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
«РАБОТА СО СТЕКОМ»
Вариант 6

Студент Кладницкий А. Б.

Преподаватель

Группа ИУ7 – 32Б

2022 г.

1. **Описание условия задачи**

   Реализовать операции работы со стеком, который представлен в виде динамического массива 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. Используя стек, перевести выражение в постфиксную форму.

2. **ТЗ**

   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

   1. Исходные данные:
      - Входная строка, содержащая выражение
   2. Результирующие данные:
      - Выражение в постфиксной форме
      - Затраченное время и память
      - Адреса элементов
   3. Задача программы:
      - Чтение строки-выражения
      - Обработка выражения
      - Логирование действий, связанных с динамической памятью
   4. Способ обращения к программе:
      - Запуск через терминал (./app.exe)
      - Опциональный ключ –s, с которым программа подавляет логирование и замеряет время и память
   5. Возможные ошибки:
      - Неверный ключ
      - Ошибка выделения динамической памяти
      - Переполнение стека
      - Ошибка чтения

3. Описание внутренних структур данных

Тип данных `stack_t` (стек на списке):

```
typedef struct stack
{
  node_t *top;
  size_t len;
  size_t len_max;
  void (*free_data)(void *);
} stack_t;
```

Тип содержит в себе указатель на «вершину» стека, текущий и максимальный размеры, а также указатель на функцию для освобождения файла из под данных (для очистки стека).

Тип данных `astack_t` (стек на массиве):

```
typedef struct astack
{
  int **btm;
  int **top;
  size_t len;
  size_t len_max;
} astack_t;
```

Тип `astack_t` содержит в себе указатель на «дно» и «вершину» стека, текущий и максимальный размеры.

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

Тип `node_t` хранит в себе значения `data` узла односвязного списка и ссылку на следующий узел (NULL, если конец). Значение поля `data` типа `void*`, поэтому узлы реализованы универсально.

4. **Алгоритм**
    1. Ввести строку
    2. Читать подстроки, разбивая строку по знакам
    3. Считанные подстроки-выражения сразу добавляются в результирующую строку, знаки обрабатываются по правилам и добавляются вначале в стек:
    - Если стек пуст, или если открывающая скобка лежит на его «вершине», или если это текущий знак, положить текущий знак в стек
    - Если текущий знак закрывающая скобка, удалять знаки из стека и дописывать в результат, пока не встретилась открывающая скобка. Удалить ее из стека
    - Если встречен знак с более высоким приоритетом, добавить его в стек
    - Если встречен знак с менее высоким приоритетом, удалять знаки из стека и дописывать их в результат, пока не встретится знак с менее высоким приоритетом. Добавить в стек текущий знак
    - Считанные подстроки-выражения сразу добавляются в результирующую строку, знаки обрабатываются по правилам и добавляются вначале в стек:
    4. После окончания обработки исходной строки переместить все элементы из стека в результирующую строку
5. **Набор тестов**

	Описание	Входные данные	Результат
1.	Некорректный ключ	-h, -q, -abc и т.д.	EXIT_BAD_KEY
2.	Невозможно открыть лог-файл	???	Сообщение о невозможности открытия лог-файла
3.	Ошибка чтения строки	-	EXIT_WRONG_READ
4.	Ошибка выделения динамической памяти	???	EXIT_ALLOCATE
5.	Переполнение стека	Выражение, занимающее в стеке более 64 позиций (знаки)	EXIT_OVERFLOW
6.	Числовое выражение	Числовое выражение	Переведенное в постфиксную форму числовое выражение, адреса памяти
7.	Буквенное выражение	Буквенное выражение	Переведенное в постфиксную форму буквенное выражение, адреса памяти
8.	Замеры времени и памяти	Строка-выражение, ключ –s	Выражение в постфиксной форме и результаты замеров

6. Оценка эффективности

Приведена эффективность реализации на массиве относительно реализации на списке.

Время (мкс):

Кол-во знаков	Список	Массив	Относит. эффективность
5	5	4	125%
10	8	6	133%
15	13	10	130%
25	30	17	176%
50	46	22	209%

Память (байт):

Кол-во знаков	Список	Массив	Относит. эффективность
5	80	56	143%
10	96	64	150%
15	128	80	160%
25	160	96	167%
50	176	104	169%

7. Вывод

Реализация стека на списке уступает реализации стека на динамическом массиве, как по памяти, так и по времени, для любых размерностей. Использование списка не выгодно.

8. Контрольные вопросы

1. Что такое стек?
   Структура данных типа LIFO (last in first out)
   По сути, представляет собой структуру-«стопку», в которой имеется доступ только к ее верхнему элементу.
2. Каким образом и сколько памяти выделяется под хранение стека при различной его реализации?
   2 реализации: список и массив
   Для массива выделяется память по размеру массива + память под хранение указателей и размеров.
   Для списка выделяется память отдельно под каждый его узел + под хранение размеров.
   Как правило, список требует дополнительно 8 байт для хранения каждого дополнительного элемента.
3. Каким образом освобождается память при удалении элемента стека при различной реализации стека?
   Для массива достаточно просто смещать указатель на «вершину» стека и менять размер.
   Для списка нужно освобождать память под «верхний» узел и смещать указатель на предыдущий узел
4. Что происходит с элементами стека при его просмотре?

Можно просматривать только верхний элемент стека. Чтобы получить доступ к следующему элементу, необходимо извлечь все предыдущие.

5. Каким образом эффективнее реализовывать стек? От чего это зависит?
   Стек на списке занимает больше памяти, чем на массиве, и в большинстве случаев программы с ним работают медленнее. Преимущество стека над статическим массивом в том, что его размер ограничен размером оперативной памяти (кучи), в то время как размер статического массива ограничен аппаратным стеком. Реализация на динамическом массиве выигрывает реализацию на списке в большинстве случаев.