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 «МДК 05.02. Разработка кода информационных систем»**

Выполнил: студент учебной группы

ИСПк-202-52-00

Караваев Александр Сергеевич

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2024

**Цель работы:** изучение принципов работы с базовыми структурами данных, получение навыков организации case-меню.

Вариант 10.

**Задание**:

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

Описание алгоритма.

1. Объявляем тип записи “Node” для хранения целочисленных данных и ссылки на следующий узел, а также определяем запись стека с ссылкой на верхний узел.
2. Реализуем функции для работы со стеком:

* “InitializeStack”: устанавливает вершину стека в “nil”.
* “IsEmpty”: проверяет, пуст ли стек, сравнив вершину со значением “nil”.
* “IsFull”: всегда возвращает “False” для стека с динамической памятью.
* “Push”: выделяет новый узел, устанавливает его данные в заданное значение и добавляет узел на вершину стека.
* “Pop”: удаляет верхний узел из стека и возвращает его данные, освободив память, занятую узлом.
* “DisplayStack”: проходится по стеку и выводит его содержимое.

1. Задаём переменные для динамического и статического стеков, для определения типа памяти и переменную для отслеживания вершины статического стека.
2. Инициализируем динамический стек и устанавливаем использование динамической памяти по умолчанию.
3. Создать цикл с меню для выполнения операций со стеком на основе ввода пользователя:

* Позволить пользователю переключаться между динамической и статической памятью.
* Добавить элемент в стек, учитывая выбранный тип памяти.
* Удалить элемент из стека, снова зависит от выбранного типа памяти.
* Отобразить содержимое стека.
* Выйти из программы.

Структура данных “Стек”:

1. Стек — это абстрактная структура данных, которая работает по принципу LIFO (Last In, First Out), что означает, что последний добавленный элемент в стек будет первым, который можно удалить.

Свойства стека:

* Элементы добавляются и удаляются с конца стека, при этом последний добавленный элемент будет первым, который можно удалить.
* Стек имеет фиксированный или ограниченный размер, что определяется при его создании.

1. Структура стека хранится в массиве “StaticStack”. Узлы в этом случае представлены значениями массива, а связи между элементами неявны и обеспечиваются позициями элементов в массиве.
2. Основные операции со структурой:

* Добавление элемента (Push): Создается новый узел, в который помещается переданное значение. Затем новый узел становится вершиной стека.
* Удаление элемента (Pop): Возвращает и удаляет значение верхнего узла из стека.
* Вывод на экран (DisplayStack): Отображает содержимое стека, проходя по всем узлам и выводя их значения.

1. Организация кейс-меню:

В программе реализовано меню операций. Пользователю предлагается выбрать действие (изменить тип памяти, добавить элемент, удалить элемент, вывести содержимое, выйти). В зависимости от выбора пользователя выполняется соответствующая операция с использованием соответствующего типа памяти (динамической или статической).

Схема алгоритма:

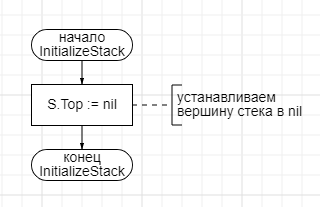


Рис.1 – функция инициализации стека

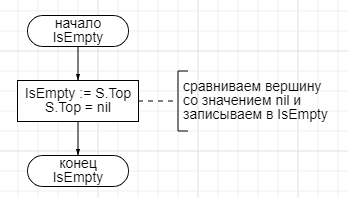


Рис.2 – функция проверки стека на полноту

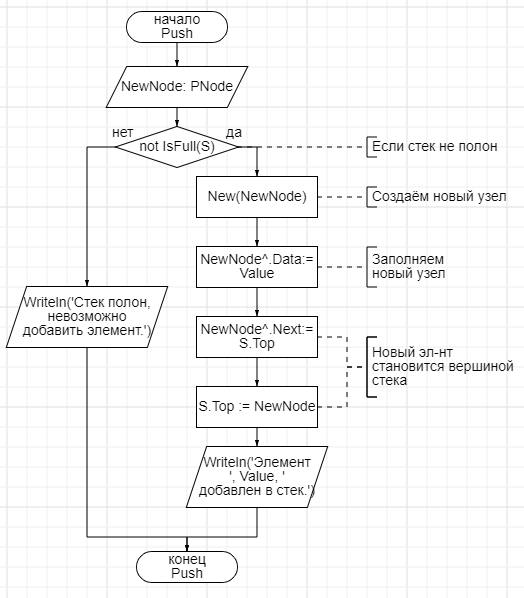


Рис.3 – процедура добавление элемента в стек

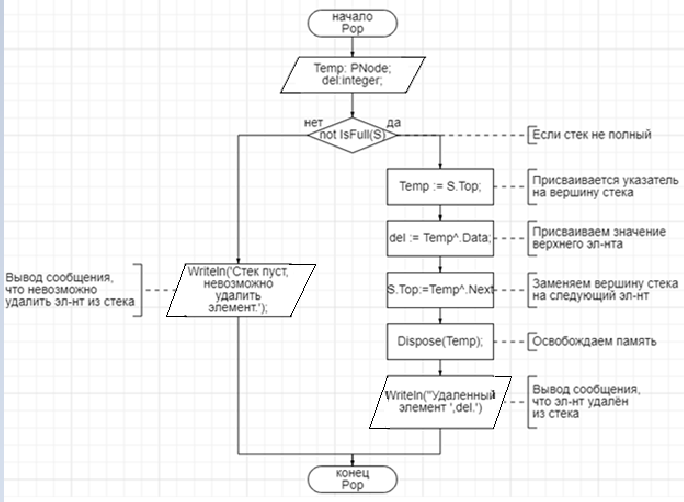


Рис.4 – функция удаления элемента из стека

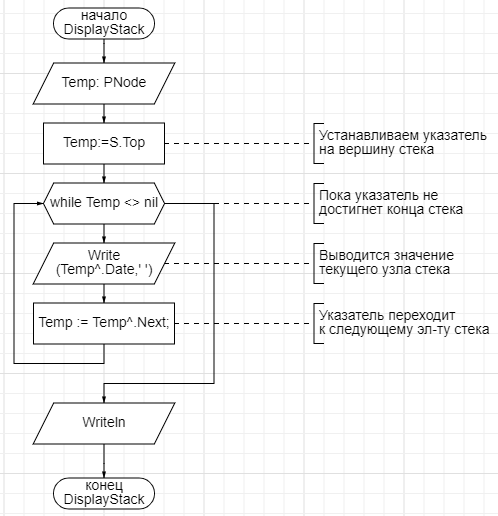


Рис.5 – Процедура отображения содержимого стека

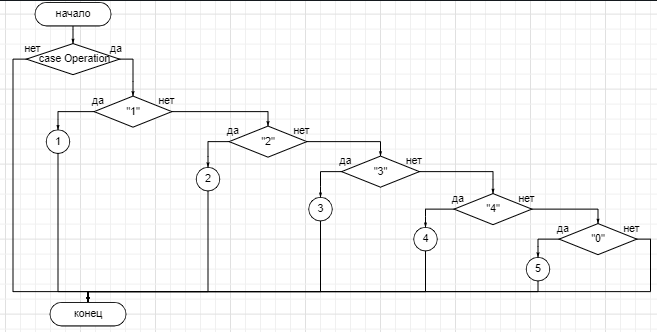


Рис.6 – реализация сase-меню

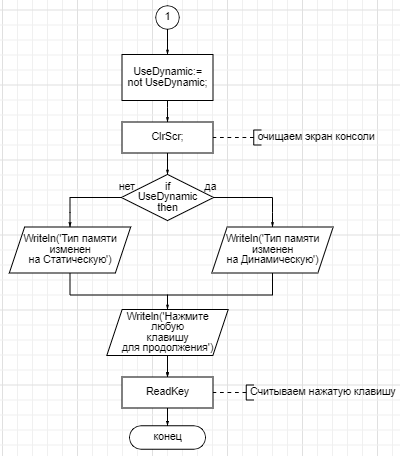


Рис.7 – смена памяти стека при нажатии “1”

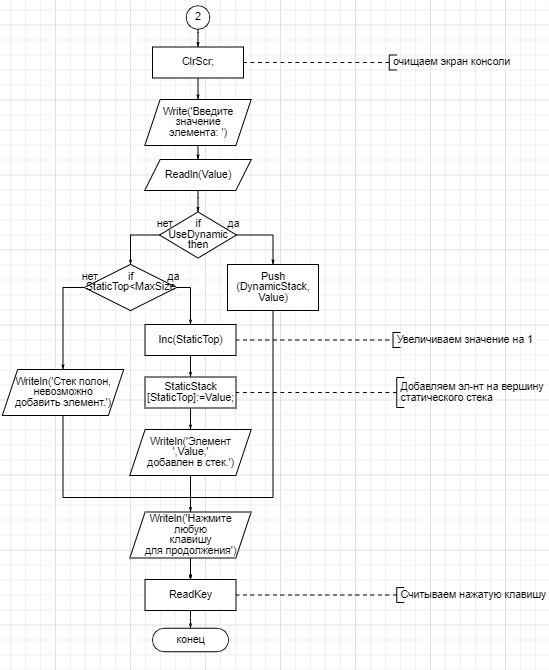


Рис.8 – добавление элемента при нажатии “2”

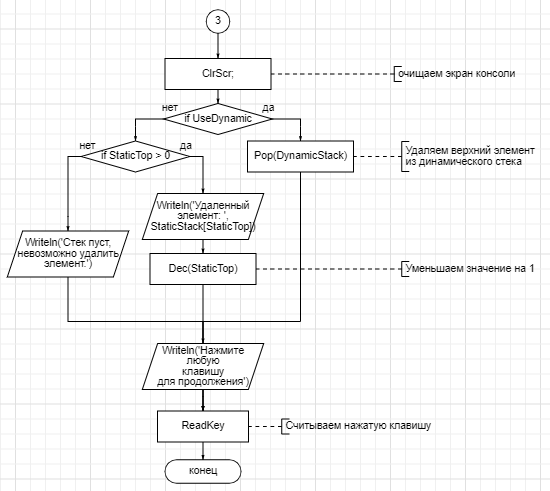


Рис.9 – удаление элемента при нажатии “3”

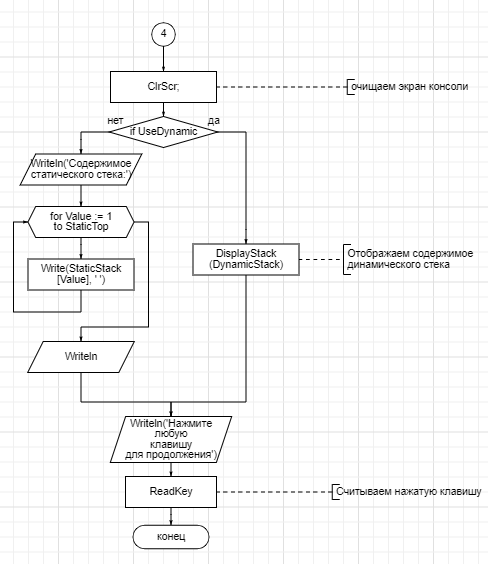


Рис.10 – отображение содержимого стеков при нажатии “4”

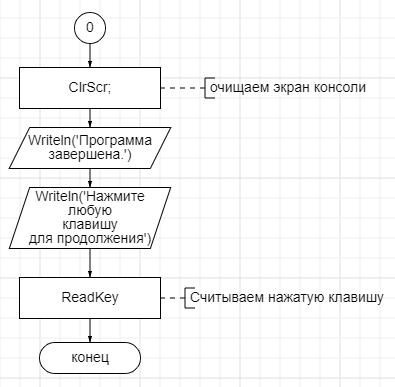


Рис.11 – выход из программы при нажатии “0”

**Вывод**

Таким образом, данная программа на Pascal позволяет визуализировать

фрактал “Кривая Дракона”.

Для построения множества ломанных, образующих фрактал, был создан отдельный модуль. Были предусмотрены возможности масштабирования, изменения глубины и перемещения фигуры.

В процессе работы были изучены основные принципы работы рекурсии и были отработаны навыки реализации рекурсивных алгоритмов для решения различных задач.

При выполнении работы проблем не возникло.