|  |  |  |
| --- | --- | --- |
|  | |  | | --- | | **Министерство образования и науки 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

По дисциплине «Типы и структуре данных»

### Название Работа со стэком

### Студент Мансуров Владислав Михайлович

*фамилия, имя, отчество*

### Группа ИУ7-36Б

Тип лабораторной работы Учебная

### Название

предприятия

|  |  |  |
| --- | --- | --- |
| Студент |  | Мансуров В.М. |
| Преподаватель | *подпись, дата* | *фамилия, и.о.*  Никульшина Т.А. |
|  | *подпись, дата* | *фамилия, и.о.* |

*2021 г.*

***Содержание***

### Условие задачи 4

### Техническое задание… 4

### Описание алгоритма 5

### Анализ алгоритмов… 8

### Контрольные вопросы… 12

### Заключение… 12

***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*

***Условия задачи***

***Вариант 6***

Создать программу работы со стеком, выполняющую операции добавление, удаления элементов и вывод текущего состояния стека.

Реализовать стек:

а) массивом;

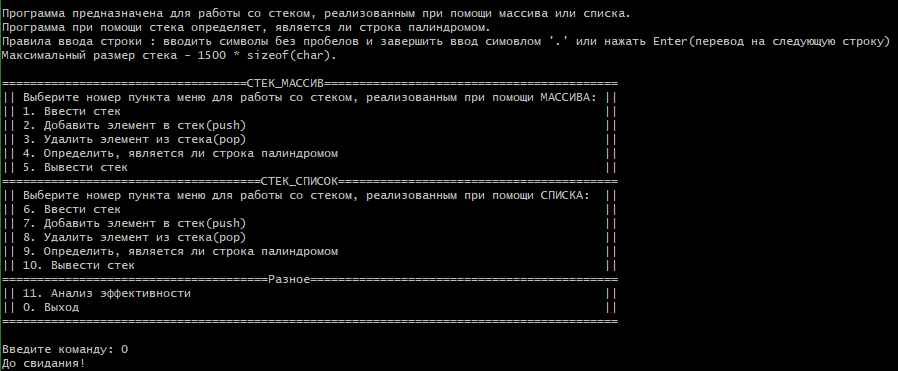
б) списком.

Все стандартные операции со стеком должны быть оформлены подпрограммами. При реализации стека списком в вывод текущего состояния стека добавить просмотр адресов элементов стека и создать свой список или массив свободных областей (адресов освобождаемых элементов) с выводом его на экран.

*Используя стек, определить, является ли строка палиндромом*

***Требование к работе с программой:***

- Взаимодействие с программой строго по меню.



- Водить можно любые цифры за значение меню, в случае не нахождения такого пункта меню выводит сообщение об ошибке и запрашивается ввод заново.

- Чтобы выйти из программы необходимо вести «0».

- При вводе существующих пунктов пользователю предоставляются данные, либо выходит

- Заданы ограничения ввода при работе со стэком, то есть размер 1500

***Входные данные:***

Для взаимодействия с меню вводятся только целые числа, которые указаны в меню.

В некоторых пунктах меню (ввести стек) вводится строка, в пунктах добавить вводится символ, для добавления в стек.

***Выходные данные:***

Вывод данных совершается только в пунктах меню:

* «Вывести стек»

Выводятся адресс элемента и сам элемент

* «Определить, является ли строка палиндромом»

Выводится «Палиндром» или «Не палиндром»

* «Анализ эффективности»

Выводится таблицы с результатами данных об эффективности.

***Описание алгоритма:***

В реализации программы были созданы две структуры для хранения двух версий работы со стеком:

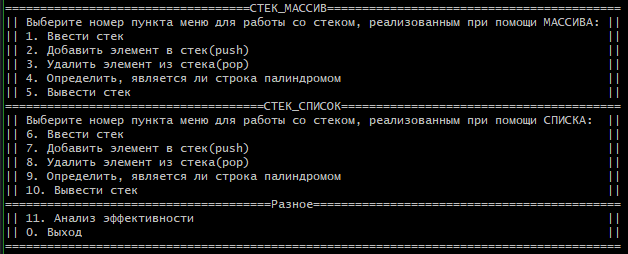
*Стек массив*

typedef struct stack\_array  
{  
 char \*string; //указатель на строку  
 int size; // количество символов  
} stack\_array\_t;

*Стек список*

typedef struct node\_el  
{  
 int size\_stack; //номер символа  
 char symbol; .// символ  
 struct node\_el \*next; // указатель на следующий символ  
} stack\_list\_t;

Для взаимодействия с программой было создано консольное меню:



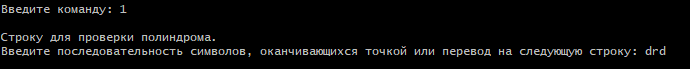
При вводе корректного значения выполняются определенные действия или операции, при некорректном выводится в консоль сообщение об ошибке и предоставляется ввести пункт еще раз. Так происходит пока не будет введен корректное значение меню.

Для реализации программы, была использована динамическая типизация, поэтому были написана функции выделения памяти и очищения памяти, чтобы при работе не возникали утечки памяти.

*Пункта 1 и 6 консольного меню:*

В этих пунктах создается стек массива или списка, заполняются посимвольно.

Где ввод считается до первого вхождения точки или перевода на следующую строку



*Пункт 2 и 7 консольного меню*

Реализованы добавление элемента в стек, если стек до этого не был создан, то он создается и заполняется введенным символом.



*Пункт 3 и 8 консольного меню*

Реализовано удаление элемента из стека, если стек пустой, то выводится сообщение об ошибке.3





*Пункт 4 и 9 консольного меню*

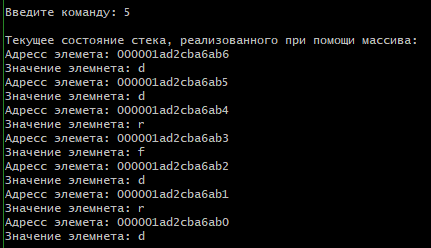
Реализованы для проверки слов палиндромов, где алгоритм сравнение символов следующий:

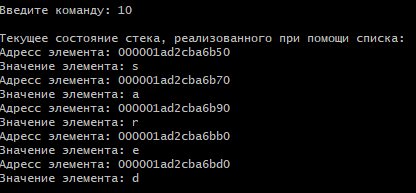
Из стека удаляется половина элементов и записывается в дополнительный стек. Затем извлекается тот элемент, который находится в середине, если размер нечетный. После одновременно извлекаются из каждого стека элементы и сравниваются, затем удаляются. Если все элементы совпали – строка является палиндромом, иначе – не является. Алгоритм также производится с копией стека.

*Пункт 5 и 10 консольного меню*

Осуществляет вывод стека массива или списка, копирую стек и удаляя элемент по его использованию. Реализован вывод освобождённых адресов и элементов.





*Пункт 11 консольного меню*

В пункте реализован анализ эффективности, для замеров нужно ввести количество символов, затем массив и список авто заполняются рандомными символами латиницы. После чего выполняются замеры времени операции добавления, удаления и проверки палиндрома. Также выполняется замер по памяти. А также вывод процентов эффективности работы двух методов.

*Пункт 0 консольного меню* Выход из программы и очищение памяти матрицы и двух векторов-столбцов

***Аварийные выходы из программы***

Выходов аварийных нет, в том случае, если программа не будет принудительна закрыта.

Выход осуществляется только пунктом меню «0»

В ошибочных случаях выводится сообщение об ошибке и ввод повторяется, пока программа не получит корректные данные.

Если в результате операции – умножения или вводе матрицы и векторов, память не была выделена, то пользователь возвращается в главное меню и выводится сообщение об не выделении память в определенной операции.

***Анализ эффективности методов***

Для замеров эффективности хранение и времени операции работы со стеком были взяты несколько размеров строк:

* Строка размером 100 символов
* Строка размером 500 символов
* Строка размером 1300 символов

Хранение стека массива - **size\_stack \* sizeof(char) + sizeof(int)**.

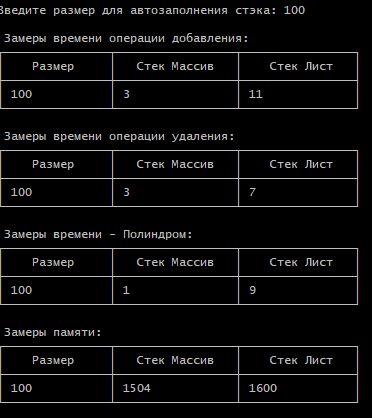
Хранение стека списка - **size \* sizeof(stack\_list\_t)**.

где:

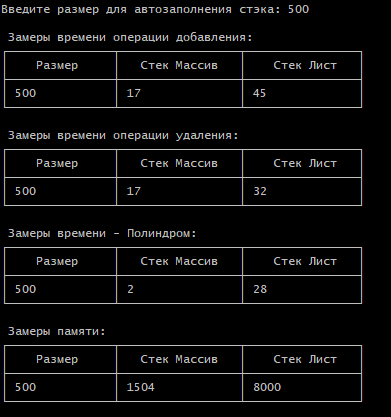
* **size\_stack**  - маскимальный размер стека
* **size** – количество символов
* **sizeof(char)** – размер символова в байтах
* **sizeof(int)** – размер целого числа в байтах
* **sizeof(stack\_list\_t)** – размер структуры для стека списка

В таблицы выводятся замеры времени и пямять.

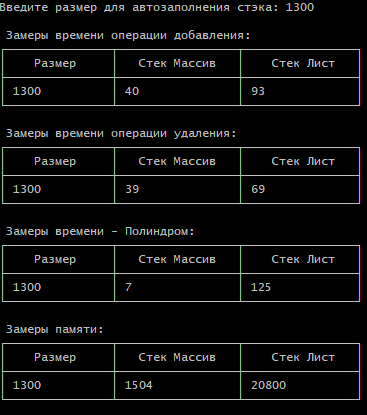
* ***Замеры строки на 100 символов***



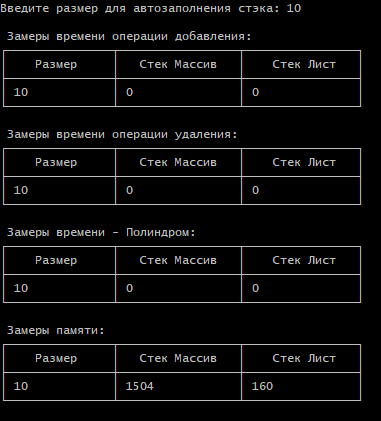
* ***Замеры строки на 500 символов***



* ***Замеры строки на 1300 символов***



* ***Замеры строки на 10 символов***



В результате измерений, были получены выше предоставленные данные, по которым можно сделать вывод, что каждый из 2-х методов эффективный в разных ситуациях, так:

* Использование стека как массива, более эффективнее, по времени всегда во всех проделанных операциях примерно:
  + Добавление – 17-50%
  + Удаление – 42-56%
  + Проверка на палиндром – 80-90 %
* Использование стека как список, не всегда эффективный, но с ним быстрее работа, когда надо вставить элемент, это более легче, чем в массив, передвигая все элементы назад, так сказать, работая с указателями на следующий элемент.
* По памяти хранения эффективность при малых размерах, например – 10 символов, то эффективнее xранить стек в виде списка, так при работе как с массивом, выделяется сразу же память максимального размера стека. Но при больших размерах, эффективнее хранить стек, как массив.

***Контрольные вопросы***

*Что такое стек?*

Стек – это структура данных - последовательный список с переменной длиной, в котором включение и исключение элементов происходит только с одной стороны – с его вершины.

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

При реализации стека при помощи связанного списка: **(sizeof(type) + sizeof(\*type\_t))\*count**,

где **count** — число элементов, **type** — тип элементов, *type\_t* — тип узла.

При реализации стека при помощи массива: **sizeof(type)\*count**,

где **count** — число элементов, **type** — тип элементов.

При реализации стека, как массив память выделяется сразу же, то есть один буфером, а при стека как лист память выделяется под каждый элемент, в зависимости от введенного количество элементов.

*Каким образом освобождается память при удалении элемента стека при различной реализации стека?*

При удалении элементов в стеке-массива очищение массива происходит только при выходе из программы.

При удалении элементов в стеке-списка очищение каждый элемент.

*Что происходит с элементами стека при его просмотре?*

Элементы стека извлекаются из стека — уничтожаются.

*Каким образом эффективнее реализовывать стек? От чего это зависит?*

Реализовывать стек при помощи списка эффективнее в том, что память для него выделяется в куче и ограничена размером оперативной памяти, в то время как для массива память ограничена размером стека. По времени работы реализация стека при помощи массива эффективнее.

***Вывод***

В результате выполнения данной лабораторной работы, было использованы два алгоритма хранения и операции работы со стеком, при этом использование двух способа работы со стеком:

* Cтек, как массив показал большую эффективность по времени работы различных операции, то есть такой стек использовать лучше когда нам необходим, чтобы программа работала более быстрее.
* Стек, как список показал эффективность только в том случае, если количество элементов не больше 100, в таком случае список хранить лучше, то есть такой стек использовать лучше, тогда когда необходима эффективность по памяти хранения, но при этом время программы будет работать менее эффективнее, чем стек-массив.