|  |  |
| --- | --- |
|  | **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)** |

Отчет по лабораторной работе № 4

«Работа со стеком»

Работу выполнил:

студент группы ИУ7-32Б

Богатырев И.С.

Работу проверил:

Барышникова М.Ю.

Никульшина Т.А.

Москва, 2020 г.

**Условие задачи**

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.

**Техническое задание**

Создать программу работы со стеком, выполняющую операции добавления, удаления элементов и вывод текущего состояния стека, выполнить особое задание:

Элементами стека являются адреса памяти.

При реализации массивами - их вводить, при реализации списком – брать адрес выделенной памяти под элемент.

Сравнить реализации стека на динамическом массиве и односвязном списке.

Если стек реализован на односвязном списке, то проверить, происходит ли фрагментация памяти.

**Входные данные:**

Числа, вводимые в стек, представленный в виде массива, либо номера команд, выбор реализаций.

**Выходные данные:**

Элементы стека, список адресов, либо время выполнения, затрачиваемое на разное количество разных операций, также области освобожденной памяти в виде односвязного списка, и занимаемая память.

**Функция программы:**

Реализация работы со стеком. Демонстрация распределения памяти при работе со стеком. Анализ времени и объема памяти, которое требуется на добавление и удаление элементов из стека при реализации динамическим массивом и списком. Реализация специального задания.

**Аварийные ситуации:**

Попытка добавить элемент в заполненный стек. Программа выводит “STACK OVERFLOW”

Попытка убрать элемент из пустого стека. Программа выводит “STACK UNDERFLOW”

Попытка вывести элементы пустого стека Программа выводит “NULL STACK”

**Структуры данных**

**Реализация стека в виде динамического массива**

*typedef struct stack\_array*

*{*

*size\_t size;*

*size\_t top;*

*char \*\*data;*

*} stack\_array\_t;*

*size* - размер выделенной памяти

*top* - индекс последнего элемента

\*\**data* - ссылка на данные

**Реализация стека в виде односвязного списка**

*typedef struct node*

*{*

*struct node \*this;*

*struct node \*next;*

*} stack\_list\_t;*

*\*this* - указатель на текущую ноду

*\*next* - указатель на следующую ноду

**Алгоритмы**

**Односвязный список**

**Функция добавления нового элемента:**

*int push\_list(stack\_list\_t \*\*head)*

*{*

*stack\_list\_t \*tmp = malloc(sizeof(stack\_list\_t));*

*if (tmp == NULL)*

*{*

*return STACK\_OVERFLOW;*

*}*

*tmp->next = \*head;*

*tmp->this = tmp;*

*\*head = tmp;*

*return 0;*

*}*

**Функция удаления верхнего элемента:**

stack\_list\_t \*pop\_list(stack\_list\_t \*\*head)

{

stack\_list\_t \*out;

if ((\*head) == NULL)

{

exit(STACK\_UNDERFLOW);

}

out = \*head;

\*head = (\*head)->next;

return out;

}

**Динамический массив**

**Функция добавления нового элемента:**

*int push\_arr(stack\_array\_t \*\*stack, char value[100])*

*{*

*if ((\*stack)->top >= (\*stack)->size)*

*{*

*if (resize\_arr(stack, sizeof(int)) != 0)*

*{*

*return MEMORY\_ALLOCATION\_ERROR;*

*}*

*}*

*(\*stack)->data[(\*stack)->top] = strdup(value);*

*(\*stack)->top++;*

*return 0;*

*}*

**Функция удаления верхнего элемента**

*int pop\_arr(stack\_array\_t \*stack, char \*\*element)*

*{*

*if (stack->top == 0)*

*{*

*return (STACK\_UNDERFLOW);*

*}*

*stack->top--;*

*char \*tmp = stack->data[stack->top];*

*\*element = tmp;*

*return 0;*

*}*

**Тесты**

| **SIZE** | **ARRAY STACK TICKS** | **LIST STACK TICKS** | **ARRAY STACK EFFICIENCY** | **ARRAY STACK MEMORY** | **LIST STACK MEMORY** | **ARRAY MEMORY EFFICIENCY** |
| --- | --- | --- | --- | --- | --- | --- |
| 10 | 183588 | 77292 | -137% | 96 | 160 | 40% |
| 40 | 136078 | 133007 | -2% | 336 | 640 | 48% |
| 41 | 114729 | 142603 | 20% | 656 | 656 | 0% |
| 50 | 138717 | 152341 | 9% | 656 | 800 | 18% |
| 60 | 199532 | 258300 | 23% | 656 | 960 | 32% |
| 70 | 229063 | 216957 | -5% | 656 | 1120 | 42% |
| 79 | 353913 | 254664 | -38% | 656 | 1264 | 49% |
| 81 | 371686 | 442839 | 17% | 1296 | 1296 | 0% |
| 100 | 537420 | 598612 | 11% | 1296 | 1600 | 19% |
| 500 | 1322108 | 1651920 | 20% | 5136 | 8000 | 36% |
| 1000 | 5281786 | 4288275 | -23% | 10256 | 16000 | 36% |
| 5000 | 11922094 | 20535489 | 42% | 40976 | 80000 | 49% |
| 10000 | 26564814 | 35535024 | 26% | 81936 | 160000 | 49% |
| 100000 | 270013088 | 348933313 | 23% | 1310736 | 1600000 | 19% |

**Вывод**

Можно сделать вывод, что динамический массив не дает существенного выигрыша ни по быстродействию, ни по памяти: по быстродействию максимальный выигрыш равен 42% при 5000 элементах и заполненности массива на ~97%, но также очень много времени тратится при работе с пограничным числом элементов, когда приходится увеличивать величину массива - тогда массив начинает проигрывать реализации в виде односвязного списка по быстродействию (проигрыш составляет 20-30%).

По памяти массив не проигрывает никогда, но не всегда имеет существенное преимущество: сразу после перераспределения памяти в динамическом массиве обе структуры равны по занимаемой памяти, максимальный выигрыш по памяти составляет 49% (при полной заполненности массива).

Также в ходе лабораторной было выявлено, что при использовании односвязного списка не происходит фрагментации памяти (элементы стека располагаются в памяти друг за другом).

В итоге можно сказать, что динамический массив и односвязный список по эффективности различаются не сильно.

**Ответы на вопросы**

**Что такое стек?**

Стек – последовательный список с переменной длиной, в котором включение и исключение элементов происходит только с одной стороны. Функционирует по Last In First Out (LIFO).

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

При реализации стека списком, память выделяется динамически по мере добавления новых элементов; число элементов в стеке ограничено только количеством доступной ОП.

При реализации стека массивом, выделяется фиксированный участок памяти; в стеке не может быть больше заданного числа элементов. Добавление нового элемента происходит путём смещения индекса вершины.

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

При реализации списком память из-под элемента освобождается после его удаления.

При реализации массивом память из-под элемента не освобождается, происходит лишь изменение значения индекса вершины.

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

В общем случае доступ есть только к вершине стека; при просмотре она удаляется из стека, а указатель смещается далее. Для отображения состояния стека требуется последовательно проходить по всем его элементам, не «снимая» их.

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

Реализация стека массивом даёт огромный выигрыш во времени, поскольку не нужно каждый раз заново выделять и освобождать память. Тем не менее, в этом случае количество элементов в стеке приходится контролировать вручную – определять, когда размер стека подходит к концу и расширять массив функцией realloc.