Министерство образования и науки 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

По дисциплине «Типы и структуре данных»

Название Работа со стеком

Студент Мамврийский Иван Сергеевич
фамилия, имя, отчество

Группа ИУ7-36Б

Студент _____ Мамврийский И.С.
подпись, дата *фамилия, и.о.*

Преподаватель _____ Никульшина Т.А.
подпись, дата *фамилия, и.о.*

2022г.

Условие 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.

Ввести арифметическое выражение типа: число|знак| ... число|знак| число. Вычислить значение выражения.

Требование к работе с программой

-

```
==========================Меню==========================
|| 1. Ввести стек, реализованного массивом.            ||
|| 2. Добавить элемент в стек.                         ||
|| 3. Удалить элемент из стека.                        ||
|| 4. Вывести стек                                     ||
|| 5. Ввести стек, реализованного связанным списком.   ||
|| 6. Добавить элемент в стек.                         ||
|| 7. Удалить элемент из стека.                        ||
|| 8. Вывести стек.                                    ||
|| 9. Вывод адресса удаленных элементов                ||
|| 10. Арифметическое выражение                        ||
|| 11. Вывод замеров и эффективности                   ||
|| 0. Выход                                            ||
================================================================
```

Взаимодействие с программой строго по меню.

- Водить можно любые цифры за значение меню, в случае не нахождения такого пункта меню выводит сообщение об ошибке и запрашивается ввод заново.

- Чтобы выйти из программы необходимо вести «0».

- При вводе существующих пунктов пользователю предоставляются данные, либо выходит.

- Заданы ограничения ввода при работе со стеком, то есть размер 1000.

Аварийные выходы из программы

-Выходов аварийных нет, в том случае, если программа не будет принудительна закрыта.

-Выход осуществляется только пунктом меню «0».

-В ошибочных случаях выводится сообщение об ошибке и ввод повторяется, пока программа не получит корректные данные.

Описание алгоритма

В реализации программы были созданы две структуры для хранения двух версий работы со стеком:

```
//Стек в виде массива
typedef struct array_stack
{
    char *symbol;
    int size;
} array_stack_t;

//Стек в виде связанного списка
typedef struct list_stack
{
    char symbol;
    struct list_stack *next;
} list_stack_t;
```

Алгоритм решения арифметического выражения:

Программа достает символ из стека. Происходит проверка на то, является ли символ числом. Если да то уже созданное число умножается на 10, и к нему добавляется текущее. Создание числа происходит до того момента, пока не будет считан арифметический знак(+, -, *, /). Число записывается в стек. Идет проверка на предыдущий арифметический знак. Если предыдущим арифметическим знаком были знаки «*» или «/», то достаётся из стека два числа(предыдущее и текущее). Происходит умножение или деление. Результат записывается в стек с числами, а текущий знак в стек с арифметическими знаками. Алгоритм продолжает работать, пока не будет считано последнее число, которое просто записывается в стэк с числами. Если был пользователем был введен посторонний символ, то выводится сообщение об ошибке.

Тестовые случаи

Ввод	Вывод	Проверяемый случай
Основное меню: c	Данные введены неверно.	Правильность ввода пункта меню
Основное меню: 15	Данные введены неверно.	Правильность ввода пункта меню
Пункт 1: ab23	Стек успешно создан!	Правильность ввода стека
Пункт 1: 23+12	Стек успешно создан!	Правильность ввода стека
Пункт 2: Введите количество новых элементов(доступно ..). Для выхода введите ‘0’: 0	Основное меню	Выход из пункта 2
Пункт 2: Введите количество новых элементов(доступно ..). Для выхода введите ‘0’: -1	Данные введены неверно!	Неправильность ввода количества новых элементов
Пункт 2: Введите количество новых элементов(доступно 995). Для выхода введите '0': 1000	Стек будет переполнен	Количество новых элементов превышает максимальный размер стека
Пункт 4: Пустой стек	Стек пуст	Вывод пустого стека
Пункт 4: Стек: 12 43	12 43	Вывод непустого стека
Пункт 5: ab23	Стек успешно создан!	Правильность ввода стека
Пункт 5: 23+12	Стек успешно создан!	Правильность ввода стека
Пункт 6: Введите количество новых элементов(доступно ..). Для выхода введите ‘0’: 0	Основное меню	Выход из пункта 2
Пункт 6: Введите количество новых элементов(доступно ..). Для выхода введите ‘0’: -1	Данные введены неверно!	Неправильность ввода количества новых элементов

Пункты 3/7: Введите количество элементов для удаления(доступно 6). Для выхода введите ‘0': 0	Основное меню	Выход из пункта 2
Пункты 3/7: Введите количество элементов для удаления(доступно 6). Для выхода введите ‘0': -1	Данные введены неверно!	Неправильность ввода количества новых элементов
Пункты 3/7: Введите количество элементов для удаления(доступно 6). Для выхода введите ‘0': 2	Элементы удалены	Правильность удаления элементов
Пункт 4: Пустой стек	Стек пуст	Вывод пустого стека
Пункт 4: Стек: 12 43	12 43	Вывод непустого стека
Пункт 9: Удаления не происходили	Элементы не удалялись, либо память освобождена.	Вывод удаленных элементов стека
Пункт 9: Удаление произошло	Вывод адрессов удаленных элементов: 0x7fcbddf05a70 0x7fcbddf05a60	Вывод удаленных элементов стека
Пункт 10: 213№123	Данные введены неверно	Правильность ввода
Пункт 10: 10-10	Result: 0 Time: 0.000024 Result: 0 Time: 0.000009	Правильность решения арифметического выражения
Пункт 10: 2+2*2	Result: 6 Time: 0.000024 Result: 6 Time: 0.000009	Правильность решения арифметического выражения

Сравнение эффективности:

Размерность от 100 до 1000:

	Стек Массив	Стек Лист
Рамерность	100	100
Добавление	0.000008 с	0.000016 с
Удаление	0.000002 с	0.000018 с
Память	104 байт	1600 байт

Процентное соотношение добавление: 200.000000 %
Процентное соотношение удаления: 900.000000 %

	Стек Массив	Стек Лист
Рамерность	200	200
Добавление	0.000006 с	0.000032 с
Удаление	0.000003 с	0.000027 с
Память	204 байт	3200 байт

Процентное соотношение добавление: 533.333333 %
Процентное соотношение удаления: 900.000000 %

	Стек Массив	Стек Лист
Рамерность	300	300
Добавление	0.000008 с	0.000043 с
Удаление	0.000004 с	0.000040 с
Память	304 байт	4800 байт

Процентное соотношение добавление: 537.500000 %
Процентное соотношение удаления: 1000.000000 %

	Стек Массив	Стек Лист
Рамерность	400	400
Добавление	0.000010 с	0.000062 с
Удаление	0.000004 с	0.000051 с
Память	404 байт	6400 байт

Процентное соотношение добавление: 620.000000 %
Процентное соотношение удаления: 1275.000000 %

	Стек Массив	Стек Лист
Рамерность	500	500
Добавление	0.000013 с	0.000077 с
Удаление	0.000005 с	0.000063 с
Память	504 байт	8000 байт

Процентное соотношение добавление: 592.307692 %
Процентное соотношение удаления: 1260.000000 %

	Стек Массив	Стек Лист
Рамерность	600	600
Добавление	0.000014 с	0.000085 с
Удаление	0.000006 с	0.000075 с
Память	604 байт	9600 байт

Процентное соотношение добавление: 607.142857 %
Процентное соотношение удаления: 1250.000000 %

	Стек Массив	Стек Лист
Рамерность	700	700
Добавление	0.000017 с	0.000105 с
Удаление	0.000012 с	0.000103 с
Память	704 байт	11200 байт

Процентное соотношение добавление: 617.647059 %
Процентное соотношение удаления: 858.333333 %

	Стек Массив	Стек Лист
Рамерность	800	800
Добавление	0.000019 с	0.000117 с
Удаление	0.000007 с	0.000111 с
Память	804 байт	12800 байт

Процентное соотношение добавление: 615.789474 %
Процентное соотношение удаления: 1585.714286 %

	Стек Массив	Стек Лист
Рамерность	900	900
Добавление	0.000020 с	0.000128 с
Удаление	0.000008 с	0.000113 с
Память	904 байт	14400 байт

Процентное соотношение добавление: 640.000000 %
Процентное соотношение удаления: 1412.500000 %

	Стек Массив	Стек Лист
Рамерность	1000	1000
Добавление	0.000023 с	0.000146 с
Удаление	0.000009 с	0.000124 с
Память	1004 байт	16000 байт

Процентное соотношение добавление: 634.782609 %
Процентное соотношение удаления: 1377.777778 %

Вывод:

- Использование стека в виде массива по времени более эффективно, чем стека в виде списка:
  - Добавление при больших размерах эффективнее в 5 - 6 раз
  - Удаление в 10 - 15 раз

- При больших размерах эффективнее хранить стек, как массив.

Контрольные вопросы

Что такое стек?

Стек – это структура данных - последовательный список с переменной длиной, в котором включение и исключение элементов происходит только с одной стороны – с его вершины.

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

При реализации стека при помощи связанного списка: **(sizeof(type) + sizeof(*type_t))*count**,

где **count** — число элементов, **type** — тип элементов, *type_t* — тип узла.

При реализации стека при помощи массива: **sizeof(type)*count**,

где **count** — число элементов, **type** — тип элементов.

При реализации стека, как массив память выделяется сразу же, то есть один буфером, а при стека как лист память выделяется под каждый элемент, в зависимости от введенного количество элементов.

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

При удалении элементов в стеке-массива очищение массива происходит только при выходе из программы.

При удалении элементов в стеке-списка очищение каждый элемент.

Что происходит с элементами стека при его просмотре?

Элементы стека извлекаются из стека — уничтожаются.

Каким образом эффективнее реализовывать стек? От чего это зависит?

Реализовывать стек при помощи списка эффективнее в том, что память для него выделяется в куче и ограничена размером оперативной памяти, в то время как для массива память ограничена размером стека. По времени работы реализация стека при помощи массива эффективнее.