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 ЛАБОРОТОРНОЙ РАБОТЕ № 4
«РАБОТА СО СТЕКОМ»

Студент Байрамгалин Ярослав Ринатович
Группа ИУ7-33Б
Преподаватель Силантьева Александра Васильевна

Москва 2021
МГТУ им. Н.Э. Баумана

1. Описание условия задачи

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

При разработке программы работы со стеком реализовать операции добавления и удаления элементов из стека и отображения текущего состояния стека.

При разработке программы реализовать стек:

- массивом;
- 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.

При разработке интерфейса программы следует предусмотреть:

- вывод на экран операции, производимой программой,
- указание формата и диапазона вводимых данных,
- указание формата выводимых данных,
- наличие пояснений при выводе результатов.

2. Описание технического задания

Входные данные:

Тип стека: ввести 1 – для работы со стеком, реализованным через связный список, 2 – для работы со стеком, реализованным через массив.

Пункт меню: целое число, указывающее на необходимый пункт меню (1 – удаление элемента из списка 1; 2 – удаление элемента из списка 2; 3 – добавление элемента в список 1; 4 – добавление элемента в список 2; 5 – просмотр списка 1; 6 – просмотр списка 2; 0 – для выхода).

При выборе пункта меню 1 или 2 ввести целые число: от -2147483648 до 214748364.

Выходные данные:

При выборе пункта меню 1: значения отсортированного стека, каждое из которых указывается вместе с адресом ячейки памяти, в которой находится элемент стека. После указываются адреса памяти, которые были освобождены во время выполнения операций со всеми стеками, во время работы программы.

При выборе пункта меню 2: значения отсортированного стека.

Действие программы:

Сортировка двух стеков с целыми числами.

Обращение к программе:

Запускается командой $./app.exe$ через терминал, находясь в директории, содержащей программу.

Аварийные ситуации:

1. Переполнение стека.
2. Недостаток оперативной памяти.
3. Пустой ввод пункта меню.
4. Пустой ввод значения элемента стека.

3. Описание структуры данных

Для хранения стека в виде списка используется структура данных `stack`, определенная как `my_stack_t`.

```
struct stack {
    int value;
    struct stack *prev;
};
typedef struct stack my_stack_t;
```

Поля структуры:

- `value` – значение элемента стека,
- `*prev` – указатель на предыдущий элемент стека.

Для хранения стека в виде массива используется структура данных `arr_stack`, определенная как `arr_stack_t`.

```
struct arr_stack {
    int values[MAX_ARR_STACK_SZ];
    int count_in_stack;
};
typedef struct stack my_stack_t;
```

Поля структуры:

- `values` – массив элементов стека,
- `count_in_stack` – текущее количество элементов в стеке.

4. Описание алгоритма

1. Пользователь выбирает тип стека (через связный список или через массив).
2. Пользователь вводит первый стек.
3. Пользователь вводит второй стек.
4. На экран выводится 1 стек, состоящий из двух отсортированных с необходимой дополнительной информацией.

5. Набор тестов

	Описание теста	**Ввод**	**Ожидаемый вывод**
1	Обычный тест, стек через список	Пункт меню: 1 Первый стек: 3 2 1 Второй стек: 4 5 6 7	value = 1 at 0x…… value = 2 at 0x…… value = 3 at 0x…… value = 4 at 0x…… value = 5 at 0x…… value = 6 at 0x…… value = 7 at 0x…… Freed chunks of memory: from 0x…… to 0x…… …….
2	Обычный тест, стек через массив	Пункт меню: 2 Первый стек: 3 2 1 Второй стек: 4 5 6 7	value = 1 value = 2 value = 3 value = 4 value = 5 value = 6 value = 7
3	Пустой пункт меню	Пункт меню: (пусто)	Input error
4	Неверный пункт меню	Пункт меню: 3	Input error
5	Переполнение стека через список	Пункт меню: 1 Первый стек: 1 2	Stack overflow Error occurred!

		3 … 99 100 101	
6	Переполнение стека через список (суммарное количество элементов в стеках превышает допустимое значение)	Пункт меню: 1 Первый стек: 1 2 … 98 Второй стек: 1 2 3	Stack overflow Error occurred!
7	Переполнение стека через массив	Пункт меню: 1 Первый стек: 1 2 … 99 100 101	Stack overflow Error occurred!
8	Неверный ввод элемента стека	Пункт меню: 1 или 2 Первый стек: 1 value = 2	Input error (Продолжение ввода)

6. Оценка эффективности

Кол-во эл.	Список		Массив	
	память, bytes	время, ticks	память, bytes	время, ticks
10	192	19	8008	31
50	832	152	8008	108
100	1632	1349	8008	324
250	4032	7015	8008	966
500	8064	15433	8008	2318

Примечание: для оценки времени при использовании списка была удалена строка, запоминающая адреса.

7. Выводы

1. Как реализация стека статическим массивом, так и списком имеет свои преимущества и недостатки.
2. Преимущества статического массива для реализации стека: скорость работы.
3. Преимущества связного списка. Элементы могут располагаться в разных частях оперативной памяти, проще реализовать динамическое выделение памяти. Не нужен сложный аллокатор. Не требует большое количество дополнительной памяти (лишь sizeof(struct stack*) * n для указателя на след элемент).
4. Недостатки статического массива: из-за фиксированного размера для сортировки требуется большой объем дополнительной памяти (для помещения в такой же стек), также даже при маленьком числе элементов будет выделяться максимальное количество необходимой памяти.
5. Недостатки связного списка: относительно медленно работает, так как для каждого добавления нового элемента требуется вызов malloc.
6. При достаточно большом объеме входных данных скорость сортировки одинаковым способом стека через список примерно в 7 раз медленнее сортировки стека через массив. Предположительно, такое увеличение во времени связано с тем, что для каждой

операции push вызывается malloc, который увеличивает время работы программы.

7. Использование стека через статический список может быть затруднено.
8. Для реализации сортировки имея доступ только к последнему элементу может потребоваться O(n) дополнительной памяти.

8. Ответы на контрольные вопросы

Что такое стек?

Абстрактный тип данных, работающий по принципу LIFO (last in first out), реализующий операции добавления элемента в конец, удаления элемента с конца, просмотра значения последнего элемента.

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

Для стека через список: не менее чем sizeof(struct stack *) + sizeof(typeof(value)) для каждого элемента.

Для стека через массив: не менее чем sizeof(typeof(value)) для каждого элемента.

Что происходит с элементами стека при его просмотре?

Стек обычно реализуется таким образом, чтобы для просмотра (n-1)-го элемента необходимо было удалить n-ый элемент. Для реализации операций над стеком (сортировка, вставка в середину, …) элементы «снятые» сверху можно помещать в другой стек, затем возвращая их в исходный.

Каким образом эффективнее реализовывать стек? От чего это зависит?

Нельзя ответить однозначно. Как реализация через связный список, так и через массив имеет свои преимущества и недостатки.

Для связного списка

Преимущества: нет необходимости выделять 1 большой последовательный раздел оперативной памяти для хранения элементов. То есть, присутствует возможность хранить разные элементы стека в разных частях оперативной памяти.

Недостатки: несколько больший размер требуемой памяти.

Для массива

Преимущества: возможность использовать статическую память.

Недостатки: требуется один большой последовательный раздел оперативной памяти.