|  |  |
| --- | --- |
|  | **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 ПО ДИСЦИПЛИНЕ ТИПЫ И СТРУКТУРЫ ДАННЫХ**

Студент **Паншин Сергей Константинович**

Группа **ИУ7-33Б**

Название предприятия **НУК ИУ МГТУ им. Н. Э. Баумана**

|  |  |
| --- | --- |
| Студент | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Паншин С. К.** |
| Преподаватель | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **Никульшина Т. А.** |
| Преподаватель | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **Барышникова М. Ю.** |

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*2023 г.*

**Условие задачи**

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

**Техническое задание**

Реализовать операции работы со стеком, который представлен в виде статического массива и в виде односвязного списка, оценить преимущества и недостатки каждой реализации, получить представление о механизмах выделения и освобождения памяти при работе с динамическими структурами данных.

**Входные данные:**

Пункт меню (число от 0 до 8):

Menu:

1) Push array stack

2) Pop array stack

3) Print array stack

4) Push list stack

5) Pop list stack

6) Print list stack

7) Check brackets sequence

8) Print stacks compare

0) Quit program

Также добавляемый символ, строка со скобочной последовательностью.

**Выходные данные:**

Элементы стека и Yes/No в проверке скобочной последовательности.

**Возможные аварийные ситуации:**

Некорректный ввод: пункта меню, не символ, попытка очистить пустой стек, попытка добавить элемент в полный стек.

**Способ обращения к программе**

В папке с программой запустить команду *make run*.

**Структуры данных**

// элемент списка

typedef struct list\_node

{

char value;

struct list\_node \*prev\_el;

} list\_node\_t;

// стек в виде списка

typedef struct stack\_list

{

list\_node\_t \*ptr;

size\_t size;

void \*\*free\_area;

size\_t free\_area\_size;

size\_t free\_area\_size\_alloc;

} stack\_list\_t;

// стек в виде статического массива

typedef struct stack\_array

{

char \*data;

size\_t size;

} stack\_array\_t;

// используемые дефайны

#define ITER\_COUNT\_TIME 50

#define MAX\_STACK\_LEN 1000

#define MAX\_STR\_LEN 100

#define ERROR\_MEMORY 1

#define ERROR\_EMPTY\_STACK 2

#define ERROR\_STACK\_OVERFLOW 3

#define ERROR\_EMPTY\_INPUT 4

#define ERROR\_STR\_LEN 5

#define ERROR\_WRONG\_NUM 6

#define ERROR\_WRONG\_MENU\_ITEM 8

#define ERROR\_INVALID\_NUM 9

// удаление из стека в виде списка

int stack\_list\_pop(stack\_list\_t \*list);

// добавление в стек в виде списка

int stack\_list\_push(stack\_list\_t \*list, char value);

// вывод на экран данных стека в виде списка

void stack\_list\_print(stack\_list\_t \*list);

// освобождение памяти из под стека в виде списка

void stack\_list\_free(stack\_list\_t \*stack);

// выделение памяти под массив свободных адресов

int init\_free\_area(stack\_list\_t \*stack, size\_t size);

// добавление адреса в список свободных

int add\_free\_area(stack\_list\_t \*stack, void \*ptr);

// удаление из стека в виде массива

int stack\_array\_pop(stack\_array\_t \*array);

// добавление в стек в виде массива

int stack\_array\_push(stack\_array\_t \*array, char value);

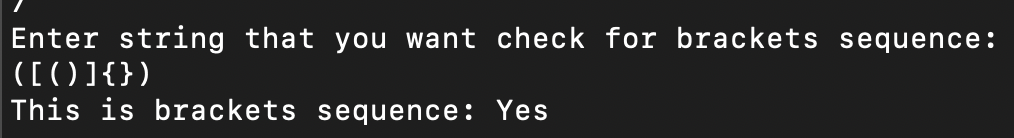
// вывод на экран данных стека в виде массива

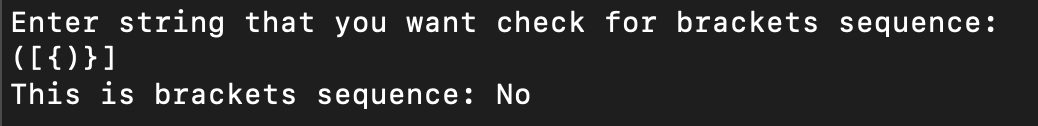
void stack\_array\_print(stack\_array\_t \*array);

**Алгоритм**

В стек помещается скобка если она «открывающего вида», если скобка «закрывающего вида», то проверяется если элемент стека скобка «открывающего вида» такого же типа (круглая/квадратная/фигурная), то вытаскиваем ее из стека, иначе возвращаем что не скобочная последовательность. Делаем действия выше пока не закончиться строка или не определим, что не скобочная последовательность.

**Пример работы**



****

**Замеры**

Производиться 50 итераций при замерах.

Время выполнения проверки скобочной последовательности в наносекундах:

|  |  |  |
| --- | --- | --- |
| **Длина строки** | **Стек в виде массива** | **Стек в виде списка** |
| 8 | 480 | 1080 |
| 14 | 800 | 1620 |
| 26 | 1220 | 2940 |
| 52 | 2880 | 5420 |
| 104 | 4320 | 14780 |
| 182 | 7820 | 19200 |
| 286 | 12160 | 30020 |

Память, занимаемая при проверке скобочной последовательности в байтах:

|  |  |  |  |
| --- | --- | --- | --- |
| **Длина cтроки** | **Максимальный размер занимаемой памяти массивом** | **Выделенная память под массив** | **Максимальный размер занимаемой памяти списком** |
| 8 | 4 | 1000 | 64 |
| 14 | 7 | 1000 | 112 |
| 26 | 8 | 1000 | 128 |
| 52 | 21 | 1000 | 336 |
| 104 | 42 | 1000 | 672 |
| 182 | 50 | 1000 | 800 |
| 286 | 90 | 1000 | 1440 |

**Выводы по проделанной работе**

Стек, реализованный связанным списком, проигрывает как по памяти, так и по времени обработки. Таким образом, можно сделать вывод, что если нужно реализовать такую структуру данных как стек, то лучше использовать массив, а не связанны список.

**Контрольные вопросы**

*Что такое стек?*

Стек – структура данных, в которой можно обрабатывать только последний добавленный элемент. На стек действует правило LIFO — последним пришел, первым вышел.

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

При хранении стека с помощью списка, память всегда выделяется в куче. При хранении с помощью статического массива, память выделяется на стеке. Для каждого элемента стека, реализованного списком, выделяется на 4 или 8 байт (на большинстве современных ПК) больше, чем для элемента массива. Эти дополнительные байты занимает указатель на следующий элемент списка. Размер указателя (4 или 8 байт) зависит от архитектуры. Также из-за выравнивания выделяется больше памяти под символ. Поэтому размер больше в 8 или в 16 раз соответственно.

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

При хранении стека связанным списком, верхний элемент удаляется путем освобождения памяти для него и смещения указателя, указывающего на начало стека. При удалении из стека, реализованного массивом, смещается лишь указатель на вершину стека.

*Что происходит с элементами стека при его просмотре?*

Элементы стека уничтожаются, так как каждый раз достается верхний элемент стека.

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

Реализовывать стек эффективнее с помощью массива. Он выигрывает как во времени обработки, так и в количестве занимаемой памяти. Но, если не известен размер стека, то в таком случае стоит использовать списки.