Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Вятский государственный университет»

Колледж ВятГУ

**ОТЧЕТ**

**ПО домашней контрольной РАБОТЕ №6**

**«РЕАЛИЗАЦИЯ ЭЛЕМЕНТАРНЫХ СТРУКТУР ДАННЫХ НА ОСНОВЕ СТАТИЧЕСКОЙ ПАМЯТИ»**

**ПО ДИСЦИПЛИНЕ**

**«МДК 05.02 Разработка кода информационных систем»**

Выполнил: студент учебной группы

ИСПк-101-51-00

Пешнин Владислав Константинович

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2023

Цель работы: изучение принципов работы с базовыми структурами данных, получение навыков

организации case-меню.

Задание:

1. Написать программу для работы со структурой данных "Дек".

2. Структура данных должна быть реализована на основе статической памяти.

3. Работа со структурой должна осуществляться с помощью case-меню. Предусмотреть наглядную визуализацию содержимого структуры.

**Выполнение работы:**

1.Код Pascal:  
**uses** crt;

**const**

MAX\_SIZE = 100;

**type**

Deck = **record**

data: **array**[0..MAX\_SIZE] **of** Integer;

size: Integer;

**end**;

**procedure** Init(**var** deck: Deck);

**begin**

deck.size := 0;

**end**;

**procedure** Push(**var** deck: Deck; element: Integer);

**begin**

**if** deck.size < MAX\_SIZE **then**

**begin**

deck.data[deck.size] := element;

inc(deck.size);

**end**

**else**

print('Ошибка: ДЕК заполнен!');

**end**;

**procedure** InsertFirst(**var** deck: Deck; element: Integer);

**begin**

**if** deck.size = MAX\_SIZE **then**

println('Дек переполнен!')

**else**

**begin**

**for var** i := deck.size - 1 **downto** 0 **do**

deck.data[i + 1] := deck.data[i];

deck.data[0] := element;

inc(deck.size)

**end**;

**end**;

**procedure** Pop(**var** deck: Deck);

**begin**

**if** deck.size > 0 **then**

dec(deck.size)

**else**

println('Ошибка: ДЕК пустой!');

**end**;

**procedure** DeleteFirst(**var** deck: Deck);

**begin**

**if** deck.size = 0 **then**

println('Дек пуст!')

**else**

**begin**

**for var** i := 0 **to** deck.size - 1 **do**

deck.data[i] := deck.data[i + 1];

deck.size := deck.size - 1;

**end**;

**end**;

**procedure** Clear(**var** deck: Deck);

**begin**

deck.size := 0;

println('Дек очищен')

**end**;

**procedure** Print(**const** deck: Deck);

**begin**

**if** deck.size <> 0 **then begin**

print('ДЕК: ');

**for var** i := 0 **to** deck.size - 1 **do**

print(deck.data[i], ' ');

println

**end**;

**if** deck.size = 0 **then** println('ДЕК пуст!')

**end**;

**var**

d: Deck;

choice: integer;

**begin**

Init(d);

**repeat**

println();

println('1. Добавить элемент в конец');

println('2. Добавить элемент в начало');

println('3. Удалить последний элемент');

println('4. Удалить первый элемент');

println('5. Вывести ДЕК');

println('6. Очистить ДЕК');

println('7. Выход');

choice := ReadInteger('Ваш выбор:');

**case** choice **of**

1:

**begin**

**var** element := ReadlnInteger('Введите элемент для добавления: ');

Push(d, element);

**end**;

2:

**begin**

**var** element := ReadlnInteger('Введите элемент для добавления: ');

InsertFirst(d, element);

**end**;

3: Pop(d);

4: DeleteFirst(d);

5: Print(d);

6: Clear(d);

**end**;

**until** choice = 7;

**end**.

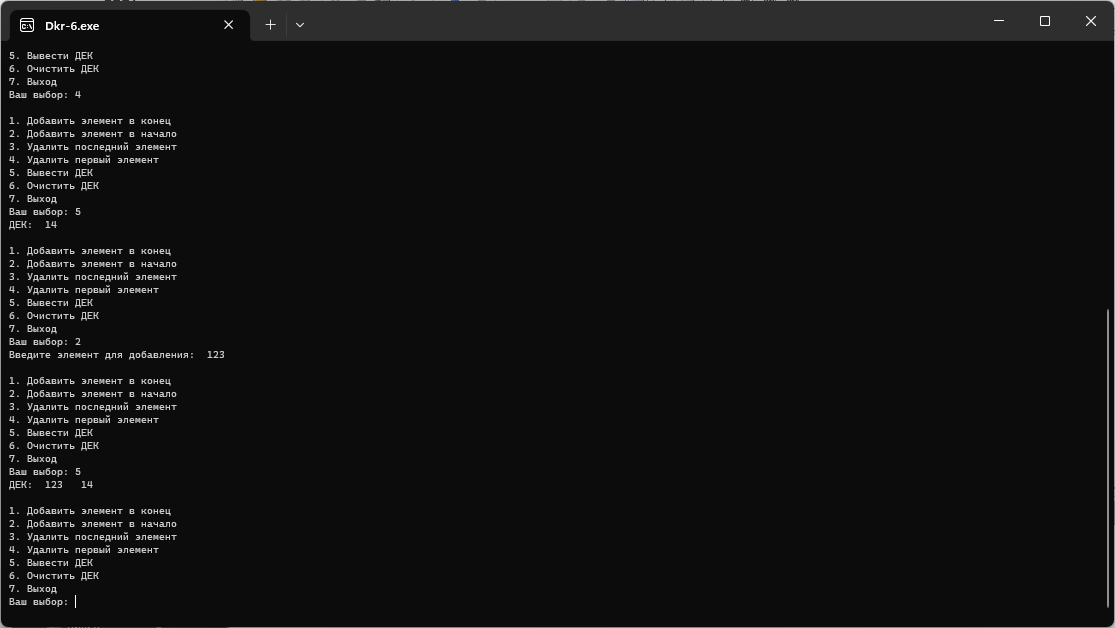


Рисунок 1 – Результат выполнения программы

2. Описание программы

Структура дека (Stack) позволяет хранить данные в порядке последовательности от последнего добавленного элемента к первому до последнего. Доступные операции: добавление элемента в конец стека (push), извлечение элемента с вершины стека (pop).

Для реализации стека с использованием статичной памяти необходимо создать массив для хранения данных и указатель указывающий на вершину стека.

Push и pop осуществляются так:

Push: проверяем указатель вершины стека, если он указывает на последний элемент массива, то сообщаем, что стек переполнен, иначе увеличиваем указатель вершины стека, и вставляем новый элемент в массив.

Pop: проверяем указатель вершины стека, если он указывает на начальный элемент массива, то сообщаем, что стек пустой, иначе возвращаем значение с вершины стека и уменьшаем указатель вершины стека.

3. Вывод

Вывод: В ходе практической работы усвоили различные структуры. Основная характеристика дека заключается в том, что добавление и удаление элементов происходит с одной из двух вершин стека (обычно с верхней), а не из середины. Такая структура может использоваться для решения многих задач программирования, а именно: хранение и переходы между различными состояниями приложения; обработка данных в порядке добавления; использование для реализации алгоритмов обхода графа; и использование для реализации стратегии поиска, такой как поиск в глубину.