Отчет по 4 лабораторной работе

По дисциплине «Типы и структуры данных»

Подготовил Жабин Дмитрий

Группа ИУ7-34Б

Вариант 2

**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.

**Задача**

Создать программу работы со стеком, выполняющую операции добавления, удаления элементов и вывод текущего состояния стека. Реализовать стек: а) массивом; б) списком. 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.

При реализации стека массивом располагать два стека в одном массиве. Один стек располагается в начале массива и растет к концу, а другой располагается в конце массива и растет к началу. Заполнять и освобождать стеки произвольным образом с экрана. Элементами стека являются вещественные числа. Списком реализовать один стек.

**Внешняя спецификация программы.**

**Входные данные:** введённые пользователем вещественные числа.

**Выходные данные:** текущее состояние стека, массив свободных областей. Информация об эффективности реализации стека списком и массивом.

**Задача программы:**

Обеспечить выполнение стандартных операций со стеком с помощью двух его реализаций - списком и массивом. Сравнить затраты по времени и памяти в этих случаях.

**Способ обращения к программе:** Запуск программы в консоли (./app.exe).

**Возможные ошибки**

1) Некорректный тип вводимых данных

Сообщение “Wrong input”, добавление элемента не происходит, программа продолжает работу.

2) Ошибки выделения памяти

Сообщение "Memory error”, добавление элемента не происходит, программа продолжает работу.

3) Превышение максимального размера стека

Сообщение "Stack overflow”, добавление элемента не происходит, программа продолжает работу.

4) Попытка удаления элемента из пустого стека

Сообщение "Empty stack”, удаление элемента не происходит, программа продолжает работу.

**Структуры данных.**

Для двух реализаций стека выбраны следующие структуры данных:

1) для односвязного линейного списка:

typedef struct sNode{ - узел списка

float value; - значение элемента

struct sNode \*next; - адрес следующего узла в списке

} tNode;

typedef struct{ - сам список

tNode \*p\_begin; - адрес первого узла в списке

int size; - длина списка

} tStack;

2) для массива (используется для хранения двух стеков с максимальным суммарным размером в 100 элементов):

#define MAX\_STACK\_SIZE 50

float arr\_stack[2 \* MAX\_STACK\_SIZE];

**Функции**

void print\_empty\_adr(void); - функция печати массива свободных областей

void print\_list(tNode \*p\_begin); - функция печати списка

int stack\_push(tStack \*s, float value); - функция добавления элемента в список

int stack\_pop(tStack \*s, float \*value); - функция удаления элемента из списка

void delete\_list(tNode \*p\_begin); - функция освобождения памяти из-под узлов списка

tStack stack\_create(void); - функция создания списка

int arr\_push(float \*\*stack, int num, float value); - функция добавления элемента в массив

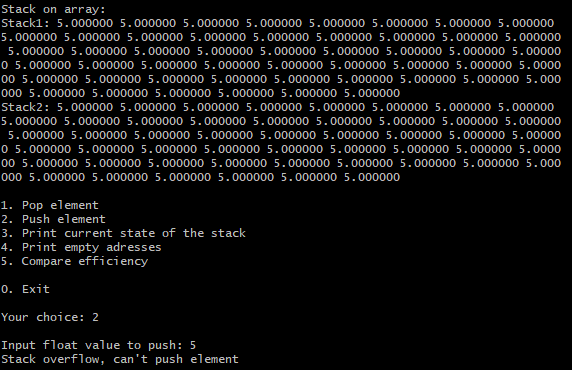
int arr\_pop(float \*\*stack, int num, float \*value); - функция удаления элемента из массива

void arr\_print(float \*stack, int num); - функция печати массива

**Тесты.**

|  |  |
| --- | --- |
| Выбор опций меню | Результат |
| 2)Push element -> 1)To stack on list  Input float value to push: abc | Wrong input |
| 2)Push element -> 1)To stack on list (x50)  2)Push element -> 1)To stack on list  Input float value to push: 5 | Stack overflow, can't push element |
| 1)Pop element -> 2)From 1st stack on array | Trying to pop from empty stack! |
| 2)Push element -> 1)To stack on list  Input float value to push: 3.75  2)Push element -> 1)To stack on list  Input float value to push: -2.09  2)Push element -> 1)To stack on list  Input float value to push: 10  2)Push element -> 1)To stack on list  Input float value to push: -0.8  1)Pop element -> 1)From stack on list  1)Pop element -> 1)From stack on list  3)Print current state of the stack  4)Print empty adresses | Stack on list:  -2.090000 0000000000981440  3.750000 0000000000981420  Stack on array:  Stack1: [Empty stack]  Stack2: [Empty stack]  Empty addresses:  0000000000981480  0000000000981460 |

Пример переполнения стека на массиве (суммарный размер двух стеков достиг 100 элементов)



**Оценка эффективности**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Размер стека | Время, тактов процессора | | | | Память, байт | |
| Push | | Pop | |
| Массив | Список | Массив | Список | Массив | Список |
| 10 | 18 | 4596 | 39 | 2063 | 200 | 176 |
| 20 | 18 | 3646 | 55 | 1775 | 336 |
| 40 | 18 | 3336 | 42 | 2261 | 656 |

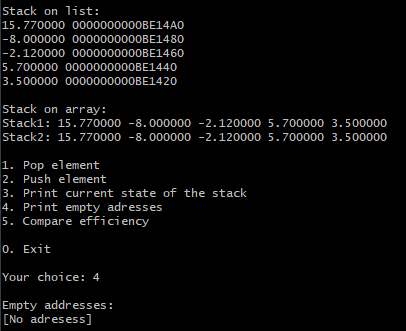
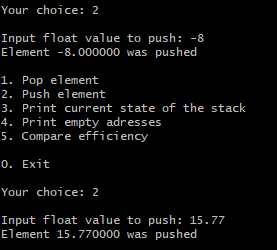
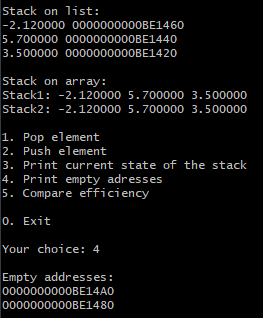
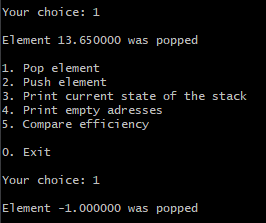
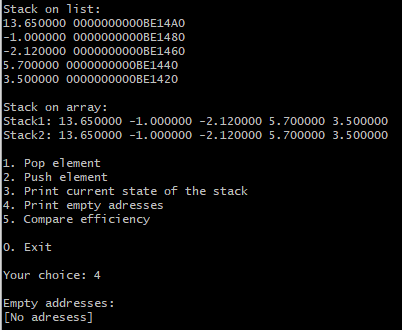
\*под массив резервируется память на 50 элементов (макс. размер стека)

**Вывод**

В результате работы были реализованы операции работы со стеком, а также сделаны обе реализации стека - через массив и через список. Массив дает преимущество по времени, но проигрывает по памяти при небольшом его заполнении, так как в списке размер необходимой памяти определяется текущим количеством элементов, а под массив необходимо выделить довольно большой (с запасом, если количество элементов заранее не известно) непрерывный кусок памяти, который не всегда есть достаточного размера. Массив использовать целесообразно при заранее известном количестве элементов. Реализация стека на списке эффективнее, чем на массиве, если размер стека во время работы зачастую оказывается значительно меньше размера массива.

В данной работе заполнение массива на 24% и больше обеспечивает выигрыш по памяти в сравнении с таким же количеством элементов на списке.

При работе программы фрагментация памяти почти не происходит (новые элементы записываются по адресам последних удаленных элементов):



**Контрольные вопросы**

**1) Что такое стек?**

Стек – это последовательный список с переменной длиной, в котором включение и исключение элементов происходит только с одной стороны – с его вершины. Стек функционирует по принципу: последним пришел – первым ушел, Last In – First Out (LIFO).

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

Массив: для хранения стека обычно отводится непрерывная область памяти ограниченного размера, имеющая нижнюю и верхнюю границу.

Список: при каждом включении элемента в стек происходит выделение области памяти, адрес которой записывается в указатель стека.

В случае статического массива объем памяти определяется заранее ожидаемым максимальным количеством элементов в стеке, в случае динамического массива и списка - количеством по факту добавленных в стек элементов, ограничением является лишь количество оперативной памяти, выделенной для работы программы.

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

Массив: при исключении элемента из стека сначала считываются данные, а затем происходит перемещение указателя PS к предыдущему элементу. Если указатель стека выходит за нижнюю границу массива, то стек пуст. Список: при исключении элемента сначала по указателю стека считывается информация об исключаемом элементе, а затем указатель смещается к предыдущему элементу. После чего освобождается память, выделенная под элемент. Если указатель имеет значение нулевого адреса, то стек пуст.

**4) Что происходит с элементами стека при его просмотре?**

Классическая реализация стека предполагает, что просмотреть содержимое стека без извлечения (удаления) его элементов невозможно.

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

Массив дает преимущество по времени и по памяти при почти полном его заполнении, иначе по памяти имеет преимущество список. Массив использовать целесообразно при известном количестве элементов стека.  
Также имеет значение тип элементов в стеке - использование списка для реализации стека эффективно по памяти, если объем памяти для хранения значения элемента значительно превышает объем памяти для хранения указателя на элемент.