Отчет по 4 лабораторной работе
По дисциплине «Типы и структуры данных»

Подготовил Пересторонин Павел
Группа ИУ7-33Б
Вариант 15

Условие задачи

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.

Техническое задание

Элементами стека являются адреса памяти. При реализации массивами - их вводить, при реализации списком – брать адрес выделенной памяти под элемент.

Входные данные

Требуемое количество элементов для добавления/удаления элементов.
Значения элементов (только при записи в стек-массив).

Выходные данные

Временная статистика и информация о фрагментации.

Возможные аварийные ситуации

Некорректный ввод, ошибки выделения памяти.

Структуры данных

Структура узла стека-списка:

```
typedef struct node_t
{
    void *value;
    struct node_t *next;
    int number;
} node_t;
```

value — значение, хранящееся в узле.
Next — указатель на следующий элемент.
Number — номер элемента в стеке (в моем случае помогает отслеживать переполнение)

Структура стека-массива:

```
typedef struct
{
   void *data[ARRAY_SIZE];
   int cur_index;
} stack_a;
```

data — массив значений размера ARRAY_SIZE;
cur_index — указатель «за голову» стека (то есть верхний элемент стека лежит по индексу cur_index - 1)

Алгоритм

При добавлении элемента в список сначала добавляется элемент, затем проверяется, был ли он взят из списка свободных областей.

При удалении элемента из стека списка элемент удаляется, а затем по возвращенному функцией «pop» значению в список свободных областей добавляется новый адрес.

При работе со стеком-массивом просто в опредленном интервале ([0,ARRAY_SIZE]) сдвигается указатель cur_index (описан выше).

Тесты

1. Добавление в стек-список

1.1. Добавление корректного количества элементов, не переполняющее массив

```
Введите количество элементов (от 1 до 1000): 6
Элементы добавлены.
```

1.2. Добавление некорректного количества элементов

```
Введите количество элементов (от 1 до 1000): 1001
Неверное значение количества элементов.
```

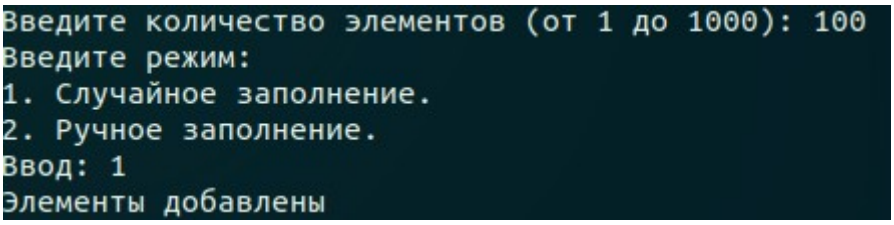
1.3. Добавление количества элементов, переполняющего стек-список

```
Введите количество элементов (от 1 до 1000): 999
Кажется, стек переполнился... (Количество добавленных элементов: 994)
```

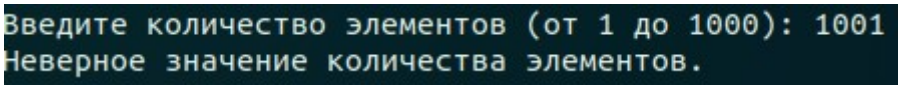
2. Добавление в стек-массив.

2.1. Случайное заполнение.

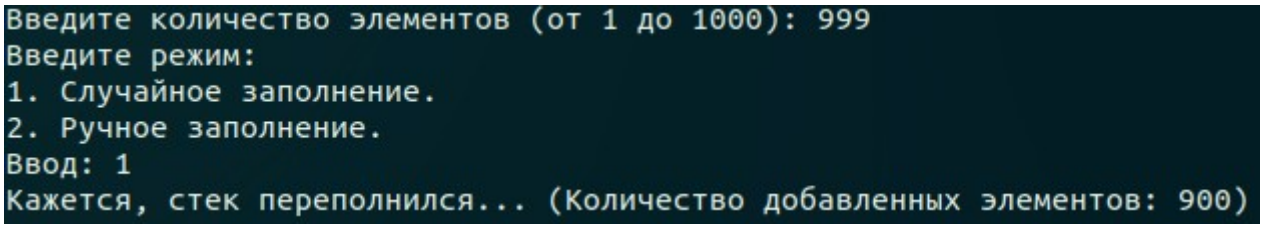
2.1.1. Добавление корректного количества элементов



2.1.2. Добавление некорректного количества элементов.



2.1.3. Добавление количества элементов, переполняющего стек-массив.



2.2. Ручное заполнение.

2.2.1. Добавление корректного количества элементов.

```
Введите количество элементов (от 1 до 1000): 6
Введите режим:
1. Случайное заполнение.
2. Ручное заполнение.
Ввод: 2
Введите адрес в виде десятичного или шестнадцатиричного числа
(помните, что адрес должен быть в пределах от 0 до 0xFFFFFFFF):
Число №1: 0x666
Число №2: 0x222
Число №3: 2323232323
Число №4: 0x1
Число №5: 0
Число №6: 1000
Элементы добавлены
```

2.2.2. Добавление некорректного количества элементов.

```
Введите количество элементов (от 1 до 1000): 1001
Неверное значение количества элементов.
```

2.2.3. Добавление количества элементов, переполняющего стек.

```
Введите количество элементов (от 1 до 1000): 100
Введите режим:
1. Случайное заполнение.
2. Ручное заполнение.
Ввод: 2
Введите адрес в виде десятичного или шестнадцатиричного числа
(помните, что адрес должен быть в пределах от 0 до 0xFFFFFFFF):
Число №1: 1
Число №2: 2
Число №3: 3
Число №4: 4
Число №5: 5
Кажется, стек переполнился... (Количество добавленных элементов: 4)
```

3. Очищение стека-списка.

3.1. Очищение корректного кол-ва элементов.

```
Введите количество элементов, которое нужно удалить
(от 1 до 1000; примечание: при вводе числа, большего количества элементов в стеке
стек очистится полностью): 100
Нужное количество элементов очищено!
```

3.2. Попытка очищения слишком большого количества элементов.

```
Введите количество элементов, которое нужно удалить
(от 1 до 1000; примечание: при вводе числа, большего количества элементов в стеке
стек очистится полностью): 1001
Неверное значение количества элементов.
```

3.3. Очистка количества элементов, большего, чем количество элементов в стеке.

```
Введите количество элементов, которое нужно удалить
(от 1 до 1000; примечание: при вводе числа, большего количества элементов в стеке
стек очистится полностью): 1000
Кажется, стек закончился... (Количество очищенных элементов: 900)
```

4. Очищение стека-массива.

4.1. Очищение корректного кол-ва элементов.

```
Введите количество элементов, которое нужно удалить
(от 1 до 1000; примечание: при вводе числа, большего количества элементов в стеке
стек очистится полностью): 100
Нужное количество элементов очищено!
```

4.2. Попытка очищения слишком большого количества элементов.

```
Введите количество элементов, которое нужно удалить
(от 1 до 1000; примечание: при вводе числа, большего количества элементов в стеке
стек очистится полностью): 1001
Неверное значение количества элементов.
```

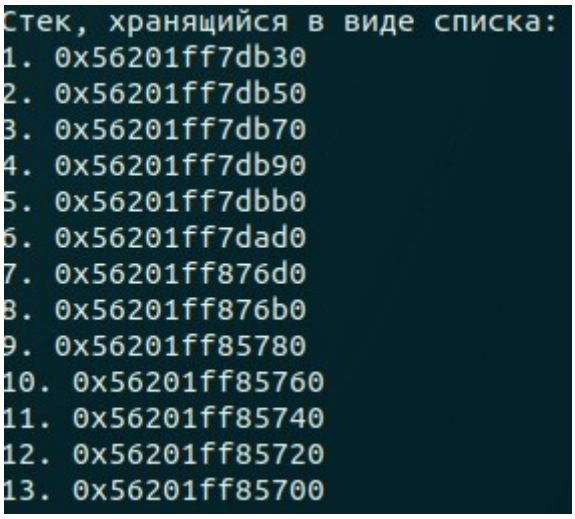
4.3. Очистка количества элементов, большего, чем количество элементов в стеке.

```
Введите количество элементов, которое нужно удалить
(от 1 до 1000; примечание: при вводе числа, большего количества элементов в стеке
стек очистится полностью): 1000
Кажется, стек закончился... (Количество очищенных элементов: 900)
```

5. Вывод стека-списка.
5.1. Пустой стек.

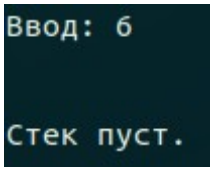
```
Ввод: 5


Стек пуст.
```

5.2. Непустой стек.

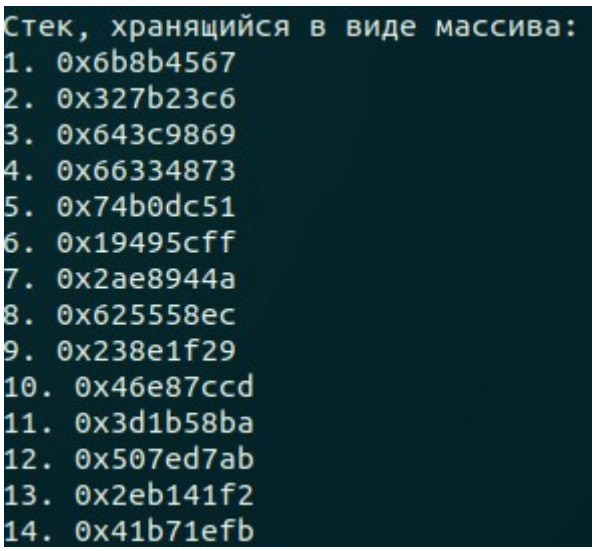


6. Вывод списка-массива.

6.1. Пустой стек.



6.2. Непустой стек.



7. Вывод массива свободных областей.

7.1. Массив пустой.

```
Ввод: 7


Количество элементов = 0;
```

7.2. Массив непустой.

```
Количество элементов = 14;
1. 0x55d25b8e5aa0
2. 0x55d25b8e5aa8
3. 0x55d25b8e5ab0
4. 0x55d25b8e5ab8
5. 0x55d25b8e5ac0
6. 0x55d25b8e5ac8
7. 0x55d25b8e5ad0
8. 0x55d25b8e5ad8
9. 0x55d25b8e5ae0
10. 0x55d25b8e5ae8
11. 0x55d25b8e5af0
12. 0x55d25b8e5af8
13. 0x55d25b8e5b00
14. 0x55d25b8e5b08
```

8. Вывод статистики.

8.1. Еще не было выделений памяти.

```
Количество памяти, взятой по-новой: 0
Количество памяти, взятой из уже использованной: 0
```

8.2. Еще не было освобождений памяти, но было выделение.

```
Количество памяти, взятой по-новой: 14
Количество памяти, взятой из уже использованной: 0
```

8.3. Было выделение после освобождения.

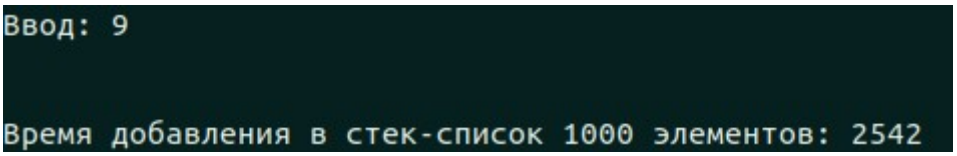
```
Количество памяти, взятой по-новой: 24
Количество памяти, взятой из уже использованной: 7
```

9. Вывод времени вставки и удаления.

9.1. Ни одного времени еще не могло быть замерено.

```
Ввод: 9


Время еще не замерялось...
```

9.2. Было замерено хотя бы одно удаление/добавление.



Расчеты времени и памяти.

Время

(время измерялось в тиках для большей наглядности)

push

Количество добавляемых элементов	Стек-список	Стек-массив
100	61884	9121
500	250117	50129
1000	331981	80332

pop

Количество удаляемых элементов	Стек-список	Стек-массив
100	22904	8113
500	54346	32799
1000	80332	63796

Память

(в данной таблице размер стека-массива указан минимально возможный, в общем случае для хранения, к примеру 100 элементов, памяти мы выделим на 1000 (все зависит от разброса количества элементов))

Количество элементов в стеке	Стек-список	Стек-массив
100	2000 байт	804 байт
1000	20000 байт	8004 байт
10000	200000 байт ~ 200кБ	80004 байт ~ 80кБ

Выводы по проделанной работе

Из расчетов времени видно, что операции добавления и удаления элементов быстрее в стеке-массиве, однако у него есть один минус: если количество элементов имеет большой разброс и в теории, например, среднее значение составляет 5% от максимального, массив будет хранить слишком много избытычного объема памяти. Таким образом, когда среднее значение составляет меньше 40% от максимального (в таком случае стек-список при средней заполненности будет занимать примерно такой же объем памяти, как и стек-массив; рассчитано из таблицы с памятью), разумнее использовать стек-массив, в противном случае — стек-список.

Контрольные вопросы

1. Что такое стек?

Стек — структрура данных, работающая по принципу LIFO (last in — first out), поддерживающая 2 операции: push — вставка в стек; pop — удаление (с чтением) из стека.

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

От 8 байт на элемент в виде непрерывного куска при реализации массивом.

От 16 байт (8 байт на данные, 8 байт на указатель на след элемент) в виде «узлов» при реализации списком.

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

При удалении элемента из стека-списка удаляется узел. При удалении элемента из стека-массива (если массив статический) просто перемещается указатель на текущий элемент и память не меняется.

4. Что происходит с элементами стека при его просмотре?

Чтобы вывести (просмотреть) стек, учитывая, что эта СД допускает только push и pop, нужно удалять вершины стека и выводить их последовательно.

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

Эффективнее по времени выполнения операций стек-массив, однако при не имении примерной оценки расхода памяти под стек, или при сильной «пульсации» (то есть она сильно может разнится), лучше использовать стек-список (так как размер хранимой памяти пропорционален количеству элементов в стеке).