**Типы и структуры данных.**

**Лабораторная работа №2**

## Работа со стеком.

**Выполнил**: Кузнецов Александр

**Группа**: ИУ7-33

**Цель работы:** реализовать операции работы со стеком, который представлен в виде массива (статического или динамического) 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.

Создать программу работы со стеком, выполняющую операции добавление, удаления элементов и вывод 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**.**

**Вариант 13 (задача 2)**

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.

**Входные данные:**

Действительные числа, команды для обработки стека;

**Пример:**

//ввод команды для стека

>>1

//ввод действительного числа в стек

Float: 2

Float: 4.0

**Выходные данные**

Список элементов стека, список адресов стека(для списка), освобожденные адреса стека(для списка), время на работу функций стека, процентное соотношение времени между временем массива и временем списка.

**Функция программы**

Работа с 2мя стеками в одном массиве; Работа со стеком в списке;

Показ адресов, отводимых под адресацию к элементам списка;

Вывод времени работы исполнительных функций над стеком и сравнение времени;

**Допущения**

* Вводятся только действительные числа в стек.
* Операция над стеком выбирается в соответствии с порядком в интерфейсе.

**Аварийные ситуации**

**-** Переполнение стека (MAX\_SIZE\_STACK = 1000000)

(Error: Stack is overflow!**)**

**-** Удаление элемента из пустого стека

**(**Error: Stack is empty!**)**

**-** Вывод пустого стека

**(**Mas\_Stack1 is empty!**)**

- Добавление элемента в стек не FLOAT

(incorrect input, please return input...)

- Команда над стеком не удовлетворяет интерфейсу

(incorrect input, please return input...)

**Описание использованных структур данных**

typedef struct stack\_mas{

float mas[MAX\_SIZE\_MAS]; //Массив с размером

int top1; //размер 1го стек(указатель)

int top2; //размер 2го стека(указатель)

} Stack\_mas;

typedef struct Stack\_list{

float num; //вводимый элемент списка

struct Stack\_list \*next;//указатель на след.элемент списка

} Stack\_list;

typedef struct Free\_list{

Stack\_list \*num; //указатель на свободную ячейку памяти

struct Free\_list \*next; //указатель на след. Элемент списка свободной области

} Free\_list;

**Алгоритм**

**Обработка ввода:**

1. Вывести интерфейс программы
2. Проверить корректность ввода(Интерфейс)
3. Проверить корректность ввода элемента(float)

-//////- Вывод сообщение об ошибке, если она есть -//////-

Иначе

**Обработка стеком:**

1. Передача (float) элемента в stack1 или stack2, если это массив или в stack, если это список

**Обработка операций над стеком:**

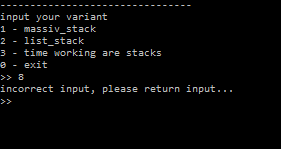
1. Добавление элемента в стек
2. Удаление элемента из стека
3. Вывод стека в консоль

**Обработка времени:**

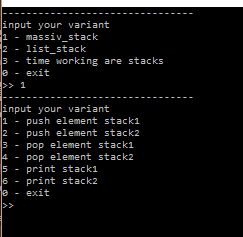
1. Расчет времени работы алгоритмов
2. Вывод в % соотношении на сколько список быстрее массива или наоборот.

**Тесты**

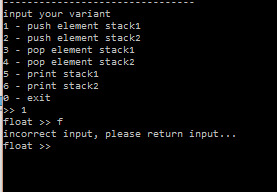
**Тест ввода:**Некорректный вод номера операции над стеком:



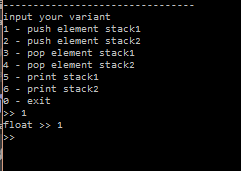
Корректный ввод номера операции над стеком:



Некорректное добавление элемента в stack(не float):

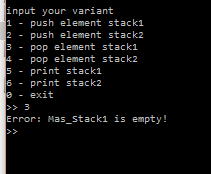


Корректное добавление элемента в stack(float):

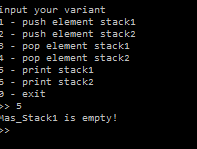


**Тест операций над стеком:**

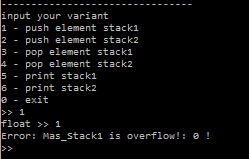
Удаление пустого элемента в стеке



Вывод пустого стека

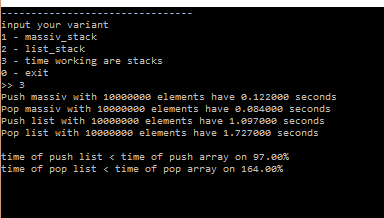


Переполнение стека:



**Сравнение методов**

По результатам выполнения программы, видно, что реализация стека массивом дает быстроту, по сравнению с реализацией стека списком, потому что при реализации списком происходит выделение и очистка памяти, в то время как при реализации массивом происходит только изменение значения элемента массива.



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Elements(N) | Push(mas)(s) | Push(list)(s) | Pop(mas)(s) | Pop(list)(s) |
| 1000000 | 0.015 | 0.205 | 0.016 | 0.301 |
| 10000000 | 0.122 | 1.097 | 0.084 | 1.727000 |
| 100000 | 0.000 | 0.015 | 0.000 | 0.016 |
| 500000 | 0.000 | 0.053 | 0.016 | 0.084 |
| 5000000 | 0.063 | 0.534 | 0.037 | 0.887 |
| 50000000 | 0.618 | 4.354 | 0.363 | 6.985 |

**Вопрос – Ответ**

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).

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

Если стек реализован в виде статического или динамического массива (вектора), то для его хранения обычно отводится непрерывная область памяти ограниченного размера, имеющая нижнюю и верхнюю границу.

При физической реализации стека в виде односвязного линейного списка дескриптор будет отличаться отсутствием верхней границы стека. В этом случае объем стека ограничивается только объемом доступной оперативной памяти.

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

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.

При исключенииэлемента 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.

1. **Что происходит с элементами стека при его просмотре?**

Принцип организации стека предполагает, что в текущий момент времени доступен элемент, индекс которого хранится в указателе стека PS. Граничные адреса стека – это параметры физической структуры. Физическая структура обычно дополняется дескриптором стека, в котором хранится имя стека, адрес нижней границы стека, адрес верхней границы стека (или максимальное количество элементов в нем), указатель стека (номер последнего записанного элемента) и описание (тип) элемента стека.

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

Реализация стека массивом дает огромный выигрыш во времени, поскольку нет необходимости повторного выделения и освобождения памяти. Но количество элементов в стеке здесь ограничено, возможно переполнение стека;

Метод, которым будет реализован стек напрямую зависит от условий решаемой задачи – нужно знать примерное число элементов, которые могут храниться в стеке.