Условие задачи:

Создать программу работы со стеком, выполняющую операции добавления, удаления элементов и вывод текущего состояния стека. Реализовать стек:
а) статическим массивом (дополнительно можно реализовать динамическим 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. Элементами стека являются слова. Распечатайте слова в обратном порядке, каждое слово в перевернутом виде.

Техническое задание:

1. Программа обрабатывает слова (под словами подразумеваются любая комбинация символов, без пробельных символов, длиной до 20 символов).
2. Максимальная длина стека – 1000 записей.
3. Операции со стеком производятся при вызове команд по их номерам.

Аварийные ситуации:

1. При вводе пустой строки программа будет заканчиваться с ошибкой – введена пустая строка.
2. При попытке удаления из пустого стека программа будет заканчиваться с ошибкой – невозможно удаление из пустого стека.
3. При попытке ввода неправилььного номера действия программа будет заканчиваться с ошибкой – неправильно введен номер действия.

Алгоритм:

1. Программа выводит краткую информацию по тому, как она работает.
2. Пользователю предлагается выбрать действие, которое будет совершено над стеком.
3. Программа выполняет команду и предлагает ввести новый номер команды.
4. Для завершения команды вводится 0.

Типы данных:

В программе используются 2 структуры:

```
MAX_LEN  = 1000
WORD_LEN  = 20


typedef struct
{
char words[MAX_LEN][WORD_LEN];
int num;
} mas;

typedef struct li
{
char word[WORD_LEN];
struct li *next;
} list;
```

Первая структура описывает форму записи стека в виде статического массива, вторая – в виде односвязного списка.

Тесты:

Входные данные	Выход программы	Код возврата
Удаление из пустого стека-массива	Ошибка! Нет элементов для удаления.	3
Удаление из пустого стека-списка	Ошибка! Нет элементов для удаления.	3
Попытка переполнения стека-массива	Ошибка! Переполнение стека.	5
Попытка переполнения стека-списка	Ошибка! Переполнение стека.	5
Попытка ввода пустого слова	Ошибка! Пустой ввод.	6
Попытка ввода некорректного слова	Ошибка! Неправильно введено слово.	1

Ошибка выделения памяти	Ошибка! Невозможно выделить память.	4
Попытка ввода неправильного номера действия	Ошибка! Неправильно введен номер действия.	2
Добавляем в стек-массив 4 записи, удаляем 2, выводим новый стек.	Первые два элемента, добавенные в стек в обратном порядке.	0
Добавляем в стек-список 6 записей, удаляем 2, выводим новый стек.	Первые 4 элемента введенных в стек, в обратном порядке	0
Добавляем в стек-список 5 записей, удаляем все записи, выводим массив адресов удаленных элементов.	5 адресов удаленных из стека элементов	0

Результаты замеров времени и памяти:

Замеряем заполнение стеков на 1000 элементов, вывод стеков на 1000 записей и удаление 1000 записей.

Производимое действие	Стек на массиве	Стек на списке
Удаление элемента	0.000451	0.000770
Добаление элемента	0.000833	0.001328
Вывод стека	0.002694	0.003523

Память, выделенная на стек - 32000 байт, на стек-массив 20000 байт.

Вывод:

Из результатов замерного эксперимента видно что работа со стеком на основе списка намного дольше, чем работа со стеком на основе статического массива, потому что нам постоянно приходится обращаться к элементу по его месту в памяти и копировать один элемент в другой. Очевидно, что если будет выделятся много лишней памяти под массив для стека, эта память не будет использоваться, но будет затрачена, так что, в смысле экономии затраченной

памяти, стек на списке лучше. Но в случае, если количество элементов в стеке известно заранее, мы можем выделить удобный нам по размерам массив, не выделяя лишнюю память.

Контрольные вопросы:

Что такое стек?

Стек это - абстрактный тип данных, представляющий собой список элементов, организованных по принципу last in — first out.

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

Если мы реализуем стек на статическом массиве, он занимает память этого массива, если стек реализуется на основе односвязного списка, он занимает память, зависящую от количества элементов в нем.

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

При удалении элемента стека из стека-массива мы просто меняем переменную отвечающую за количество элементов стека, при удалении же элемента из стека-списка мы очищаем целую структуру.

Что происходит с элементами стека при его просмотре?

Элементы стека при просмотре выводятся в обратном порядке, с конца в начало. При этом стек очищается.

Каким образом эффективнее реализовывать стек? От чего это зависит?

Если количество элементов заранее известно намного удобнее использовать стек-массив, потому что мы не будем задействовать лишнюю память и все действия будут происходить быстрее. Если же количество элементов не известно, удобнее использовать стек-список, чтобы не трогать лишнюю память.