|  |  |
| --- | --- |
|  | **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**

Студент **Паламарчук А.Н.**

Группа **ИУ7-33Б**

Предмет **Типы и структуры данных**

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

|  |  |
| --- | --- |
| Студент | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Паламарчук А.Н.** |
| Преподаватель | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Никульшина Т. А.** |
| Преподаватель | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Барышникова М. Ю.** |

*2023 г.*

# Условие задачи

Создать программу работы для работы с типом данных «стек». Используя созданный тип данных, распечатать убывающие серии последовательности целых чисел в обратном порядке.

# Техническое задание

Реализовать стек: двумя способами – с помощью массива и списка. Оформить операции работы со стеком в виде подпрограмм. Ввести арифметическое выражение типа: число|знак| … число|знак| число. Вычислить значение выражения. (Приоритетность операций необязательна). Оценить преимущества и недостатки каждой реализации.

## Входные данные

* Команда (число) из меню программы (см. описание алгоритма) – обозначение необходимой операции.

## Выходные данные

* Таблица эффективности
* Исходные стеки
* Результат вычисления выражения

## Возможные аварийные ситуации

* Некорректный исходный файл
* Ошибка открытия файла
* Отсутствие исходного файла
* Нехватка памяти для работы программы
* Некорректная команда

## Описание внутренних структур данных

typedef struct node\_t node\_t;

struct node\_t

{

node\_t \*next;

int val;

};

int stack\_arr[MAX\_LEN\_ARR];

int \*ps = stack\_arr;

node\_t \*stack\_list = NULL;

node\_t \*pl = stack\_list;

**Константы**

#define MAX\_LEN\_ARR 10000

#define COUNT\_TESTS 100

**Используемые функции**

## unsigned long long cur\_mks\_gettimeofday(void);

Возвращает текущее время в мкс

int pop\_stack\_arr(int \*stack\_arr, int \*\*ps, int \*el)

Pop из стека (массив)

int push\_stack\_arr(int \*stack\_arr, int \*\*ps, int el)

Push в стек (массив)

int pop\_stack\_list(node\_t \*list, node\_t \*\*pl, int \*el)

Pop из стека (список)

int push\_stack\_list(node\_t \*\*list, node\_t \*\*pl, int el)

Push в стек (список)

int fscanf\_exp(char \*src, node\_t \*\*list, node\_t \*\*pl, int \*stack\_arr, int \*\*ps, size\_t print\_flag)

Считывает выражение из файла в стеки

int input\_exp(node\_t \*\*list, node\_t \*\*pl, int \*stack\_arr, int \*\*ps)

Ручное заполнение стеков

int print\_stack\_arr(int \*stack\_arr, int \*ps, int i)

Распечатать стек (массив)

int print\_stack\_list(node\_t \*stack\_list, node\_t \*pl, int i)

Распечатать стек (лист)

int calc\_exp\_arr(int \*stack\_arr, int \*ps, size\_t print\_flag)

Вычисляет значение выражения в стеке (массив)

int calc\_exp\_list(node\_t \*stack\_list, node\_t \*pl, size\_t print\_flag)

Вычисляет значение выражения в стеке (лист)

## void efficency\_table(void);

Вывод таблицы эффективности

## Описание алгоритма

Программа обрабатывает нужную команду:

Комманды:

1 - Заполнить стек используя данные из файла

2 - Заполнить стек вручную

3 - Вывести исходный стек

4 - Вычислить выражение (стек - статический массив)

5 - Вычислить выражение (стек - список)

6 - push (стек - статический массив)

7 - pop (стек - статический массив)

8 - push (стек - список)

9 - pop (стек - список)

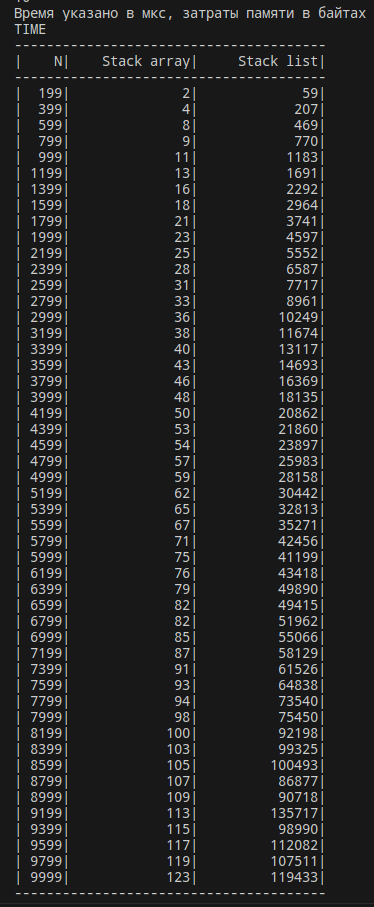
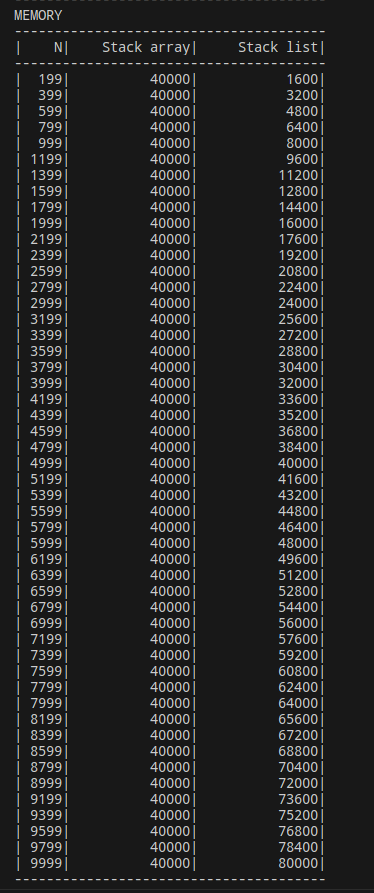
10 - Получения таблицы производительности

0 - Завершить работу программы

Записываем выражение в стек. Переписываем во временный стек и тем самым переворачиваем исходное выражение. Берём первый элемент записываем в результирующую переменную. Далее циклически идем по стеку и в соответствии с операцией вычисляем значение результирующего выражения.

**Производительность**

Замеры проводились по 100 раз.



# Фрагментация

Фрагментация редко наблюдается.

# Выводы по проделанной работе

Стек на основе связанного списка, стабильно проигрывает стеку на основе статического массива по скорости работы, однако при объёме элементов, меньшем, чем 50%, стек, выполненный в виде статического массива проигрывает стеку, выполненному в виде списка по памяти. Таким образом, можно сделать вывод, что если нужно реализовать стек, то нужно, ориентируясь на объём доступной памяти и ожидаемое количество элементов выбрать либо стек в виде массива – для прироста в скорости, либо для меньшего объёма занимаемой памяти при объёме элементов, больше чем 50%. Однако если вы рассчитываете обрабатывать меньшее количество элементов и вам критически важен объём занимаемой памяти, то можно выбрать стек на основе списка. Если же нам известно количество элементов в списке, то следует использовать статический массив для реализации стека, иначе следует использовать стек на основе списка.

# Контрольные вопросы

1. **Что такое стек?**

Стек – структура данных, работающая по правилу Last In First Out, в которой можно обрабатывать только последний добавленный элемент.

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

При хранении стека с помощью списка, мы выделяем блок динамической памяти для каждого узла. При хранении с помощью статического массива, память выделяется на стеке. Для каждого элемента стека, реализованного списком, выделяется значительно большее количество памяти, чем для очередного элемента массива. Эти дополнительные байты занимает указатель на следующий элемент списка. Размер указателя (4 или 8 байт) зависит от архитектуры.

Для стека на основе статического массива память выделяется до начала работы со стеком и сразу на все элементы стека, а для списка память выделяется по мере необходимости на каждый элемент стека.

1. **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. При удалении из стека, реализованного массивом, смещается лишь указатель на вершину стека, а сами данные на самом деле остаются в памяти, но мы считаем, что их там нет (затираем данные). Для стека на основе статического массива память освобождается после окончания работы со стеком, а для списка память очищается при удалении элемента из стека.

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

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

1. **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. Если же нам известно количество элементов в списке, то следует использовать статический массив для реализации стека, иначе следует использовать стек на основе списка.