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»

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4
«РАБОТА СО СТЕКОМ»

Студент: Лысцев Никита Дмитриевич

Группа: ИУ7-33Б

Студент _____ Лысцев Н.Д

подпись, дата *фамилия,и.о.*

Преподаватель _____ Барышникова М. Ю

подпись, дата *фамилия,и.о.*

Оценка _____

2022 г

Оглавление

1. Цель работы

Цель работы -- реализовать операции работы со стеком, который представлен в виде массива (статического или динамического) и в виде односвязного списка, оценить преимущества и недостатки каждой реализации, получить представление о механизмах выделения и освобождения памяти при работе с динамическими структурами данных.

2. Описание условия задачи

Разработать программу работы со стеком, реализующую операции добавления и удаления элементов из стека и отображения текущего состояния 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.

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

3. Описание технического задания

3.1. Входные данные

Целое число от 0 до 13 – номер пункта меню, позволяющий пользователю взаимодействовать со стеком, реализованным массивом, и стеком, реализованным односвязным линейным списком.

Целое число – размерность стека для его реализации с помощью массива (не более 10000 элементов).

Элемент стека – любой один символ.

Для проверки выражения, содержащего скобки, на сбалансированность, само выражение – массив символов, длины не более 10000 элементов.

3.2. Выходные данные

В зависимости от выбора пользователя либо выход из программы, либо выполнение одной из предложенной по меню взаимодействия возможности.

3.3. Описание задачи, реализуемой программой

Задача программы – с помощью меню взаимодействия осуществить работу с помощью такой структурой данных, как стек.

3.4. Способ обращение к программе

Ввод и вывод всех данных осуществляется через консоль.

Для вывода времени работы программы для проверки расстановки скобок в выражении, а также времени работы функций для работы со стеком (добавление, удаление элемента из стека, очистка стека, создание стека), выражение, содержащее скобки, считывается из заранее подготовленного текстового файла.

3.5. Описание возможных аварийных ситуаций и ошибок пользователя

Аварийные ситуации:

- Пустое поле ввода – ничего не произойдет, программа будет ждать ввода пользователя;
- При вводе элемента стека (это один символ) ввод нескольких символов – будет считан первый символ из всей введенной последовательности символов.

Ошибки пользователя:

- Некорректный ввод: ввод неверного пункта меню, ввод неверной размерности для создания стека как массива, добавление в переполненный стек, удаление из пустого стека.

4. Описание внутренних структур данных

Элемент стека для реализации стека как массивом, так и односвязным списком описывается структурой типа «data_t».

```
typedef struct data_t data_t;

struct data_t
{
    char symbol;
};
```

Листинг 1. Структура «data_t», содержащая в себе описание одного элемента стека

Поля структуры «data_t»:

1. symbol – символ длины один, элемент стека.

Реализация стека с помощью одномерного динамического массива описывается структурой «stack_array_t».

```
typedef struct stack_array_t stack_array_t;

struct stack_array_t
{
    data_t *array_data;
    int top;
    int capacity;
};
```

Листинг 2. Реализация стека как массива с помощью структуры типа «stack_array_t»

Поля структуры «stack_array_t»:

1. array_data – массив структур типа «data_t», содержащий в себе элементы стека;
2. top -- целочисленная переменная, содержащая в себе индекс вершины стека, то есть индекс последнего добавленного элемента;
3. capacity – целочисленная переменная, содержащая в себе максимальный размер стека;

Реализация стека с помощью линейного односвязного списка описывается структурой типа «stack_list_t»

```
typedef struct stack_list_t stack_list_t;

struct stack_list_t
{
    data_t data;
    stack_list_t *next;
};
```

Листинг 3. Реализация стека как линейного односвязного списка с помощью структуры типа «stack_list_t»

Поля структуры «stack_list_t»:

1. data – переменная структурного типа «data_t», содержащая в себе элемент стека;
2. next – указатель на следующий элемент стека.

Также при реализации стека линейным односвязным списком для вывода ранее освобожденных адресов в результате удаления элемента из стека или очищения стека используется структура типа «free_addr_t».

```
typedef struct free_addr_t free_addr_t;

struct free_addr_t
{
    size_t *array_free_addr[MAX_FREE_ADDR_SIZE];
    int top;
};
```

Листинг 4. Структура типа «free_addr_t»

Поля структуры «stack_list_t»:

1. array_free_addr – статический массив указателей, содержащий ранее освобожденные адреса для реализации стека односвязным списком. Длина этого массива не превышает MAX_FREE_ADDR_SIZE = 10000 элементов;
2. top – индекс последнего добавленного элемента в массив адресов.

4.1. Описание алгоритма

Взаимодействие пользователя с программой осуществляется через консоль с помощью специального меню, в котором пользователю предлагается выполнить то, или иной действие.

В начале меню в одном предложении кратко изложена суть программы. Далее в пунктах с 1 по 13 кратко изложены возможности программы. Пункт с номером 0 служит для принудительного завершения работы с программой.

```
-------------------------------------------------------------------------|
Программа для проверки правильности скобок в выражении с помощью стека.
Стек реализован двумя способами:
а) массивом;
б) списком.

Операции со с стеком в виде массива:
1  - создать стек;
2  - добавить элемент в стек;
3  - удалить элемент из стека;
4  - вывести текущее состояние стека;
5  - проверить правильность расстановки скобок в выражении;
6  - очистить стек;

Операции со с стеком в виде односвязного линейного списка:
7  - создать стек;
8  - добавить элемент в стек;
9  - удалить элемент из стека;
10 - вывести текущее состояние стека;
11 - проверить правильность расстановки скобок в выражении;
12 - очистить стек;

13 - вывод времени работы программы и
     различных функций для работы со стеком;

0  - выйти из программы.
-------------------------------------------------------------------------|
Выберите пункт меню:
```

Листинг 5. Меню взаимодействия

4.2. Ограничения на входные данные

Максимальная вместимость стека в случае обоих реализаций – 10000 элементов.

Максимальная вместимость массива свободных адресов – 10000 элементов.

Элементом стека является символ, его длина, соответственно, равна единице. При попытке ввести количество символов больше, чем один, будет считан и воспринят как элемент стека только первый символ.

Попытка удаления из пустого стека или добавление в полностью заполненный стек обрабатывается программой, выдавая соответствующие сообщения, но не завершая программу с ненулевым кодом возврата.

При вводе любых не валидных данных программа выдает соответствующее сообщение и завершает работу с ненулевым кодом возврата.

Перед проверкой выражения на правильность расстановки скобок стек очищается.

5. Оценка эффективности работы алгоритмов

Было проведено 10000 измерений и взято среднее из полученных временных интервалов.

Время работы функций для работы со стеком (микросекунды)

	Стек как массив	Стек как список
Добавление элемента в стек	0.021200	0.030600
Удаление элемента из стека	0.023300	0.024600
Очистка стека	0.023700	0.033700
Создание стека	0.023600	0.023800

Время работы программы (в микросекундах), проверяющей сбалансированность скобок в выражении

Количество элементов в стеке	Стек как массив	Стек как список
100	0.29	0.92
1000	2.11	8.95
5000	9.84	36.77
7000	13.20	53.41
10000	18.76	75.31

Размер памяти (в байтах), необходимый для хранения стека

Количество элементов в стеке	Стек как массив	Стек как список
100	108	900
1000	1008	9000
5000	5008	45000
7000	7008	63000
10000	10008	90000

6. Выводы

Анализируя данные из таблиц выше, можно сказать, что реализация стека с помощью односвязного списка проигрывает как по времени работы, так и по памяти.

Время работы функций для обработки стека в обоих реализациях практически одинаково.

Время обработки выражения, содержащего скобки, у реализации стека массивом на всех размерностях самого выражения примерно в 3-4 раза меньше, чем обработка этого же выражения с использованием такой реализации стека, как односвязный список.

Хранение в памяти стека как односвязного списка также сильно уступает хранению в памяти стека в виде массива: стек как список занимает в памяти в 9 раз больше байт, чем стек как массив.

7. Набор тестов

№ Теста	Входные данные	Выходные данные	Результат
01	Пункт меню – 1 Стек ранее не был создан Ввод валидной размерности	Сообщение об успешном создании стека	Ожидание следующего действия.
02	Пункт меню – 1 Стек ранее не был создан Размерность стека <= 0 или> 10000	Сообщение о неверно введенной размерности стека	Завершение программы с ненулевым кодом возврата.
03	Пункт меню – 1 Стек ранее был создан	Сообщение о том, что стек был создан ранее	Ожидание следующего действия.
04	Пункт меню – 2, 8 В стеке 10000 элементов	Сообщение о том, что стек переполнен	Ожидание следующего действия.
05	Пункт меню – 2, 8 В стеке <10000 элементов Ввод символа	Добавление элемента в стек	Ожидание следующего действия.
06	Пункт меню – 3, 9 В стеке 0 элементов	Сообщение о том, что стек уже пуст	Ожидание следующего действия.
07	Пункт меню – 3, 9 Стек ранее не был создан В стеке >0 элементов	Сообщение об успешном удалении элемента из стека	Ожидание следующего действия.
08	Пункт меню – 4, 10 Стек не пустой	Вывод текущего состояния стека. Для пункта 10 вывод непустого массива ранее освобожденных адресов	Ожидание следующего действия.
09	Пункт меню – 4, 10 Стек пустой	Сообщение о том, что стек пустой. Для пункта 10 вывод непустого массива ранее освобожденных адресов	Ожидание следующего действия.
10	Пункт меню – 5, 11 Стек ранее не создан	Сообщение о том, что стек еще не создан	Ожидание следующего действия.
10	Пункт меню – 5, 11 Стек создан Выражение: «qwerty»	Сообщение о том, что скобки в выражении сбалансированы	Ожидание следующего действия.
11	Пункт меню – 5, 11 Стек создан Выражение: «qwerty((((»	Сообщение о том, что скобки в выражении не сбалансированы	Ожидание следующего действия.
12	Пункт меню – 5, 11 Стек создан Выражение: [()]{}{[()()]()}	Сообщение о том, что скобки в выражении сбалансированы	Ожидание следующего действия.
13	Пункт меню – 5, 11 Стек создан Количество элементов в выражении превышает размер стека	Сообщение о том, что размеров стека будет недостаточно для анализа введенного выражения	Ожидание следующего действия.
14	Пункт меню – 5, 11 Стек создан Выражение: [()]{}{[()()]()	Сообщение о том, что скобки в выражении не сбалансированы	Ожидание следующего действия.
15	Пункт меню – 7 Стек ранее не был создан	Сообщение об успешном создании стека	Ожидание следующего действия.

16	Пункт меню – 7 Стек ранее был создан	Сообщение о том, что стек был создан ранее	Ожидание следующего действия.
17	Пункт меню – 6, 12 Стек ранее был создан Стек непустой	Сообщение об успешной очистке стека	Ожидание следующего действия.
18	Пункт меню – 6, 12 Стек ранее не был создан	Сообщение о том, что стек еще не был создан	Ожидание следующего действия.
19	Пункт меню – 6, 12 Стек ранее был создан Стек пустой	Сообщение о том, что стек пустой, и очищать нечего	Ожидание следующего действия.
20	Пункт меню – не валидный ввод	Сообщение о том, что введен неверный пункт меню	Завершение программы с ненулевым кодом возврата.

8. Ответы на контрольные вопросы

8.1. Что такое стек?

Стек – последовательный список с переменной длиной, в котором включение и исключение элементов происходит только с одной стороны – с его вершины. Стек работает по принципу LIFO – последним пришел, первым ушел.

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

При реализации стека односвязным списком память выделяется динамически на куче под каждый новый элемент. При этом для каждого элемента помимо самих данных содержится еще и указатель на следующий элемент (4-8 байт).

При реализации стека в виде массива память может выделяться как на стеке, если массив статический, или на куче, если массив динамический. В этом случае для описания стека нужно лишь два элемента – указатель на вершину стека и вместимость стека.

8.3. 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 массива, память удаляется при завершении программы (как при статическом и при динамическом).

8.4. Что происходит с элементами стека при его просмотре?

При просмотре стека его элементы удаляются, поскольку при каждом просмотре нам доступен лишь верхний элемент.

8.5. 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 - ограничен размером стека функции.