|  |  |
| --- | --- |
|  | **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 №3 по предмету

“Типы и структуры данных”

Работа со стеком.

Выполнила:

Кузнецова Анастасия

Группа:

ИУ7-31Б

Вариант 4

Проверил(а):

Силантьева Александра Васильевна

**Цель работы:** реализовать операции работы со стеком, который представлен в виде массива (статического или динамического) и в виде односвязного линейного списка; оценить преимущества и недостатки каждой реализации: получить представление о механизмах выделения и освобождения памяти при работе со стеком.

**Описание условия задачи**

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.

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

**2. Описание ТЗ:**

Описание исходных данных и результатов

*Входные данные:*

Программа представляет собой меню, пользователь должен ввести свой выбор (целое число от 0 до 9)

Если пользователь хочет добавить элемент к стеку-массиву или к стеку-списку, то входными данными является один символ.

Если же пользователь выбрал проверку скобок в стеке-массиве или стеке-списке, то на вход подаётся последовательность символов.

Ограничения для входных данных:

* При добавлении элемента должен быть введён только 1 символ
* Количество элементов стека-массива не должно превышать MAX\_STACK

*Выходные данные:*

Программа выводит меню и в зависимости от выбора пользователя выводит:

1. Вывод результата проверки правильности расстановки скобок в стеке-массиве
2. Вывод результата проверки правильности расстановки скобок в стеке-списке
3. Вывод стека-массива
4. Вывод стека-списка и адресов удалённых элементов
5. Вывод результатов сравнения времени и памяти при двух реализациях

Описание задачи, реализуемой программой

Программа выводит меню и выполняет задачу, запрошенную пользователем. В программе реализовано меню с выборами действий:

1. Выход
2. Добавление элемента в стек-массив
3. Добавление элемента в стек-список
4. Удаление элемента из стека-массива
5. Удаление элемента из стека-списка
6. Проверка правильности расстановки скобок в стеке-массиве
7. Проверка правильности расстановки скобок в стеке-списке
8. Вывод стека-массива
9. Вывод стека-списка
10. Сравнения времени и памяти при двух реализациях

Способ обращения к программе

Программу следует вызывать из командной строки с помощью команды:

./app.exe

Описание возможных аварийных ситуаций и ошибок пользователей

* При добавлении элемента к стеку-массиву введён не один элемент, а несколько
* При добавлении элемента к стеку-списку введён не один элемент, а несколько
* Ввод элемента, при котором превышается количество элементов MAX\_STACK 100 в стеке-массиве
* Удаление элемента из пустого стека-массива
* Удаление элемента из пустого стека-списка

**3. Описание внутренних СД**

Стек-массив:

#define MAX\_STACK 100  
  
typedef struct stack

{  
 char array[MAX\_STACK];  
 int top;  
} Stack;

Стек-список:

struct node

{  
 char value;  
 struct node \*next;  
};  
  
typedef struct node Node;

**4. Описание алгоритма**

1. Вывод меню

2. В зависимости от выбора пользователя:

2.0 Выход

2.1.1 Проверка переполнения стека-массива

2.1.2 При успешной проверке добавление элемента в стек-массив

2.2.1 Проверка переполнения стека-списка

2.2.2 При успешной проверке добавление элемента в стек-список

2.3.1 Проверка пустоты стека-массива

2.3.2 Удаление элемента из стека-массива

2.4.1 Проверка пустоты стека-списка

2.4.2 Удаление элемента из стека-списка

2.5 Проверка правильности расстановки скобок в стеке-массиве

2.6 Проверка правильности расстановки скобок в стеке-списке

2.7 Вывод стека-массива

2.8.1 Вывод стека-списка

2.8.2 Вывод адресов удалённых элементов из стека-списка

2.9 Вывод результатов сравнения времени и памяти при двух реализациях

**5. Набор тестов**

Позитивные тесты:

|  |  |  |  |
| --- | --- | --- | --- |
| № | Описание | Input | Output |
| 1 | Добавление элементов в стек-массив и вывод состояния стека | Enter a choice:  1  Input the symbol:  a  Enter a choice:  1  Input the symbol:  b  Enter a choice:  1  Input the symbol:  c  Enter a choice:  7 | Elements:  c b a |
| 2 | Добавление элементов в стек-список и вывод состояния стека | Enter a choice:  3  Input the symbol:  a  Enter a choice:  3  Input the symbol:  b  Enter a choice:  3  Input the symbol:  c  Enter a choice:  8 | Elements:  c b a  Addresses:  00000051b9f21330 00000051b9f21310 00000051b9f212f0 |
| 3 | Добавление и удаление элементов в стек-массив и вывод состояния стека | Enter a choice:  1  Input the symbol:  a  Enter a choice:  1  Input the symbol:  b  Enter a choice:  1  Input the symbol:  c  Enter a choice:  2  Enter a choice:  7 | Elements:  b a |
| 4 | Добавление и удаление элементов в стек-список и вывод состояния стека | Enter a choice:  3  Input the symbol:  a  Enter a choice:  3  Input the symbol:  b  Enter a choice:  3  Input the symbol:  c  Enter a choice:  4  Enter a choice:  8 | Elements:  b a  Addresses:  0000007bb0741310 0000007bb07412f0 |
| 5 | Вывод состояния пустого стека-массива | Enter a choice:  7 | Stack is empty. |
| 6 | Вывод состояния пустого стека-списка | Enter a choice:  8 | Stack is empty. |
| 7 | Проверка правильности расстановки скобок с помощью стека-массива (верные скобки) | Enter a choice:  5  Enter string:  ([][][]({})) | Correct brackets |
| 8 | Проверка правильности расстановки скобок с помощью стека-массива (неверные скобки) | Enter a choice:  6  Enter string:  ([)a{}] | Incorrect brackets |
| 9 | Проверка правильности расстановки скобок с помощью стека-списка (верные скобки) | Enter a choice:  6  Enter string:  ({a}) | Correct brackets |
| 10 | Проверка правильности расстановки скобок с помощью стека-списка (неверные скобки) | Enter a choice:  6  Enter string:  ([)a{}] | Incorrect brackets |
| 11 | Проверка правильности расстановки скобок с помощью стека-массива (нет скобок) | Enter a choice:  5  Enter string:  a | No brackets. |
| 12 | Проверка правильности расстановки скобок с помощью стека-списка (нет скобок) | Enter a choice:  6  Enter string:  a | No brackets. |

Негативные тесты:

При добавлении элемента к стеку-массиву введён не один элемент, а несколько

При добавлении элемента к стеку-списку введён не один элемент, а несколько

Ввод элемента, при котором превышается количество элементов MAX\_STACK 100 в стеке-массиве

Удаление элемента из пустого стека-массива

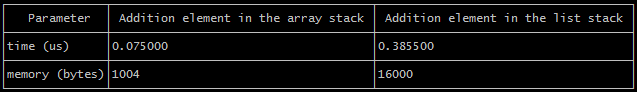
Удаление элемента из пустого стека-списка

|  |  |  |  |
| --- | --- | --- | --- |
| № | Описание | Input | Output |
| 1 | При добавлении элемента к стеку-массиву введено несколько символов | Enter a choice:  1  Input the symbol:  ajvn | Input error. |
| 2 | При добавлении элемента к стеку-списку введено несколько символов | Enter a choice:  3  Input the symbol:  akjfdb | Input error. |
| 3 | Удаление элемента из пустого стека-массива | Enter a choice:  2 | Empty stack. |
| 4 | Удаление элемента из пустого стека-списка | Enter a choice:  4 | Empty stack. |

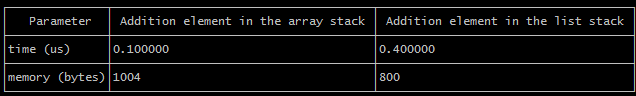
**6. Сравнение времени и памяти при двух реализациях.**

Я сравнивала время добавления и удаления элемента в стек-массив и в стек-список. Замеры проводила на 10000 повторах брала среднее значение времени. Результаты представлены ниже:

При полностью заполненном массиве (1000 элементов):



При не полностью заполненном массиве (50 элементов):



Массив оказывается эффективнее и по времени (в 4-5 раз), и по памяти в случае полного заполнения (примерно в 16 раз). Эффективность списка (на 20%) по памяти можно заметить только если элементов будет меньше максимального количества элементов в статическом массиве (в моём случае я взяла 50 из 1000), так как в статическом массиве память выделяется под весь размер, а в списке только под введённое количество элементов.

**6. Выводы по проделанной работе**

В результате выполненной лабораторной работы я приобрела навыки работы со стеком, который представлен в виде статического массива и в виде односвязного линейного списка, оценила преимущества и недостатки каждой реализации, получила представление о механизмах выделения и освобождения памяти при работе со стеком.

Преимущества массива, недостатки списка: при полностью заполненном массиве он выигрывает по времени в 4-5 раз и по памяти в 16 раз, список проигрывает.

Недостатки массива, преимущества списка: при не полностью заполненном массиве он проигрывает по памяти на 20% (проигрыш зависит от количества элементов), список выигрывает.

**7. Ответы на 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: последним пришел – первым ушел, Last In – First Out (LIFO).

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

Если стек реализован в виде статического или динамического массива (вектора), то для его хранения обычно отводится непрерывная область памяти ограниченного размера, имеющая нижнюю и верхнюю границу. В моей реализации это размер структуры, хранящей индекс последнего добавленного элемента и массив значений char, количество которых равно MAX\_STACK 100.

Если стек реализован в виде статического односвязного линейного списка, то для его хранения отводится указатель на структуру, содержащую указатель на такую же структуру и само значение элемента char. При каждом добавлении элемента выделяется новая область памяти, адрес которой записывается в указатель стека.

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

При реализации стека с помощью статического массива память выделяется при компиляции и не меняется во время работы программы.

При реализации стека с помощью линейного односвязного списка при удалении элемента сначала по указателю стека считывается информация об исключаемом элементе, а затем указатель смещается к предыдущему элементу. После чего освобождается память, выделенная под элемент.

4. Что происходит с элементами стека при его просмотре?

При просмотре стека чтобы обратиться ко всем элементам стека надо удалить предыдущий элемент, так как принцип стека LIFO мы не можем обратиться к элементу стека из середины.

Таким образом при просмотре стека все его элементы поочередно удаляются.

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

При полностью заполненном массиве эффективнее реализовывать стек с помощью массива.

Эффективность зависит от того, насколько заполнен массив. Если размер массива намного превышает количество введённых элементов стека, то эффективнее по памяти использовать односвязный список.

Однако классическое представление стека - односвязный линейный список.