|  |  |
| --- | --- |
| **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 Егорова Полина Александровна

Группа ИУ7 – 34Б

Преподаватель Силантьева Александра Васильевна

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: o 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**

Ввести целые числа в 2 стека. Используя третий стек отсортировать все введенные данные.

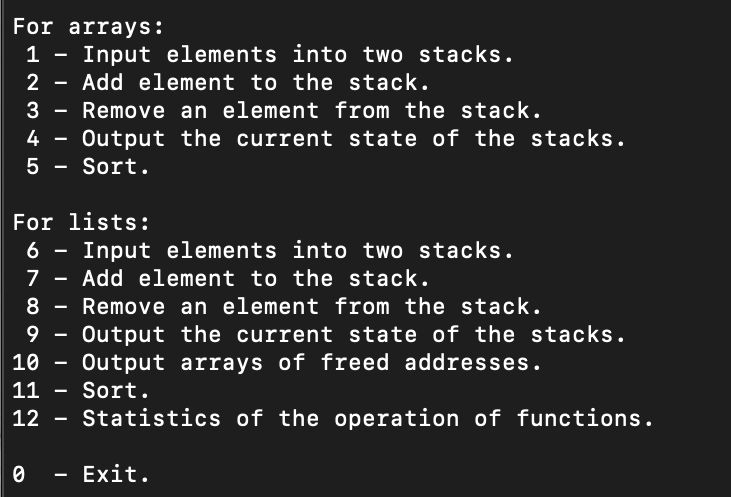
Входные данные:

1. Целое число, представляющее собой номер команды: целое число в диапазоне от 0 до 12 включительно.
2. Командно-зависимые данные: максимальное количество элемента стека, количество элементов первого/второго стека, элементы первого/второго стека, команда – выбор стека, над которым будет производится операция: число в диапазоне от 1 до 2.

Выходные данные:

1. Результат выполнения выбранной команды.
2. Характеристика сравнения вариантов дополнения стека элементом, удаления элемента из стека и сортировки стеков, при реализации стека в виде односвязного списка и в виде массива.

Функции программы:



Обращение к программе:

Запуск через терминал (./app.exe)

Аварийные ситуации:

* 1. Некорректный ввод номера команды.

На вход: число, большее чем 12 или меньшее, чем 0.

На выход: *«Incorrect number.»*

* 1. Некорректный ввод номера команды.

На вход: строковый литерал.

На выход: *«Incorrect input.»*

* 1. Некорректный ввод максимального размера стека.

На вход: ноль, отрицательное число, строковый литерал.

На выход: *«Incorrect input.»*

* 1. Некорректный ввод размера стека.

На вход: отрицательное число, нуль, число, превышающее максимально допустимое число для количества элементов стека, строковый литерал.

На выход: *«Incorrect input.»*

* 1. Совершение команды над несозданным стеком.

На вход: команда 2, 3, 4, 5, 7, 8, 9, 11.

На выход: *«Stack is NULL.»*

* 1. Создание нового стека при уже наличии введенного в программу стека (для одного типа создания стека).

На вход: команда 1 или 6.

На выход: *«Stack already exist.»*

* 1. Добавление элемента в заполненный стек.

На вход: команда 2 или 7.

На выход: *«Stack is full.»*

* 1. Добавление элемента в заполненный стек.

На вход: команда 3 или 8.

На выход: *«Stack is empty.»*

* 1. Вывод пустого стека.

На вход: команда 4 или 9.

На выход: *«Stack is empty.»*

# **Описание структуры данных**

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

// стек в виде линейного односвязного списка  
typedef struct list\_stack list\_stack\_r;  
struct list\_stack  
{  
 int data; // элемент стека  
 int ind; // индекс узла списка  
 list\_stack\_r \*next; // указатель на следующий элемент  
};

**Реализация стека с помощью массива:**

// стек в виде массива  
typedef struct  
{  
 int top; // голова стека  
 int cap; // количество элементов стека  
 int \*arr; // указатель на массив элементов стека  
} array\_stack\_r;

**Реализация массива освобожденных адресов:**

// массив освобожденных адресов  
typedef struct  
{  
 size\_t \*arr; // указатель на массив  
 int cap; // количество элементов  
 int ind; // индекс текущего элемента списка  
} addresses\_r;

# **Описание алгоритма**

1. Вывод меню программы.
2. Ввод номера команды из предложенного меню.
3. Пока не будет введен 0 (команда выхода из программы), выполнение операций в соответствии с набранной командой.

# **Набор тестов**

# **Стек реализован в виде массива (и общие тесты):**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Название теста** | **Пользовательский ввод** | **Результат** |
| 1 | Некорректный ввод команды | 100  abc | Incorrect number. |
| 2 | Пустой ввод | Пустой ввод. | Вывод меню. |
| 3 | Создание стека (команда 1). Неверный ввод максимального размера стека. | 0  -1  Abc  & | Incorrect number. |
| 4 | Создание стека (команда 1). Неверный ввод размера одного из двух стеков. | 0  -1  Abc  &  Число, более, чем максимальный размер стека | Incorrect number. |
| 5 | Создание стека (команда 1). Неверный ввод элемента стека. | A  & | Incorrect element of stack. |
| 6 | Добавление элемента в стек (команда 2). Неверный выбор стека. | 4  A  & | You have to enter 1 or 2. |
| 7 | Добавление элемента в стек (команда 2). | A  & | Incorrect element of stack. |
| 8 | Добавление элемента в переполненный стек (команда 2). |  | Stack is full. |
| 9 | Удаление элемента из стека (команда 3). Удаление из пустого стека. |  | Stack is empty. |
| 10 | Не существует стека (при вводе команд 2-5 и 7-11). |  | Stack is NULL. |
|  |  |  |  |
| 11 | Создание стека (команда 1). | Ввод верных размеров, элементов. | Стек успешно создан. |
| 12 | Добавление элемента в стек (команда 2). | Введен элемент. | Элемент добавлен в стек. |
| 13 | Удаление элемента из стека (команда 3). | Вызов команды. | Элемент удален и выведен на экран. |
| 14 | Вывод стека на экран (команда 4). | Вызов команды. | Стек выведен на экран. |
| 15 | Сортировка стеков (команда 5). | Первый стек: 4 6 3 9 Второй стек: 0 1 -4 8 | -4 0 1 3 4 6 8 9 |

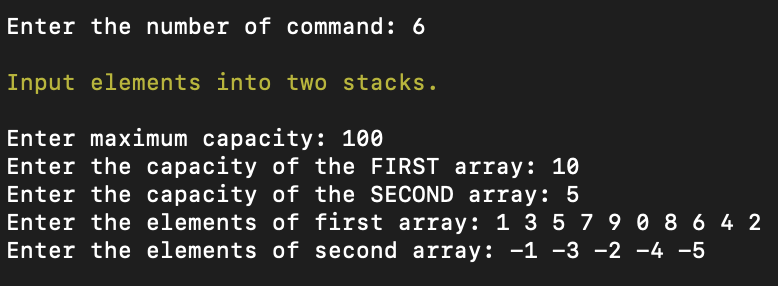
# **Стек реализован в виде списка (и общие тесты):**

|  |  |  |  |
| --- | --- | --- | --- |
| 16 | Создание стека (команда 6). Неверный ввод максимального размера стека. | 0  -1  Abc  & | Incorrect number. |
| 17 | Создание стека (команда 6). Неверный ввод размера одного из двух стеков. | 0  -1  Abc  &  Число, более, чем максимальный размер стека | Incorrect number. |
| 18 | Создание стека (команда 6). Неверный ввод элемента стека. | A  & | Incorrect element of stack. |
| 19 | Добавление элемента в стек (команда 7). Неверный выбор стека. | 4  A  & | You have to enter 1 or 2. |
| 20 | Добавление элемента в стек (команда 7). | A  & | Incorrect element of stack. |
| 21 | Добавление элемента в переполненный стек (команда 7). |  | Stack is full. |
| 22 | Удаление элемента из стека (команда 8). Удаление из пустого стека. |  | Stack is empty. |
| 23 | Сортировка стеков (команда 11). Один из стеков пустой. | Первый стек:  Второй стек: 9 3 5 | One of the stacks is empty. |
| 24 | Создание стека (команда 6). | Ввод верных размеров, элементов. | Стек успешно создан. |
| 25 | Добавление элемента в стек (команда 7). | Введен элемент. | Элемент добавлен в стек. |
| 26 | Удаление элемента из стека (команда 8). | Вызов команды. | Элемент удален и выведен на экран. |
| 27 | Вывод стека на экран (команда 9). | Вызов команды. | Стек выведен на экран. |
| 28 | Вывод массива свободных адресов на экран (команда 10). | Вызов команды. | Массив выведен на экран. |
| 29 | Сортировка стеков (команда 11). | Первый стек: 4 6 3 9 Второй стек: 0 1 -4 8 | -4 0 1 3 4 6 8 9 |

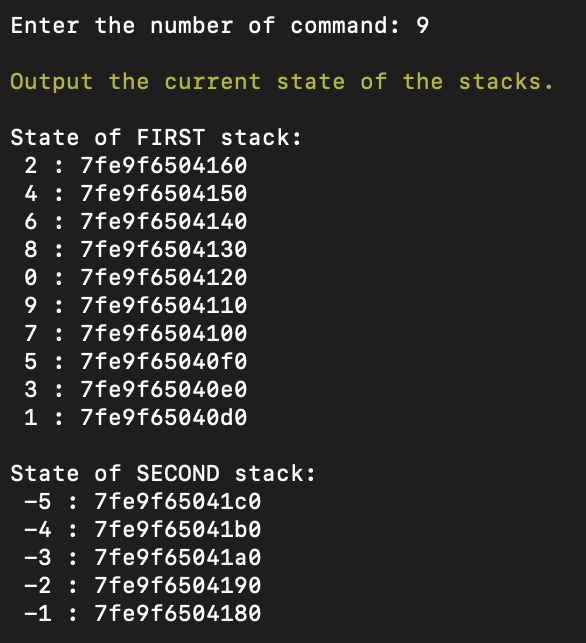
# 

# **Пример вывод массива освобожденных адресов**

* 1. Ввод элементов стеков (в виде линейных односвязных списков).



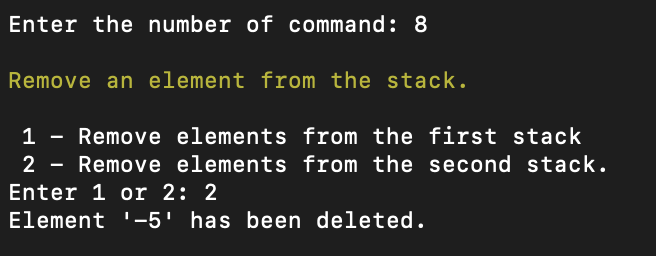
* 1. Вывод текущего состояния стеков.



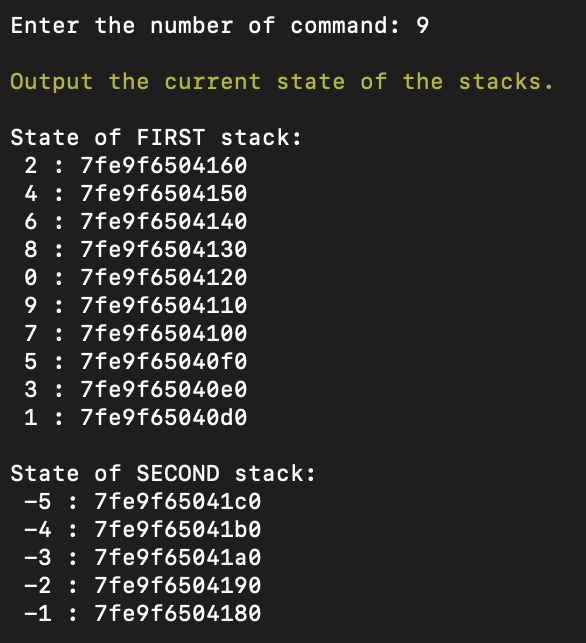
* 1. Удаление двух элементов из первого стека и одного элемента из второго стека.



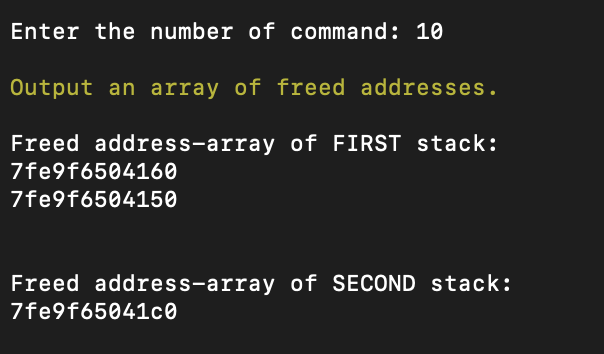




* 1. Вывод текущего состояния стеков.



* 1. Вывод массива освобожденных адресов.



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

Команда добавления элемента в стек.

|  |  |  |
| --- | --- | --- |
| **Размер** | **Список** | **Массив** |
| 10 | 3 | 1 |
| 100 | 3 | 1 |
| 500 | 3 | 1 |
| 1000 | 3 | 1 |

Команда удаления элемента из стека.

|  |  |  |
| --- | --- | --- |
| **Размер** | **Список** | **Массив** |
| 10 | 3 | 1 |
| 100 | 3 | 1 |
| 500 | 3 | 1 |
| 1000 | 3 | 1 |

Команда сортировки двух стеков при помощи третьего.

|  |  |  |
| --- | --- | --- |
| **Размер** | **Список** | **Массив** |
| 10 | 27 | 6 |
| 100 | 1830 | 26 |
| 500 | 23330 | 81 |
| 1000 | 93605 | 134 |

# **Оценка эффективности по памяти (в байтах)**

**Добавление элемента:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Макс. размер** | **Размер** | **Список** | **Массив** |
| 10 | 10 | 176 | 48 |
| 100 | 100 | 1616 | 408 |
| 500 | 500 | 8016 | 2008 |
| 1000 | 1000 | 16016 | 4008 |

**Удаление элемента:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Макс. размер** | **Размер** | **Список** | **Массив** |
| 10 | 10 | 160 | 48 |
| 100 | 100 | 1600 | 408 |
| 500 | 500 | 8000 | 2008 |
| 1000 | 1000 | 13600 | 4008 |

**Сортировка (размер результирующего стека):**

|  |  |  |  |
| --- | --- | --- | --- |
| **Макс. размер** | **Размер** | **Список** | **Массив** |
| 10 | 10 | 320 | 88 |
| 100 | 100 | 3200 | 808 |
| 500 | 500 | 16000 | 4008 |
| 1000 | 1000 | 32000 | 8008 |

**Расчет памяти:**

Стек в виде массива: (указатель на int (8 байт)) + максимальное кол-во элементов \* int (4 байта).

Стек в виде списка: (указатель на следующий элемент (8 байт) + (int \* 2, 8 байт)) \* кол-во элементов

Можно сделать вывод, что стек в виде списка проигрывает и по времени, и по памяти стеку, реализованному в виде массива. Но при неполном заполнении стека, например, не на 100%, а на 25%, то стек в виде списка будет выигрывать по памяти у стека в виде массива.

**Для примера - операция добавления элемента в стек:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Макс. размер** | **Размер** | **Список** | **Массив** |
| 100 | 10 | 176 | 408 |
| 100 | 20 | 336 | 408 |
| 100 | 26 | 432 | 408 |
| 100 | 30 | 496 | 408 |
| 100 | 40 | 656 | 408 |

# 

# **Ответы на контрольные вопросы**

*1. Что такое стек?*

Стек – это последовательный список с переменной длиной, в котором включение и исключение элементов происходит только с его вершины. Стек работает по принципу LIFO: последним пришел – первым ушел.

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

При включении элемента в стек в виде списка сначала происходит выделение области памяти, адрес которой записывается в указатель стека, а затем по значению этого указателя в стек помещается информация. 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

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 очищается при завершении программы.

*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 он выигрывает в количестве занимаемой памяти (роль играет процент заполнения) и во времени обработки стека.

Хранение с помощью списка может выигрывать, если только стек реализован с помощью статического массива, так как в данном случае размер памяти под список ограничен размером оперативной памяти (хранится в куче), а для статического массива — ограничен размером стека функции.

# 

# **Вывод**

Возможны две реализации стека: в виде списка и в виде массива.

Если стек представлен в виде массива, он будет выигрывать по памяти. Это связанно с тем, что в данной реализации необходимо выделить память для указателя на массив целых чисел и для максимально возможного количества хранящихся в нем чисел. При для хранения стека в виде списка требуется память, чтобы хранить указатели на последующие элемент и сами целочисленные элементы. Но если заполненность не превышает 25%, то стек в виде списка будет выигрывать (см. таблицы).

Так же реализация в виде массива требуется меньше времени на обработку: в случае операции сортировки стеков, время увеличивается в разы (при количестве элементов, равном 10, выигрыш в 2 раза, при 1000 элементах – в 700 раз). Это связанно с тем, что при реализации массивом доступ к нужному элементу получить проще, требуется лишь передвинуть указатель, в то время, если реализовать в виде списка, то требуется время для удаления верхнего элемента (верхушки стека), а также для перестановки указателя.

Вывод таков: для хранения стека самым оптимальным решением будет использование массива (если процент заполнения от 25 до 100), — это выгоднее и по памяти, и по времени. Если же стек заполнен менее, чем на 25%, затраты по памяти уменьшатся при использовании списка.