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 (ИУ)

КАФЕДРА: Программное обеспечение ЭВМ и информационные технологии (ИУ7)

ОТЧЕТ ПО ЛАБРАТОРНОЙ РАБОТЕ №4

Название предмета: Типы и структуры данных

Студент: Кашима Ахмед

Группа: ИУ7-33Б

2022 г.

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.

Условие задачи (вар 4):
Проверить правильность расстановки скобок трех типов (круглых, квадратных и фигурных) в выражении

Входные данные:
Элементы стека или выражение.
Выходные данные:
Флаг, указывающий на корректность выражения.
Функция программы:
Реализация операции работы со стеком; определение правильности расстановки скобок в выражении.

Обращение к программе:

Исполняемый файл app.exe создается путем автоматической сборки проекта с помощью файла makefile. Для выполнения работы следует запустить данный исполняемый файл без каких-либо аргументов.

Функция программы: программа выполняет ряд функций, указанных при её первом запуске. Она позволяет:

1. Ввести элементы стека.
2. Добавить элемент в стек.
3. Удалить элемент из стека.
4. Вывести элементы стека в виде списка.
5. Вывести аддрес элементы которые удалились.

6. Ввести элементы стека.
7. Добавить элемент в стек.
8. Удалить элемент из стека.
9. Вывести элементы стека в виде массива-стека.

Аварийные ситуации:

1. Некорректный ввод номера команды.
   На входе: число, большее чем 8 или меньшее, чем 0.
   На выходе: сообщение «Incorrect Input! Try again.»

2. **Некорректный ввод элемента в стек так что у меня храить только char поэтому вводный символ должно быть только на диарозоне -128 до 127.**

Структуры данных:

Структура узла для списка:

```
typedef struct node
{
char data;
struct node *next;
} node_t;
}
```

data- для хранение данных

next- указатель на следующий

Реализация массива-стека:

```
typedef struct
{
char *data;
int used_elems;
} arr_stack_t;
      data- указатель на массива-стека
      used_elements – колечество элементы в массива-стека
```

Функции:

Взаимодействие со стеком-массивом:

```
char pop_arr(arr_stack_t *stk);

void push_arr(arr_stack_t *stk, char data);

void print_arr_stack(arr_stack_t *stk);
```

Взаимодействие со стеком-списком:

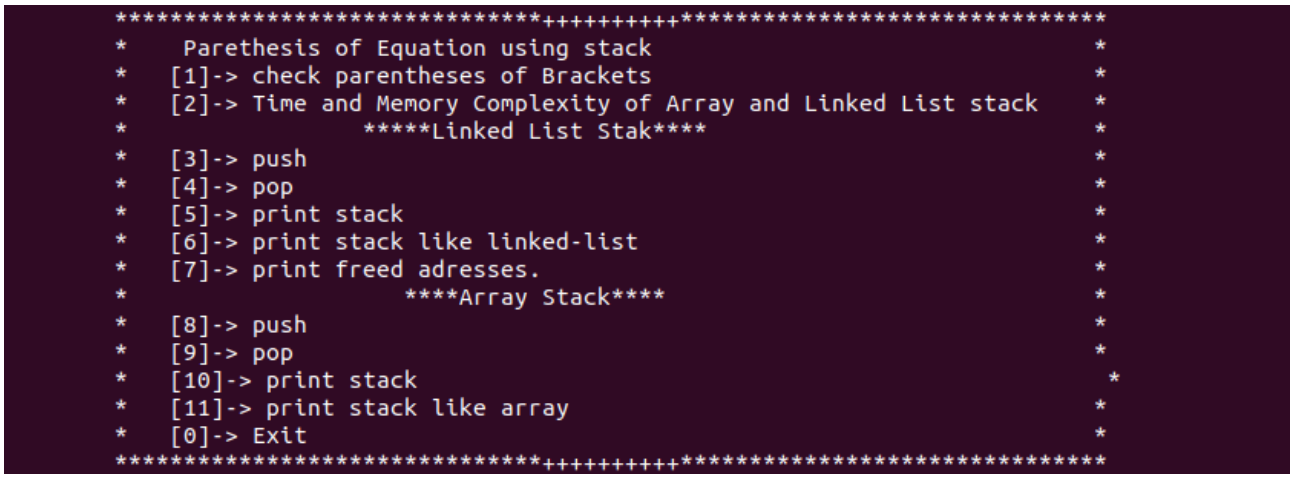
```
void print_stack(stack_t *stk);

void free_linked_list_stk(stack_t *stk);

char pop(stack_t *stk);

void push(stack_t *stk, node_t *node);
```

Анализ расстановки скобок:

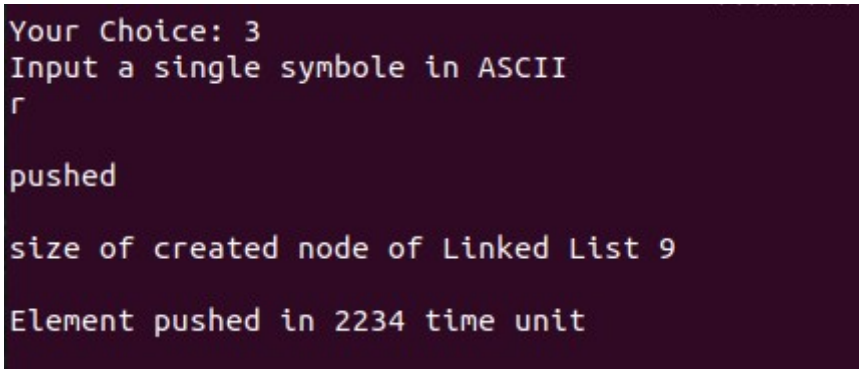
```
void check_string_using_ll(char *string);

void check_string_using_arr(char *string);
```

Интерфейс:

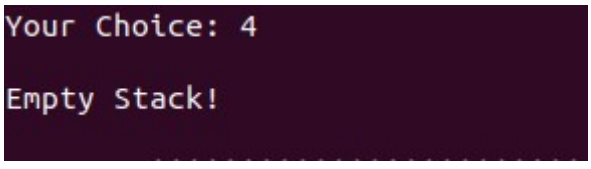
Главное меню:



Ввод в стек:



Вывод пустого стека:



Оценка эффективности

Измерения эффективности реализаций стека будут производиться в единицах измерения – тактах процессорв. Для измерения была специально написана ассемблерная функция, поэтому погрешность измерений минимальна. При записи результатов использовалось среднее количество тактов, полученное по результатам 10 измерений.

Время добавления элемента (в тактах процессора):

Количество элементов	Массив	Список
10	1446	8046
100	16844	81632
1000	180642	1116864

Время удаления элемента (в тактах процессора):

Количество элементов	Массив	Список
10	1654	3654
100	9498	42548
1000	71235	287434

Объём занимаемой памяти (в байтах):

Количество элементов	Массив	Список
10	18	94
100	108	904
1000	1008	9004

Контрольные вопросы

1. Что такое стек?

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

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

При хранении стека с помощью списка, то память всегда выделяется в куче. При хранении с помощью массива, память выделяется либо в куче, либо на стеке (в зависимости от того, динамический массив или статический). 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.

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

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

4. Что происходит с элементами стека при его просмотре?

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

5. 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 (в классическом случае). Вариант хранения списка может выигрывать только в том случае, если стек реализован статическим массивом. В этом случае, память для списка ограничена размером оперативной памяти (так как память выделяется в куче), а память для статического массива ограничена размером стека.

Вывод

Стек, реализованный связанным списком, внушительно (в 4 раза по сравнению со временем, достигнутым реализацией на основе массива) проигрывает как по времени, так и по памяти в данной реализации. Таким образом, можно сделать вывод, что если нужно реализовать такую структуру данных как стек, то лучше использовать массив, а не связанный список.