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-203-52-00

Обухов Илья Николаевич

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2024

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

**Задание:**

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

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

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

**Описание алгоритма:**

Стек - динамическая структура данных, в которой добавление и удаление элементов доступно только с одного конца (с верхнего (последнего) элемента). Эта­ структура данных, которая следует принципу “последний вошел, первый вышел” (LIFO - Last In, First Out). Это означает, что последний элемент, добавленный в стек, будет первым, который будет удален.

**Описание работы кода:**

1. **Определение стека**: Cтек представлен структурой TStack, которая содержит массив Xran для хранения элементов стека и переменную Verx для отслеживания вершины стека.
2. **Инициализация стека**: Процедура Ystanovka устанавливает Verx в 0, указывая на то, что стек пуст.
3. **Проверка на пустоту**: Функция Pysto возвращает True, если стек пуст (то есть Verx равно 0).
4. **Проверка на заполненность**: Функция Fuel возвращает True, если стек полон (то есть Verx равно MAX\_SIZE).
5. **Добавление элемента**: Процедура Push добавляет элемент на вершину стека, если стек не полон. Если стек полон, выводится сообщение “Стек полон”.
6. **Удаление элемента**: Функция Pop удаляет элемент с вершины стека и возвращает его, если стек не пуст. Если стек пуст, выводится сообщение “Стек пуст” и возвращается -1.
7. **Отображение стека**: Процедура PrintStack выводит все элементы стека, начиная с первого добавленного элемента, если стек не пуст. Если стек пуст, выводится сообщение “Стек пуст”.
8. **Основной цикл**: В основном цикле пользователю предлагается выбор действия: добавить элемент в стек, удалить элемент из стека, показать содержимое стека или выйти из программы. Выбор пользователя обрабатывается с помощью оператора case.

**Схемы алгоритмов:**

**Изображение выглядит как снимок экрана, текст, диаграмма, круг

Автоматически созданное описание**

Рисунок 1 – Процедура Pop

Изображение выглядит как снимок экрана, текст, диаграмма, круг

Автоматически созданное описание

Рисунок 2 – Процедура Push

Изображение выглядит как снимок экрана, текст, диаграмма, дизайн

Автоматически созданное описание

Рисунок 3 – Процедура PrintStack

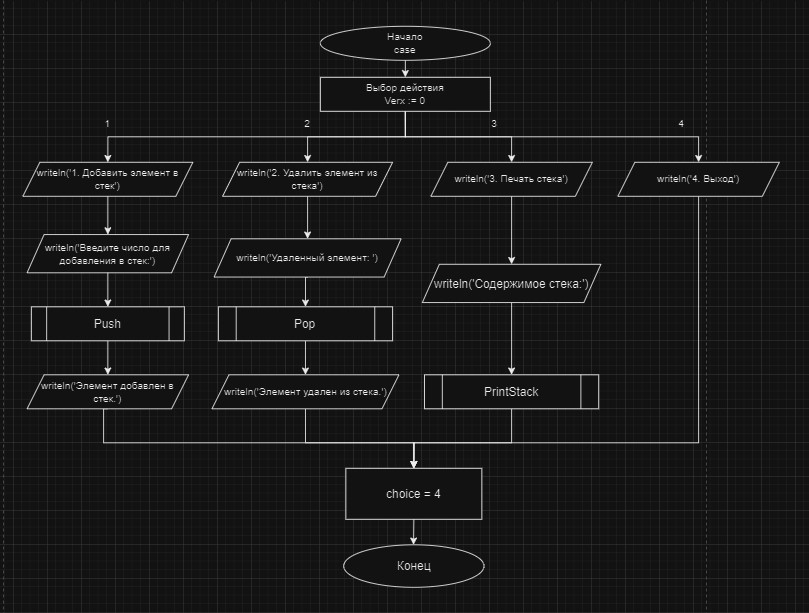


Рисунок 4 – Case-меню

**Код программы:**

**program** StackILYXA;

**const**

MAX\_SIZE = 10;

**type**

TStack = **record**

Xran: **array**[1..MAX\_SIZE] **of** Integer;

Verx: Integer;

**end**;

**procedure** Ystanovka(**var** S: TStack);

**begin**

S.Verx := 0;

**end**;

**function** Pysto(**const** S: TStack): Boolean;

**begin**

Pysto := S.Verx = 0;

**end**;

**function** Fuel(**const** S: TStack): Boolean;

**begin**

Fuel := S.Verx = MAX\_SIZE;

**end**;

**procedure** Push(**var** S: TStack; Value: Integer); //функция добавления числа в вершину стека

**begin**

**if not** Fuel(S) **then**

**begin**

Inc(S.Verx); //увеличивает значение переменной x на 1

S.Xran[S.Verx] := Value;

**end**

**else** //Если стек уже полон, то выводится сообщение

writeln('Стек полон');

**end**;

**function** Pop(**var** S: TStack): Integer; //функция удаления числа из стека

**begin**

**if not** Pysto(S) **then**

**begin**

Pop := S.Xran[S.Verx];

Dec(S.Verx);

**end**

**else** //Если стек пуст, выводится сообщение и возвращается -1

**begin**

writeln('Стек пуст');

Pop := -1;

**end**;

**end**;

**procedure** PrintStack(**const** S: TStack); //вывод всех элементов стека

**var**

i: Integer;

**begin**

**if not** Pysto(S) **then**

**begin**

writeln('Содержимое стека:');

**for** i := 1 **to** S.Verx **do**

writeln(S.Xran[i]);

**end**

**else**

writeln('Стек пуст');

**end**;

**var**

MyStack: TStack;

choice, Value: Integer;

**begin**

Ystanovka(MyStack);

**repeat**

writeln('1. Добавить элемент в стек');

writeln('2. Удалить элемент из стека');

writeln('3. Печать стека');

writeln('4. Выход');

readln(choice);

**case** choice **of**

1: **begin**

writeln('Введите значение для добавления в стек: ');

readln(Value);

Push(MyStack, Value);

writeln('Элемент добавлен в стек.');

**end**;

2: **begin**

writeln('Удаленный элемент: ', Pop(MyStack));

writeln('Элемент удален из стека.');

**end**;

3: **begin**

Printer(MyStack);

**end**;

**end**

**until** choice = 4;

**end**.

**Результат выполнения программы:**

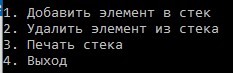


Рисунок 5 – Меню

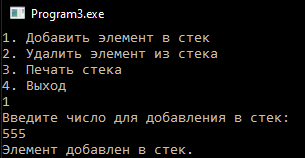


Рисунок 6 - Добавление элемента в стек

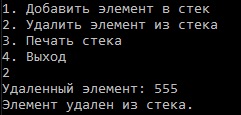


Рисунок 7 - Удаление элемента из стека

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 8 – Печать стека

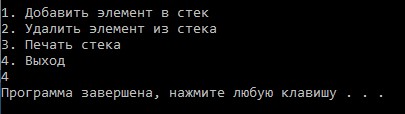


Рисунок 9 – Выход из программы

**Вывод**

В ходе выполнение домашней контрольной работы №6 мы узнали много нового и интересного, изучили принципы работы с базовыми структурами данных, и получили навыки организации case-меню. Так же написали программу для работы со структурой данных “Стек”.

Были некоторые проблемы, но мы быстро находили решение, нужно было всего получше разобраться, и проблемы пропадали. Еще в эту домашнюю контрольную работу я решил добавить частику свой души, это также заняло некоторое время, но результатом я доволен.