*Воякин Алексей, ИУ7-34*

*Типы и структуры данных*

*Лабораторная работа 4*

*Работа со стеком*

*Вариант 7*

1. **Техническое задание**
   1. **Задача:**

Создать программу работы со стеком, выполняющую операции добавления, удаления элементов и вывод текущего состояния стека. Элементами стека являются слова.

Реализовать функцию печати слов в обратном порядке. 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. **Входные данные**

На входе программа получает вариант стека - 1 или 2 (массив или список), 3- информацию, а также номера операций - 1, 2, 3, 4 (добавление, удаление, вывод, печать слов в обратном порядке) - производимых над стеком.

* 1. **Выходные данные**

В результате программа выводит состояние стека (опция 3)

Также выводятся слова (элементы стека) в обратном порядке. При этом стек очищается.

* 1. **Аварийные ситуации**

В случае переполнения стека, или если пользователь пытается удалить элемент из пустого стека, программа сообщит пользователю об ошибке.

* 1. **Основание для разработки**

Учебный процесс

* 1. **Способ обращения программой**

Консоль

1. **Структура данных**

В программе было использовано 2 реализации стека - массив и список. Содержимое стека - адреса памяти (стека). Структура стека для массива содержит в себе массив элементов, указатель на вершину массива и максимальное количество элементов. Структура стека для списка содержит в себе значение элемента (слово), а также указатель на следующий элемент (next).

*Реализация массивом:*

**Typedef struct** {

**char \*\*string** - слово

**int top** - указатель на вершину массива

**int size** - максимальное количество элементов массива

} **Stack**

*Реализация списком:*

**struct node**

{

**char \*string** - слово

**struct node \*next** - также указатель на следующий элемент

};

1. **Алгоритм**

При создании стека через массив была создана структура с массивом, индексом начала массива и размером стека. При создании стека через список был реализован алгоритм работы со списком (у элемента списка есть адрес и ссылка на следующий элемент списка (next)). При реализации стека массивом максимальное количество элементов – 10;

1. **Тесты**

|  |  |  |
| --- | --- | --- |
| **Опция меню** | **Значение\случай** | **Сообщение** |
| **Array**  1. ПУШ | Добавление 11-ого элемента | Максимум достигнут |
| **Array/List**  2. ПОП | Попытка удалить элемент из пустого стека. | Стек пуст! |
| **Array/List**  4.Показать слова в обратном порядке | Пустой стек | Стек пуст! |
| **Array/List**  4. Показать слова в обратном порядке | <-| йц| ук| ен| гш| щз| фы|| | йц ук ен гш щз фы |
| **Array/List**  2.ПОП | <-| йц| ук| ен| гш| щз| фы|| | <-| ук| ен| гш| щз| фы|| |

1. **Заключение**

В результате работы были реализованы функции работы со стеком, а также были сделаны обе реализации стека - через массив и через список. Массив дает преимущество по времени, список - по памяти. Массив использовать целесообразно при известном количестве элементов.

При работе программы фрагментация памяти почти не происходит.

|  |  |  |
| --- | --- | --- |
|  | **push**  **(добавление)** | **pop**  **(удаление)** |
| **Стек массивом** | 0.005895 | 0.000067 |
| **Стек списком** | 0.000783 | 0.003776 |

1. **Ответы на вопросы**

**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).

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

Массив: 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. Список: До начала

работы указатель стека показывает на нулевой, физически отсутствующий адрес (т. е. указатель - пустой). При включении элемента в стек сначала происходит выделение области памяти, адрес которой записывается в указатель стека, а затем по значению этого указателя в стек помещается информация.

**3) Каким образом освобождается память при удалении элемента стека при различной**

**реализации стека?**

Массив: 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. Список: При исключении элемента 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.

**4) Что происходит с элементами стека при его просмотре?**

Классическая реализация стека предполагает, что просмотреть содержимое стека без извлечения (удаления) его элементов невозможно.

**5) Каким образом эффективнее реализовывать стек? От чего это зависит?**

Массив дает преимущество по времени и по памяти при почти полном его заполнении, иначе по памяти имеет преимущество список. Массив использоватьцелесообразно при известном количестве элементов стека.