|  |  |
| --- | --- |
| 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 ???

Группа ИУ7 – 3?Б

Преподаватель Барышникова М.Ю.

*2021 г.*

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 так же 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**

Элементами стека являются слова. Распечатайте слова в обратном порядке в перевернутом виде.

Входные данные:

1. Номер команды: целое число в диапазоне от 0 до 7;
2. Элементы стека: слова.

Выходные данные:

1. Результат выполнения определенной команды.
2. Сравнение времени работы стеков.

Функции программы:

1. Добавить элемент в стек.
2. Распечатать слова в прямом порядке.
3. Распечатать слова в обратном порядке.
4. Удалить элемент из стека.
5. Вывести массив свободных областей.
6. Очистить стек.
7. Сравнить время работы.
8. Выход из программы.

Обращение к программе:

Запускается через терминал.

Аварийные ситуации:

1. Некорректный ввод номера команды.

На входе: число, большее чем 7 или меньшее, чем 0.

На выходе: сообщение «Invalid menu code»

2. Некорректный ввод размера стека.

На входе: число, которое больше размера стека.

На выходе: сообщение «Input error»

3. Удаление элемента из пустого стека.

На входе: 4

На выходе: сообщение «Sorry, but stack is empty.»

4. Добавление элемента в стек, когда он заполнен полностью.

На ввод: элемент стека.

На выход: сообщение «Sorry, but stac overflow.».

**Описание структуры данных**

Размер стека задается макросом:

*#define COUNT\_WORD 1000*

Длина слова:

*#define LEN\_WORD 15*

**Реализация стека с помощью линейного односвязного списка.**

*typedef struct stack\_list\_t stack\_list\_t;*

*struct stack\_list\_t*

*{*

*char name [LEN\_WORD + 1];*

*stack\_list\_t \*next;*

*};*

Поля структуры:

**name***–* слово в стеке

***\**next** *–* указатель на следующий элемент списка

**Реализация стека с помощью массива.**

*char stack\_arr [COUNT\_WORD] [LEN\_WORD + 1];*

*int arr\_len = -1;*

**stack\_arr** – стек в виде массива;

**arr\_len** – длина массива.

**Описание алгоритма**

Выводится меню.

1. Пользователь вводит номер команды.
2. Программа выполняет указанное действие.

**Набор тестов**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Название теста** | **Пользовательский ввод** | **Результат** |
| 1 | Некорректный ввод команды | a | Invalid menu code |
| 2 | Некорректный ввод команды | 12 | Invalid menu code |
| 3 | Добавление элемента в полный стек | Ввод команды на добавление элемента. | Sorry, but stack overflow. |
| 4 | Удаление из пустого стека | Ввод команды на удаление элемента. | Sorry, but stack is empty. |
| 5 | Добавление элемента в стек | Ввод команды на добавление элемента. | Add is done! |
| 6 | Удаление из стека | Ввод команды на удаление элемента. | Del is done! |
| 7 | Вывод слов в прямом порядке | Команда на вывод слов в прямом порядке | Список слов. |
| 8 | Вывод слов в обратном порядке | Команда на вывод слов в обратном порядке | Список слов в обратном порядке. |
| 9 | Очищение стека | Команда для очищения стека. | Clear is done. |
| 10 | Вывод пустых областей | Команда для вывода пустых областей. | Адреса свободной памяти. |

**Оценка эффективности по времени (в тактах)**

Измерения эффективности производиться в тактах процессора.

*Добавления элемента в стек.*

|  |  |  |
| --- | --- | --- |
| **Размер** | **Список** | **Массив** |
| 100 | 116580 | 6800 |
| 500 | 1136660 | 26360 |
| 1000 | 2413860 | 50920 |

*Удаления элемента из стека.*

|  |  |  |
| --- | --- | --- |
| **Размер** | **Список** | **Массив** |
| 100 | 21360 | 1200 |
| 500 | 37700 | 1220 |
| 1000 | 52720 | 1250 |

**Оценка эффективности по памяти**

|  |  |  |
| --- | --- | --- |
| **Размер** | **Список** | **Массив** |
| 10 | 240 | 16000 |
| 100 | 2400 | 16000 |
| 500 | 12000 | 16000 |
| 670 | 16080 | 16000 |
| 1000 | 24000 | 16000 |

**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, динамически или статический массив используется). Для каждого элемента стека, который хранится как список, выделяется на 4 или 8 байт (если брать современные ПК) больше, чем для элемента стека, который хранится как массив.   
Данные байты использованы для хранения указателя на следующий элемент списка.

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

Если хранить стек как список, то верхний элемент удаляется при помощи операции освобождения памяти для него и смещением указателя, который указывает на начало стека.  
При хранении стека как массив, память удаляется при завершении программы (как при статическом и при динамическом).

*4. Что происходит с элементами стека при его просмотре?*

Элементы стека удаляются, так как каждый раз достается верхний элемент стека, чтобы посмотреть следующий.

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

Стек эффективнее реализовать с помощью массива, так как обработка такого стека намного быстрее, а при полном заполнении стека, стек-массив выигрывает и по памяти.

**Вывод**

При реализации стека, односвязный список выигрывают по памяти у статического массива при заполнении до 65%. Однако, время обработки односвязного списка намного больше времени обработки массива при любом проценте заполнения. Следовательно, при заполнении стека до 65% необходимо выбирать, что важнее: время или память. При большем проценте заполнения всегда выгоднее использовать массив.

Так же, при реализации программы, было установлено, что фрагментации памяти при работе с односвязным списком не происходит.