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»

Студент Козырнов Александр Дмитриевич

Группа ИУ7 – 32Б

Преподаватель Силантьева А. В.

Вариант 6

2023 г.

Оглавление

Описание условия задачи

Разработать программу работы со стеком, реализующую операции добавления и удаления элементов из стека и отображения текущего состояния стека. Реализовать стек: а) массивом; б) списком.

Все стандартные операции со стеком должны быть оформлены отдельными подпрограммами. В случае реализации стека в виде списка при отображении текущего состояния стека предусмотреть возможность просмотра адресов элементов стека и создания дополнительного собственного списка свободных областей (адресов освобождаемой памяти при удалении элемента, который можно реализовать как списком, так и массивом) с выводом его на экран. Список свободных областей необходим для того, чтобы проследить, каким образом происходит выделение памяти менеджером памяти при запросах на нее и убедиться в возникновении или отсутствии фрагментации памяти.

Перевести выражение в постфиксную форму с учетом приоритета выполнения операций.

Описание технического задания

Входные данные:

Числовое значение номера пункта меню, цифры и выражения

Выходные данные:

Верхний элемент, результат постфиксная форма

Обращение к программе:

Запускается через терминал командой: ./app.exe.

Аварийные ситуации:

1. Неверный ввод пункта меню
2. Неверный ввод числа в стек
3. Неверно указано инфиксное выражение
4. Попытка вывести пустой стек/освобожденную область

НАБОР ТЕСТОВ

№	Название теста	Пользовательский ввод	Вывод
1	Корректно добавлено число	1 или 5 5	
2	Выражение из инфиксной формы превращено в префиксную верно	A + (B + C^D) + 1	A B C D ^ + + 1 +
3	Вывод пустого стека	13 или 14	Стек пуст
4	Ввод несуществующего пункта меню	ABC	Вы неверно указали пункт меню!
5	Некорректно добавлено число	1 или 5 abc	Вы неверно указали число!
6	Вывод пустой области освобожденных значений	4	Ничего нет

Описание структуры данных

```c
typedef struct node_t node_t;
struct node_t
{
    /* Узел
     *
     * prev - Указатель на предыдущий узел
     * */
    operation_t num;
    node_t *prev;
};

typedef struct listnode_t listnode_t;
struct listnode_t
{
    /* Узел
     *
     * prev - Указатель на предыдущий узел
     * */
    operation_t *num;
    listnode_t *prev;
};

typedef struct list_stack_t
{
    /*
     * Стак на листе
     *
     * index - Размер текущего стака
     * top - Указатель на вершину (первый) узел стака
     * */
    int index;
    node_t *top;
} list_stack_t;

typedef struct free_t
{
    /*
     * Область освобожденных значений
     *
     * Односвязный список освобожденных областей памяти для стака на листе
     * Указатель на конечный узел здесь потому, что добавление конец занимает константное время
     * */
    listnode_t *head; //next == prev
    listnode_t *tail;
} free_t;
```

```c
typedef struct array_stack_t
{
    /*
     * Стак на массиве
     *
     * beg - Указатель на начало стака
     * curr - Указатель на текущий элемент стака
     * end - Указатель на законец стака
     * */
    operation_t *beg, *curr, *end;
} array_stack_t;
```

Оценка эффективности

Таблица эффективности операций POP и PUSH.

Кол-во элементов	**Стек на листе, мс**	**Стек на массиве, мс**	**Отношение**
1	1.80	0.8	2.25
100	11.70	4.40	2.66
1000	82.5	26.7	3.09
10000	754.9	263.6	2.86
100000	4505.2	2009	2.24
100000000	3463399.7	989262.9	3.5

Для превращения из инфиксной в постфиксную

Стек на листе, мс	Стек на массиве, мс	Отношение
60494.5	43776.7	1.38

Можно заметить, что с увеличением количества элементов эффективность стека на массиве выигрывает в скорости больше, чем на меньших размерах. Это объясняется тем, что на листе память выделяется поэлементно (ищется для каждого элемента отдельное место), когда для массива все выделено в непрерывную область, отчего время, затраченное на эти операции, значительно уменьшено.

Также размер элемента массива равен 4 байта на информационную часть, 16 (12 без выравнивания) байт на 3 указателя, которые контролируют работу этой области памяти. Для листа мы используем узлы размером по 16 (8 без выравнивания) байт. То есть размер каждого элемента листа – это 16 байт, когда для массива – это 4 байта. Из этого следует, что массив в ~4 раза более эффективно использует память.

Описание алгоритма

1. Запрашивается пункт меню
2. Пользователь вводит пункт меню
3. В зависимости от пункта меню программа будет либо выводить данные, либо записывать их в стек, либо удалять их.

4. Если пользователь попросит превратить инфиксную запись в постфиксную, программа запросит ввести инфиксную запись. Далее с помощью стека она превращается в постфиксную. При возникновении ошибки, программа сообщит пользователю.

Ответы на контрольные вопросы

1. Что такое стек?

Стек – структура данных, работающая по принципу «последний пришёл – первый вышел». Это означает, что нам доступен только последний элемент стека. То есть мы не можем что-то достать по середине или в начале – только в конце.

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

При реализации стека массивом выделяется (n * 4 + 12) байта. Память выделяется одним блоком, а на последний элемент всегда есть указатель. При реализации стека листом в моей реализации узел занимает 16 байт, а сама структура занимает еще 16 байт. В итоге получаем (n + 1) * 16 байт размер структуры. Память выделяется на каждый узел отдельно. Для доступа к последнему элементу хранится указатель на этот узел.

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

При удалении элемента со стека на

1) Массиве: его указатель смещается на 1 элемент к началу. Если размер массива в два раза больше действительно заполненной памяти, мы изменяем его размер до действительно заполненного.

2) Списке: адрес удалённого элемента помещается в лист освобождённых адресов памяти и стирается (освобождается).

4. Что происходит с элементами стека при его просмотре?

Создаётся копия стека, куда помещаются все элементы. Элементы стека поочерёдно «вытаскиваются» из стека и выводятся на экран. Далее в стек возвращаются элементы в исходном порядке.

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

Стек на массиве всегда эффективнее как и по памяти, так и по скорости работы.

Вывод

Стек на массиве намного эффективнее стека на листе как по памяти, так и по скорости. Стек массивом занимает в 4 раза меньше места, а также работает в среднем в 2-3 раза быстрее. Это связано с тем, что на каждый элемент листа на стеке выделяется n раз, когда при моей реализации массива память выделяется только log2(n) раз. Как минимум из-за этого реализация на массиве выигрывает во времени. Такая же логика происходит и с очищением (удалением) всего стека.