический случай) 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

Если стек реализовать массивом, то он будет выигрывать во памяти (в моем случае ~2-4 раза). Это связанно с тем, что для хранения стека в виде списка требуется память, чтобы хранить указатели. Но если стек заполнен до 40%, то стек в виде списка будет выигрывать. (см. Таблицы и формулы выше)

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

Можно сделать вывод, что для хранения стека лучше использовать массив (если процент заполнения от 40 до 100), это выгоднее и по памяти и по времени. Но если стек будет заполнен до 40%, то выгоднее использовать список.