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)

Дисциплина “Типы и структуры данных”

Лабораторный практикум №4

по теме: «работа со стеком»

Выполнил студент: __*Клименко Алексей Константинович*_____

фамилия, имя, отчество

Группа: _____*ИУ7-35Б*_____

Проверил, к.п.н.: _____

подпись, дата

Оценка _____ Дата _____

2020 г.

цель работы

реализовать операции работы со стеком, который представлен в виде массива (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.

техническое задание

исходные данные

Исходными данными являются числа, запрашиваемые у пользователя в программе, и помещаемые в стеки.

Примеры вставки и удаления чисел из двух стеков:

Ввод пользователя	**Стек a**	**Стек b**
push a 1 2 3	(1,2,3)	()
pop a	(1,2)	()
push 2 5	(1,2)	(5)
push B 3	(1,2)	(5,3)

результат

Результатом работы программы являются три стэка, два из которых содержат введёные пользователем числа, а третий является отсортированным объединением чисел из двух других стеков.

описание задачи, реализуемой программой

Программа решает задачу слияния с одновременной сортировкой элементов из двух стеков в третий. По мере работы программы можно запросить у неё вывод текущего состояния на экран командой **show**.

способы обращения к программе

Для запуска программы необходимо запустить файл **app.exe**.

возможные аварийные ситуации и ошибки пользователя

При неверном вводе команды программа попросит ввести команду снова, а при невозможности выполнения команды (например, извлечение из пустого стека) сообщает об этом, и предлагает ввести другую команду.

Другой случай - переполнение стека при вставке или при объединении стеков. В этом случае обработка стека (стеков) прекращается, и выводится сообщение о переполнении стека.

Также при попытке извлечения из пустого стека будет выведено сообщение об этом.

Структуры данных

Для реализации стека на массиве была выбрана следующая структура:

```
struct stack_arr
{
   uint32_t size;       // размер стека
   uint32_t __capacity;  // вместимость стека
   int32_t *__data;     // массив данных стека
   int32_t *top;       // указатель на вершину стека
};
```

Реализация стека на связном списке:

```
struct __st_lst_node
{
   int32_t data;              // данные узла списка
   struct __st_lst_node *prev;  // указатель на предыдущий узел
};

struct stack_lst
{
   uint32_t size;            // размер стека
   struct __st_lst_node *top;  // указатель на вершину стека
};
```

Набор функций

```
// Создаёт новый стек с заданной реализацией.
struct stack st_create(__stack_imp_type type);

// Удаляет стек, освобождая память.
void st_destroy(struct stack *st);

// Классические операции вставки и удаления из вершины.
int st_push(struct stack *st, int32_t value);
int st_pop(struct stack *st, int32_t *value);

// Возвращает размер стека - кол-во хранящихся элементов.
```

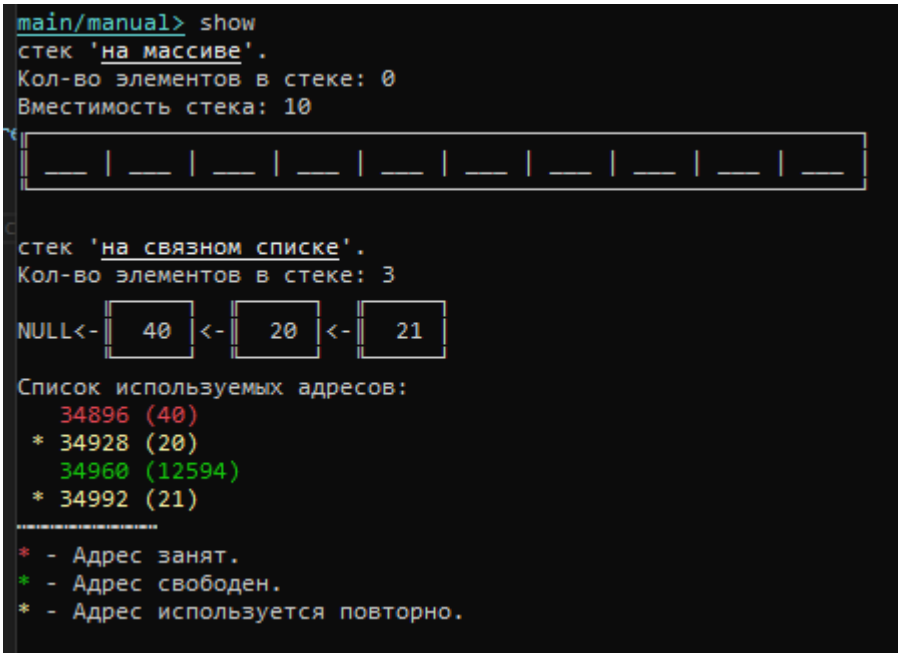
```
uint32_t st_get_size(const struct stack *st);

// Объединяет два стека в один с упорядочиванием последнего.
int st_merge(struct stack *st_out, struct stack *st_a, struct stack *st_b);
```

Описание алгоритма обработки данных

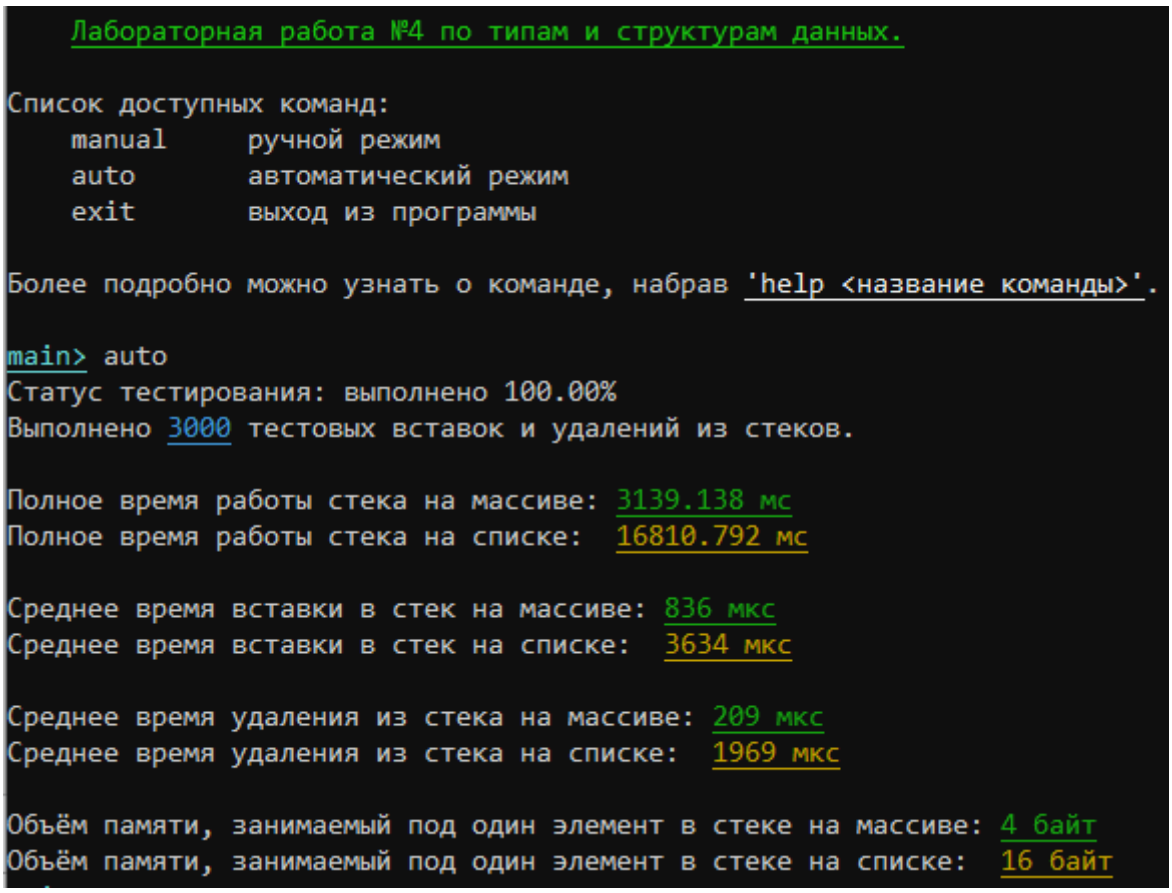
Объединение двух стеков в один состоит из последовательного добавления в конечный стек элементов из обоих начальных:

пока есть элементы в начальных стеках:
- **выбрать** начальный стек с наибольшим элементом в вершине (s_1).
- **извлечь** элемент (e) с вершины выбранного стека (s_1).
- **пока** элемент в вершине конечного стека больше (e):
  - **переложить** вершину конечного стека в (s_2)
- **вставить** элемент (e) в вершину конечного стека.
- **пока** элемент в вершине конечного стека неменьше (e):
  - **переложить** вершину стека (s_2) в конечный стек.

Пример вывода стеков на экран. (Во втором стеке лежат числа 40, 20 и 21). Также выводятся и используемые адреса для хранения узлов списка.



Результаты автоматизированного тестирования времени вставки и удаления элементов из стеков:



Как видно, эффективность реализации на массиве составляет в среднем **82%**.

Для операции вставки эффективность по времени составляет **78%**, а для удаления **91%**.

Для детального расчета эффективности по памяти учтём заполненность обоих стеков:

максимальный размер стеков	процент заполнености	Объем занимаемый массивом	Объем занимаемый списком	Эффективность списка
10	10%	40 байт	16 байт	60%
10	20%	40 байт	32 байт	20%
10	30%	40 байт	48 байт	-20%
1000	20%	4000 байт	3200 байт	20%
1000	**25%**	**4000 байт**	**4000 байт**	**0%**
1000	30%	4000 байт	4800 байт	-20%

Из таблицы видно, что **при размере стека меньше чем 25%** от максимального, реализация на списке будет эффективнее по памяти. Поэтому такая реализация будет выгоднее в случаях, когда действительный размер стека зачастую не превосходит четверти от максимального.

набор функциональных тестов

№	Описание теста	Входные данные	Выходные данные
1	Некорректная команда	`a`	Сообщение о неверной команде. Возврат в меню
2	Показ пустых стеков	`manual A L A` `show`	Вывод информации о стеках на экран

3	Вставка элементов в стек	`manual A L A`
		`push A 10 20 30`
		`show A`
			Вывод стека (10, 20, 30) на экран

4	Удаление элементов из стека	`manual A L A`
		`push A 10 20`
		`pop A`
		`pop A`
		`show`
			Вывод пустого стека

5	Удаление элементов из пустого стека	`manual A L A` `pop A`	Вывод сообщения о том, что стек пуст
6	Переполнение стека	`manual A L A` `push A 1 2 ... 15 16 17`	Вывод сообщения о том, что числа 16 и 17 не были добавлены в стек.
7	Вызов автоматического тестирования	`auto`	Вывод параметров работы стека при разных реализациях

Выводы по проделанной работе

Благодаря тестированию работы стека на массиве и на списке, мне удалось выявить достоинства и недостатки обоих реализаций:

Достоинства реализации на массиве:

+ быстрота вставки и извлечения.

+ эффективность по памяти при заполенности более 25%

Достоинства реализации на списке:

+ не занимает лишнего места при небольшой заполненности (менее 25% от теоретически возможной)

+ эффективен по памяти при больших колебаниях размера стека (то 3 элемента в стеке, то 300)

Контрольные вопросы

1. *Что такое стек?*

Стек - это абстрактная структура данных, имеющая операции вставки и удаления элементов стека. Работает по принципу LIFO - последний пришёл - первый вышел. (работаем только с вершиной стека)

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

При реализации стека на статическом массиве память выделяется на стеке (машинном) единожды и никак не меняет свой размер во время работы программы.

При реализации стека на динамическом массиве память выделяется в куче единожды в момент инициализации и впоследующем при нехватке свободного места в стеке.

При реализации стека на связном списке память выделяется каждый раз при добавлении нового элемента в стек.

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

При реализации стека на статическом массиве память очищается при выходе из области видимости стека, или проще говоря при завершении работы со стеком.

При реализации стека на динамическом массиве память очищается при необходимости сжать стэк (уменьшить его вместимость до действительного размера), но этого можно не делать при частых и сбалансированных вставках и удалениях.

При реализации стека на связном списке память очищается каждый раз при удалении элемента из стека.

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

В классической реализации стека просмотр (как элементарное действие над стеком) невозможен. Для просмотра стека необходимо использовать дополнительную структуру данных (не обязательно другой стек), переместив в неё последовательно все элементы из вершины стека. Тогда, складывая элементы обратно в стек, мы будем наблюдать их порядок при добавлении.

5. *Каким образом эффективнее реализовывать стек?*

Эффективнее реализовывать стек на массиве, когда точно известно максимальное число элементов в нём, а размер стека зачастую будет превышать четверть от максимального. Но если не представляется возможным вычислить это число заранее или зачастую в стеке будет храниться не так много элементов (менее 25%), то эффективнее будет использовать стек на связном списке.