**Отчёт лабораторной работы №4**

по курсу «Типы и структуры данных»

Отчёт выполнила:

Лаврова Анастасия

Группа ИУ7-35Б

Вариант 11

**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.

**Задание 4**:

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.

Используя стек, определить, является ли строка, состоящая из скобочек вида (), {}, [], правильной скобочной последовательностью.

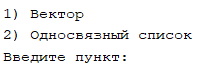
Входные данные:

Данные для стека, которые вводит пользователь.

Пункты меню:

1 – реализация стека в виде массива.

2 – реализация стека в виде списка.



Выходные данные:

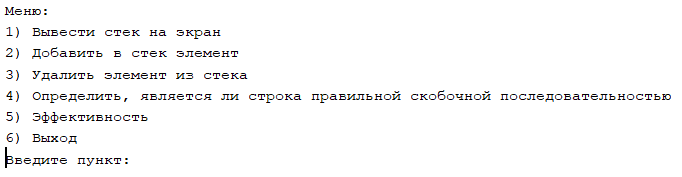
Программа выводит информацию о том, является ли строка правильной скобочной последовательностью, ошибки о переполнении стека, о его пустоте, а также оценку эффективности.

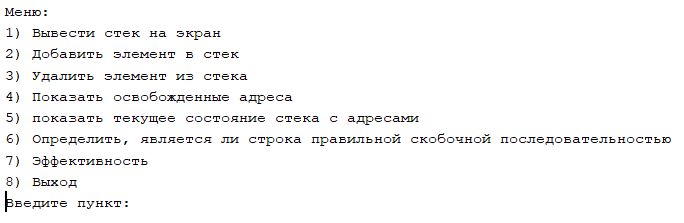
Обращение к программе:

Через консоль

**Описание задачи, реализуемой программой**

1. Показать стек.
2. Добавить элемент в стек.
3. Удалить элемент из стека.
4. Определить, является ли строка правильной скобочной последовательностью.
5. Вывести эффективность.
6. Показать освобожденные адреса (для стека-списка).
7. Показать текущее состояние стека с адресами (для стека-списка).



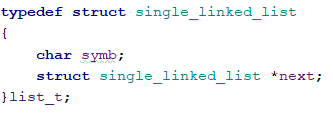


**Внутренние структуры данных:**

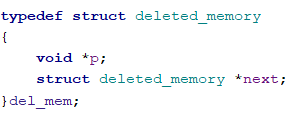
Стек-массив



Стек-список:



Освобожденная память:



**Функции:**

**void** print\_arr(**char** \*string, **int** size);

Ввод: массив символов, размер массива

Вывод: вывод на экран элементов стека

**char** arr\_pop(**char** \*\*string, **int** size);

Ввод: стек-массив, размер стека.

Вывод: стек-масив с удаленным элементом.

**void** arr\_push(**char** \*\*string, **char** symb, **int** size);

Ввод: стек-массив, размер массива, добавляемый элемент

Вывод: стек-массив с добавленным элементом

**void** push(list\_t\*\* list, **char** elem, **int** \*size);

Вход: стек-список, символ char, размер стека.

Выход: стек-список с добавленным элементом.

**char** pop(list\_t\*\* list, del\_mem \*\*list\_del, **int** \*size, **int** \*size\_del);

Вход: стек-список, список памяти, размер стека, количество удаленных элементов.

Выход: стек-список с удаленным элементом

**void** print\_from\_head(**const** list\_t \*list, **int** size);

Вход: стек-список, размер стека

Выход: вывод элементов стека на экран

**void** print\_from\_head\_mem(**const** del\_mem \*list, **int** size);

Вход: стек-список, размер стека

Выход: вывод на экран освобожденные адреса

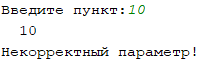
**void** print\_from\_head\_with\_addresses(**const** list\_t \*list, **int** size);

Вход: стек-список, размер стека

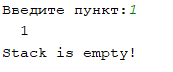
Выход: вывод на экран элементов стека с адресами

**Аварийные ситуации и ошибки пользователя:**

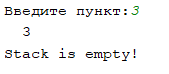
Некорректный параметр при выборе пункта меню



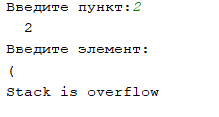
Вывод пустого стека



Удаление элемента из пустого стека.

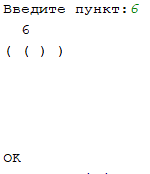


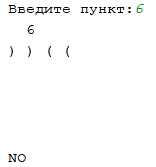
Переполнение стека:



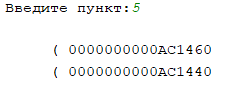
**Тесты:**

Определить, является ли строка правильной скобочной последовательностью.

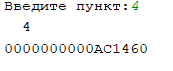




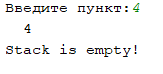
Вывод стека с адресами.



Вывод освобожденной памяти:



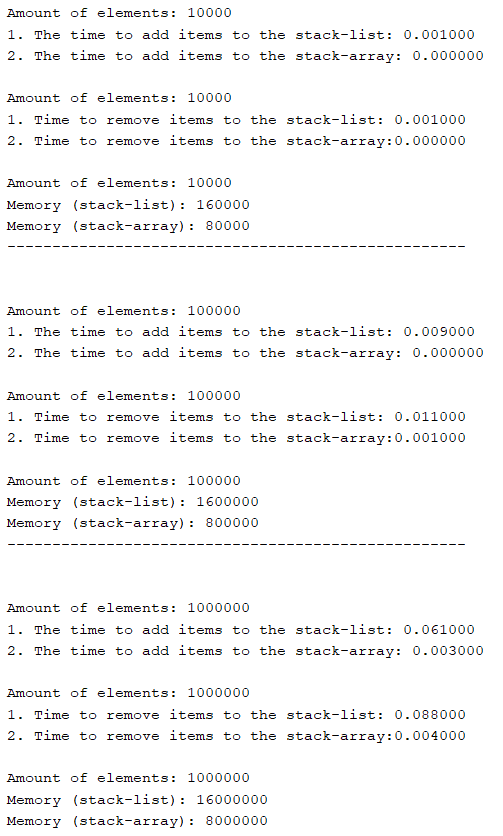
Вывод освобожденной памяти без удаления элементов:



**Оценка эффективности:**

Единица измерения времени – секунда, памяти – байты.

Указано общее время выполнения операций.



**Вывод:**

Реализация стека с помощью массива на 84% выгоднее по времени и в 5 раз выгоднее по памяти, потому что в стеке-списке помимо данных char хранятся указатели. Тем не менее, реализация стека списком удобна, когда мы не знаем количество элементов в стеке.

Существует фрагментация памяти, т.к. память выделяется нелинейно.

**Ответы на вопросы**

1. Что такое стек?

Стек – последовательный список с переменной длиной, в котором включение и исключение элементов происходит только с одной стороны. Функционирует по Last In - First Out (LIFO).

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

При реализации стека списком, память выделяется динамически по мере добавления новых элементов; число элементов в стеке ограничено только количеством доступной ОП.

При реализации стека массивом, выделяется фиксированный участок памяти; в стеке не может быть больше заданного числа элементов. Добавление нового элемента происходит путём смещения индекса вершины.

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

При реализации списком память из-под элемента освобождается после его удаления.

При реализации массивом память из-под элемента не освобождается, происходит лишь изменение значения индекса вершины.

4. Что происходит с элементами стека при его просмотре?

В общем случае доступ есть только к вершине стека; при просмотре она удаляется из стека, а указатель смещается далее. Для отображения состояния стека требуется последовательно проходить по всем его элементам, не «снимая» их.

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

Реализация стека массивом даёт огромный выигрыш во времени, поскольку не нужно каждый раз заново выделять и освобождать память. Тем не менее, в этом случае количество элементов в стеке жёстко ограничено – возможно либо переполнение стека, либо постоянный «излишек» памяти, отведённой под него.

Способ реализации напрямую зависит от условий решаемой задачи – нужно знать примерное число элементов, которые могут храниться в стеке; можно ли пренебрегать переполнением; ограничен ли объём памяти.