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 Нисуев Нису Феликсович

Группа ИУ7 – 32Б

Преподаватель Барышникова Марина Юрьевна

2023 г.

Оглавление

Описание условия задачи

Цель работы: 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.

Создать программу работы со стеком, выполняющую операции добавления, удаления элементов и вывода 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.

Проверить правильность расстановки скобок трех типов (круглых, квадратных и фигурных) в выражении.

Описание технического задания

Входные данные:

1. **Номер команды**: целое число в диапазоне {-1} ∪ [1; 4].
2. **Дополнения к таблице:** строковое или целочисленное поле (в зависимости от команды)

Выходные данные:

1. Текущее состояние стека на массиве и списке (Адреса освобожденных адресов)
2. Результат проверки валидности скобок

Обращение к программе:

Запускается через терминал командой: ./build/app.exe

Аварийные ситуации:

1. Ввод некорректного пункта меню
2. Удаление массива из пустого стека
3. Переполнение стека
4. Некорректный ввод выражения со скобками

НАБОР ТЕСТОВ

№	Название теста	Пользовательский ввод	Вывод
Негативные тесты			
1	Некорректные пункт меню	99	ERROR: Incorrect action
2	Некорректный ввод	1 про	ERROR: Incorrect input
3	Некорректная команда	1 Q 10	ERROR: Incorrect choice
4	Удаление из пустого стека	2	Stack is empty
5	Переполнение стека	{В стеке 100 элементов} 1 F 0	Stack overflow
6	Некорректный ввод выражения	3 {\n}	ERROR: Incorrect input
7	Слишком длинное вражение	3 {1000+ левых скобок}	ERROR: Stack buff overflow
Позитивные тесты			
1	Запись валидного элемента в стек	1 F 1	Element pushed successfully

2	Удаление элемента из стека с данными	2	Poped element {element}
3	Вывод стеков	4 0	{Стеки и стек свободных адресов}
4	Невалидное выражение	3 ((	[{Время проверки} mcs] Brackets are invalid
5	Валидное выражение	3 (())	[{Время проверки} mcs] Brackets are valid

Описание структуры данных

```c
/// @def STACK_SIZE - максимальный размер стека
#define STACK_SIZE 1000

/* Стек на основе односвязного списка */

/// @typedef node_t -  Узел стека
typedef struct node {
    char el; // Элемент
    size_t index; // Индекс элемента в стеке
    struct node *prev; // Предыдущий элемент
} node_t;

/// @typedef lstack_t - Стек на основе односвязного списка
typedef struct {
    size_t capacity; // Вместимость стека
    node_t *top; // Вершина стека
} lstack_t;

/// @typedef free_adresses_t - Стек свободных адресов
typedef struct {
    size_t size; // Размер стека
    void *adresses[STACK_SIZE]; // Свободные адреса
} free_adresses_t;

/* Стек на основе статического */

/// @typedef astak_t - Стек на основе статического массива
typedef struct {
    size_t capacity; // Вместимость стека
    char stack[STACK_SIZE]; // Массив елементов стека
    char *el; // Вершина стека
} astack_t;
```

Замерный эксперимент

При замере добавления и удаления замеряется указанное кол-во раз добавления и удаления соответственно. При замере проверки валидности передается строка указанного размера

	Size	Measures	Arr stack	List stack
Push	10	Time, mcs	2	6
		Mem, b	26	256
	100	Time, mcs	6	18
		Mem, b	116	2416
	1000	Time, mcs	98	233
		Mem, b	1016	24016
Pop	10	Time, mcs	4	5
		Mem, b	26	256
	100	Time, mcs	8	10
		Mem, b	116	2416
	1000	Time, mcs	71	84
		Mem, b	1016	24016
Brackets	10	Time, mcs	2	5
		Mem, b	26	256
	100	Time, mcs	7	10
		Mem, b	116	2416
	1000	Time, mcs	47	53
		Mem, b	1016	24016

Вывод: Во всех аспектах стек на основе статического массива лучше

Описание алгоритма

1. После запуска программы пользователю предлагается ввести пункт меню.

2. Пользователь вводит целочисленные или символьные данные, в зависимости от пункта меню.

3. При добавлении элемента на стек, если он на массиве, то он записывается в первую свободную ячейку массива и увеличивает длину массива на 1. Если элемент добавляется в стек на списке – выделяется память под новый узел списка и он создаётся: в узел записывается значение символа а индекс инициализируется большим на единицу значением, чем то, что находится в предыдущем узле (на который указывает ссылка).

4. При удалении элемента со стека на
   а) Массиве: его длина уменьшается на 1, но сам элемент никуда из ячейки памяти не уходит – он будет перезаписан следующим добавленным элементом.
   б) Списке: адрес удалённого элемента помещается в массив освобождённых адресов памяти, а узел списка стирается из памяти.

5. При проверке валидности скобок при нахождении левых скобок они добавляются в стек, при нахождении правых скобок удаляется левая скобка из стека и сравнивается с найденной правой скобкой

Ответы на контрольные вопросы

1. Что такое стек?

Стек – структура данных, работающая по принципу «последний пришёл – первый вышел» (**LIFO**).

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

При реализации стека на (статическом) массиве его размер зависит от установленного максимального кол-ва элементов. В моём случае стек на массиве занимает (STACK_SIZE + 8) байт. При реализации стека на

списке его размер динамичен – для каждого элемента выделяется новая область памяти. В моём случае – по 19 байт на элемент.

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

При удалении элемента со стека на
а) Массиве: его длина уменьшается на 1, но сам элемент никуда из ячейки памяти не уходит – он будет перезаписан следующим добавленным элементом.
б) Списке: адрес удалённого элемента помещается в массив освобождённых адресов памяти, а узел списка стирается из памяти.

4. Что происходит с элементами стека при его просмотре?

Создаётся копия стека, куда помещаются все элементы. Элементы стека поочерёдно «вытаскиваются» из стека и выводятся на экран. Далее в стек возвращаются элементы в исходном порядке.

5. Каким образом эффективнее реализовывать стек? От чего это зависит?

Разница подходов минимальна на малых данных. При данных, кол-во которых больше нескольких десятков, реализация стека на массиве выигрывает по всем параметрам, кроме удаления элементов (и только при больших данных больше 1000). Если Вы знаете, сколько максимум данных должно храниться в стеке – используйте массив. Если вам нужен динамический стек – используйте связный список.

Вывод

Реализация стека на массиве выигрывает в подавляющем количество случаев и по скорости, и по памяти. Поэтому, если максимальное кол-во записей известно заранее, стоит использовать именно эту реализацию. Для динамического стека подойдёт односвязный список. Но за это придётся жертвовать в первую очередь памятью, т.к. память выделяется под каждый узел.