**Отчёт лабораторной работы №4**

по курсу «Типы и структуры данных»

Отчёт выполнила:

Кондрашова Ольга

Группа ИУ7-35Б

Вариант 9

**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 6**:

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 палиндромом.

**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 палиндромом, 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 палиндромом.
5. Вывести эффективность.
6. Показать освобожденные адреса (для стека-списка).
7. Показать текущее состояние стека с адресами (для стека-списка).

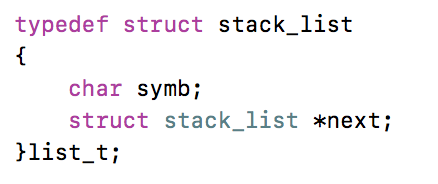
**Алгоритм**

1. Добавление элемента в стек
2. Удаление элемента из стека
3. Печать текущего состояния стека (и адресов удаленных элементов стека-списка).
4. Определение, является ли строка палиндромом.
5. Вывод информации о затраченном времени и затраченной памяти при обработке информации с помощью стека-массива и стека-списка.

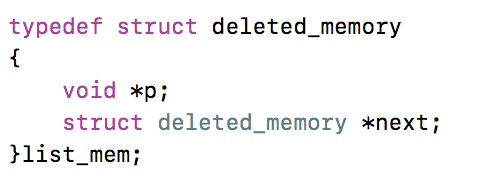
**Внутренние структуры данных**Стек-массив:

MAX\_STACK – максимальный размер стека.  
char \*arr = calloc(MAX\_STACK, sizeof(char));

Стек-список:



Освобожденная память:



**Функции**

void readChar(char \*num);

Вход: введенный элемент char.

Выход: добавление элемента в стек при корректном вводе.

void readInt(int \*num);

Вход: введенный элемент int.

Выход: добавление элемента в стек при корректном вводе.

void arrayPush(char \*\*string, char symb, const int size);

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

Выход: стек-масив с добавленным элементом.

char arrayPop(char \*\*string, int size);

Вход: стек-массив, размер стека.

Выход: стек-масив с удаленным элементом.

void printArray(char \*string, int size);

Вход: стек-массив, размер стека.

Выход: вывод стека на экран.

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

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

Выход: стек-список с добавленным элементом.

char listPop(list\_t\*\* list, list\_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 pushMemory(list\_mem \*\*list, void \*p);

Вход: список памяти, указатель.

Выход: добавление памяти в список.

void printMemory(const list\_mem \*list, int size);

Вход: список памяти, размер.

Выход: вывод адресов на экран.

void printList(const list\_t \*list, int size);

Вход: стек-список, размер стека.

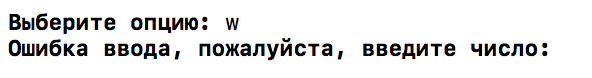
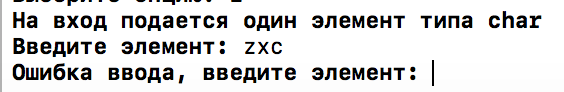
Выход: вывод на экран элементов стека.

void printListWithMemory(const list\_t \*list, int size);

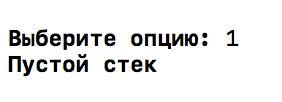
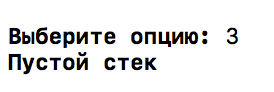
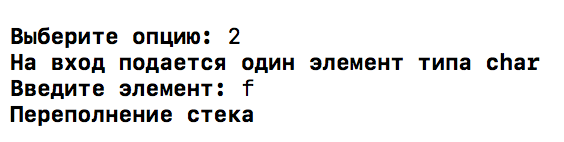
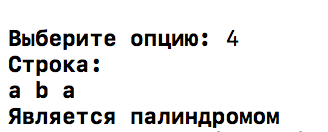
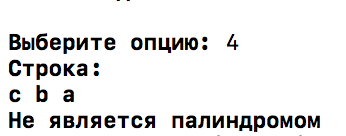
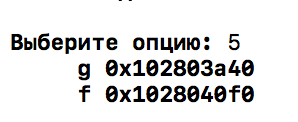
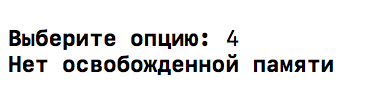
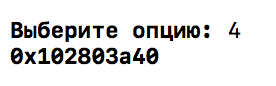
Вход: стек-список, размер стека.

Выход: вывод на экран элементов стека вместе с адресами.

**Аварийные ситуации и ошибки пользователя**

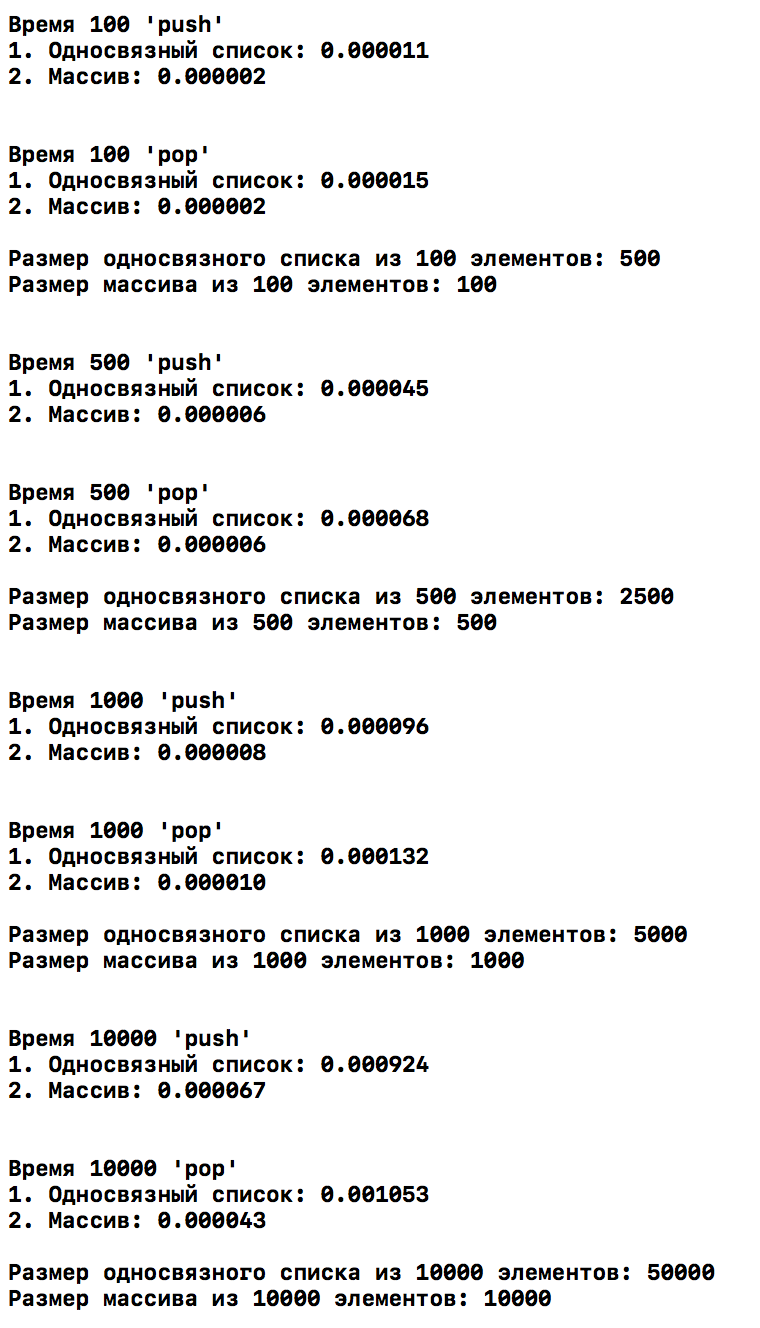
1. Ошибка ввода в меню.  
   
2. Ввод больше одного символа.  
   

**Тесты**

1. Вывод пустого стека.  
   
2. Удаление элемента из пустого стека.  
   
3. Переполнение стека.  
   
4. Определить, является ли палиндромом.  
     
   
5. Вывод стека с адресами.  
   
6. Вывод освобожденной памяти без удаления элементов.  
   
7. Вывод освобожденной памяти после 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 стеком 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.