| Gerb-BMSTU_01 | **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 Щербина Михаил Александрович

Группа ИУ7 – 35Б

Приняла Никульшина Т.А.

# 

[**Описание условия задачи**](#_vqcnleg8e87n) **3**

[**Описание технического задания**](#_1fob9te) **5**

[**Входные данные:**](#_4mag7wyzz530) **5**

[**Выходные данные:**](#_rdpk8jtjmp4b) **5**

[**Функции программы**](#_ory2zfs0hyig) **5**

[**Аварийные Ситуации**](#_ssfuoa738dwd) **6**

[**Описание Структуры Данных**](#_byx5qwtt7ucu) **6**

[**Описание алгоритма**](#_2et92p0) **7**

[**Набор тестов**](#_xodihc7vrfjq) **7**

[Оценка эффективности](#_ukdx7wmlgruz) 9

[**Ответы на контрольные вопросы**](#_298ibzszeico) **10**

[**Вывод**](#_3rdcrjn) **11**

# 

# 

# 

# 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.

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:
  + добавление элемента в стек,
  + удаление элемента из стека,
  + 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 также выдать на экран список адресов освобождаемых элементов при удалении элементов стека.

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.

# Описание технического задания

Используя стек, определить, является ли строка палиндромом

# Входные данные:

1. **Целое число, представляющее собой номер команды**: целое число в диапазоне от 0 до 11.
2. **Командно-зависимые данные:**

**целочисленные значения**

**строки**

# Выходные данные:

1. Результат выполнения определенной команды: ответ (палиндром/не палиндром), вывод стека, вывод адресов
2. Сравнение вариантов реализации стека различными способами.

# Функции программы

1. Добавить элемент в стек-массив
2. Удалить элемент из стек-массива
3. Напечатать стек-массив
4. Напечатать использование памяти стек-массивом
5. Добавить элемент в стек-список
6. Удалить элемент из стек-списка
7. Напечатать стек-список
8. Напечатать использование памяти стек-списком
9. Выполнить техническое задание
10. Напечатать удаленные адреса в массиве-списке
11. Профилировать скорость
12. Профилировать память

# 

# Аварийные Ситуации

1. Введен неверный номер команды
   1. Ввод: Число не являющееся номером команды / строка / пустой ввод
   2. Вывод: Игнорирование ввода
2. Некорректный ввод числа при добавлении в стек
   1. Ввод:Некорректное число
   2. Вывод:Игнор ввода, вывод сообщения об ошибке
3. Попытка взять элемент из пустого стека
   1. Ввод:Пустой стек, команда удаления
   2. Вывод:Игнор, вывод “Error: stack is empty”

# Описание Структуры Данных

**Реализация стека списком**

struct cons\_t;

typedef struct cons\_t cons\_t;

struct cons\_t

{

int value; // значение

struct cons\_t \*next; // след. нода

};

typedef struct

{

cons\_t \*top; // первый элемент

} list\_t;

Я создал отдельный тип для списка и ноды. Это обусловлено тем, что cons\_t не существует как значение, только как указатель.

**Реализация стека динамическим массивом**

typedef struct

{

int \*begin; // Начало массива

int \*end; // Конец массива

int \*ps; // Текущее положение

} stack\_t;

# **Описание алгоритма**

1. Выводится меню 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.

## Описание алгоритма (проверка палиндрома, по заданию)

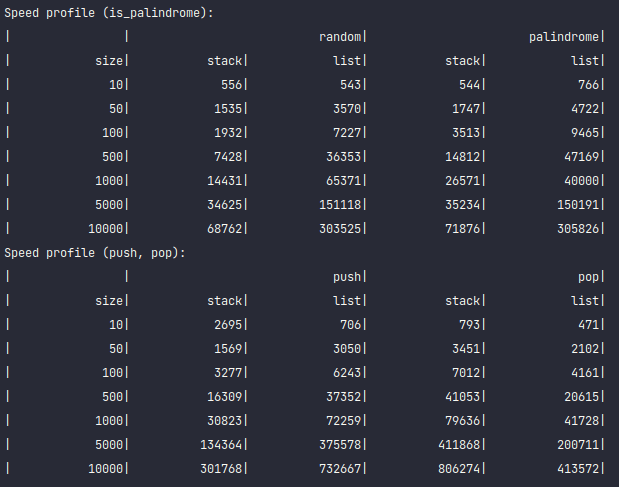
1. Len <- длина строки
2. Mid <- len / 2
3. Stack <- создать стек
4. i <- 0
5. Для i в [0..mid]
   1. ДобавитьВСтек(Stack, string[i])
6. Если Len нечетное
   1. i <- i + 1
7. Flag <- 0
8. Пока i < Len и flag
   1. Если СтекПуст(Stack)
      1. Flag <- 0
   2. Иначе
      1. Elem <- ВзятьИзСтека(Stack)
      2. Если Elem != string[i]
         1. Flag <- 0
   3. i <- i+1
9. Вернуть flag

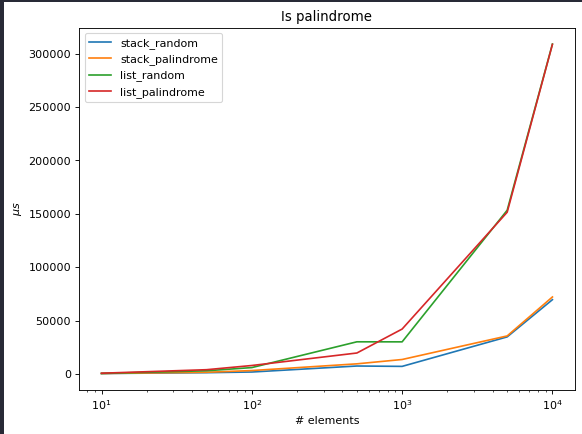
# Набор тестов

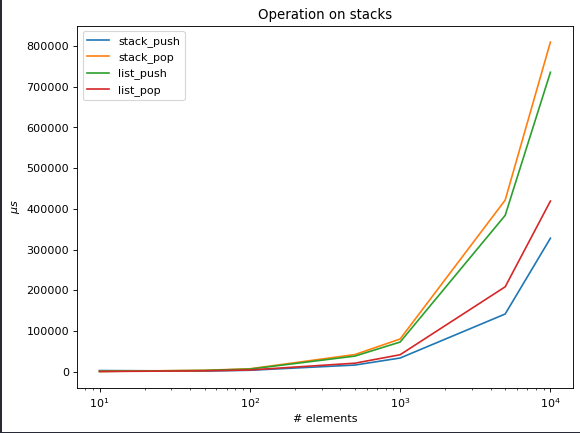
| Номер | Название теста | Ввод | Результат |
| --- | --- | --- | --- |
| 1 | Некорректный ввод команды | 39 | Игнор, вывод меню заново |
| 2 | Пустой ввод | Пустой ввод | Игнор, вывод меню заново |
| 3 | Вывод стека (стек пуст) | Команда, пустой стек | Stack is empty |
| 4 | Удаление из пустого стека | Команда, пустой стек | Вывод cannot delete from empty stack |
| 5 | [П] Добавление элемента | Команда, элемент (введен корректно) | Элемент добавляется |
| 6 | [П] Удаление элемента | Команда, стек не пустой | Элемент удаляется из стека и выводится на экран |
| 7 | [П] Вызов задачи: проверка на полиндром | Команда, Пустая строка | Вывод: String is a palindrome! Grace and rejoice! |
| 8 | [П] Вызов задачи: проверка на полиндром | Команда, Строка из 1 буквы | Вывод: String is a palindrome! Grace and rejoice! |
| 9 | [П] Вызов задачи: проверка на полиндром | Команда, Строка “ab” | Вывод: Not a palindrome! :( |
| 10 | [П] Вывод профилирования памяти | Команда | Вывод профилирования |
| 11 | [П] Вывод профилирования времени | Команда | Вывод профилирования |
| 12 | [П] Выход | Команда | Выход |

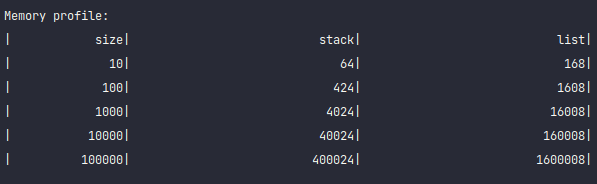
## Оценка эффективности

Измерения эффективности сортировок будут производиться в тактах процессора, с помощью специальной функции будут делаться замеры количества тактов, поэтому погрешность данного измерения минимальна. Частота процессора 3900000000 Гц.









В приведенной таблице массив всегда лучше списка. Но, в случае стека-массива большого размера, когда size ~~ capacity / 2, и произошло перевыделение с бинарным ростом массива, он будет значительно проигрывать в памяти списку по памяти и по времени добавления (по времени только когда происходит перевыделение).

# 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, динамический или статический массив используется). Для каждого элемента стека, который хранится как список, выделяется 8 байт больше, чем для элемента стека, который хранится как массив. Также, списковые стеки расположены нелинейно.

Список: память под элемент каждый раз

Массив: память под элемент**ы** выделяется один раз / по мере необходимости

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

Стек массив:

Верхний элемент удаляется смещением указателя, память не возвращается. Память удаляется в конце использования стека.

При хранении стека как списка: память освобождается при удалении элемента стека. Из моих измерений видно, что происходит фрагментация и выделяется не предыдущий объем.

4. Что происходит с элементами стека при его просмотре?

Зависит от реализации и намерений программиста. Теоретически, элементы стека удаляются, достается верхний, чтобы посмотреть следующий. Практически, этого можно не делать.

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

Стек эффективнее реализовать с помощью массива, так как он выигрывает в количестве занимаемой памяти и во времени обработки стека. Случаи неэффективного использования массива должны быть редки. Удаления динамического массива нет.

# **Вывод**

Стек, реализованный массивом, выигрывает по памяти (в моем случае ~4 раза), но зависит от того значения, от которого “рос” стек. **Размеры сделаны для массива идеальной длины!** Это связанно с тем, что для хранения стека в виде списка требуется память, чтобы хранить указатели.

В реализации массивом требуется меньше времени на обработку: при реализации массивом доступ к нужному элементу быстрее, требуется лишь передвинуть указатель, в то время, если реализовать в виде списка, то требуется время для удаления верхнего элемента (верхушки стека) + еще для перестановки указателя. Но, в случае стека-массива большого размера, когда size ~~ capacity / 2, и произошло перевыделение с бинарным ростом массива, он будет значительно проигрывать в памяти списку по памяти и по времени добавления (по времени только когда происходит перевыделение).

В результате тестирования выявлена небольшая фрагментация памяти.