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-203-52-00

Вальковская Арина Константиновна

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2024

Содержание

1. Цель работы
2. Формулировка задания (с вариантом)
3. Схема алгоритма с комментариями
4. Код программы
5. Результат выполнения программы
6. Вывод

**1.Цель работы:**

Изучение принципов работы с базовыми структурами данных, получение навыков организации case-меню.

**2. Формулировка задания (Вариант:4)**

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.

**3. Схема алгоритма с комментариями**

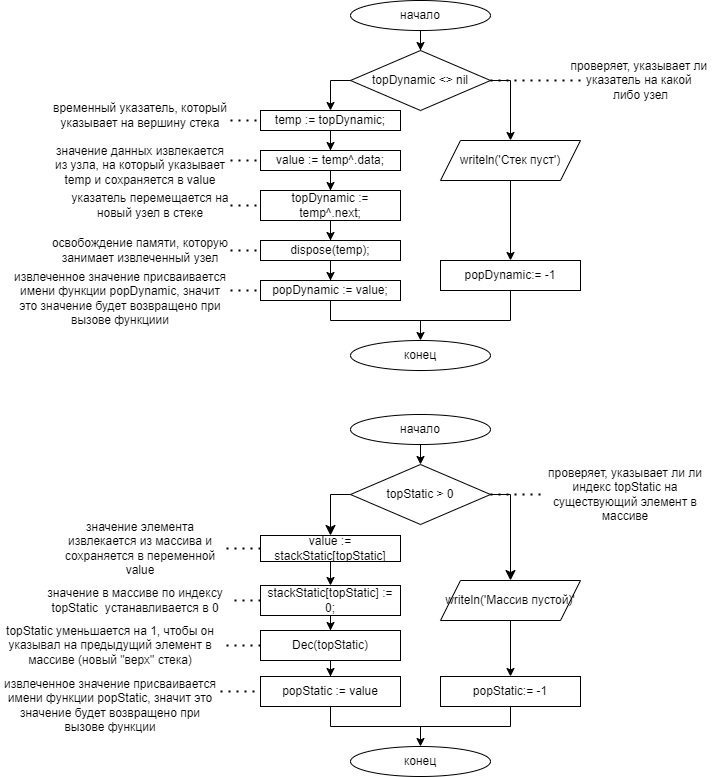


Рис. 1 – функция popDynamic

Если стек не пуст, функция извлекает элемент с вершины стека, освобождает память, занимаемую этим элементом, и возвращает его значение.

Если стек пуст, функция выводит сообщение об ошибке и возвращает значение -1.

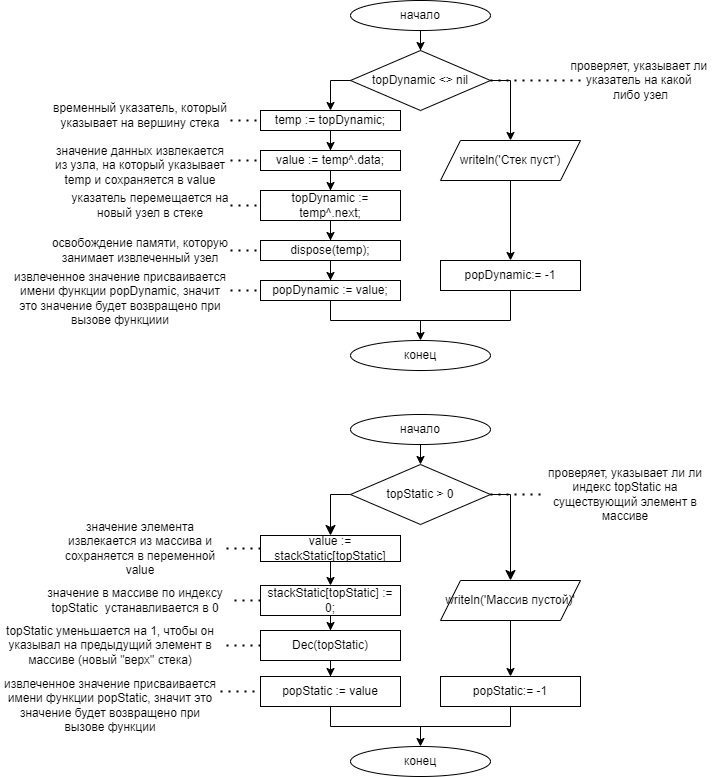


Рис. 2 – функция popStatic

Если стек не пуст, функция извлекает элемент с вершины стека, обнуляет значение в массиве, обновляет индекс вершины стека и возвращает извлеченное значение.

Если стек пуст, функция выводит сообщение об ошибке и возвращает значение -1.

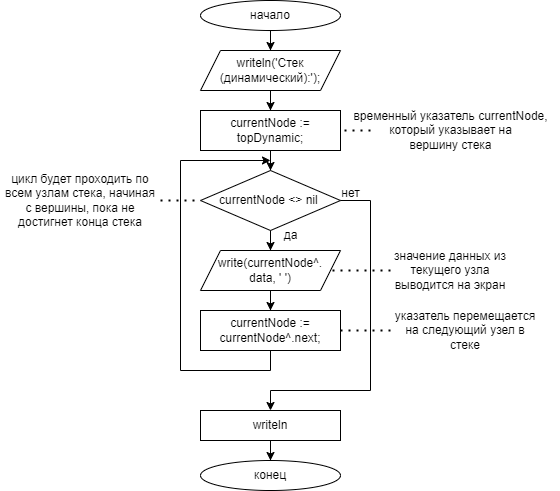


Рис. 3 – процедура printStackDynamic

Процедура проходит по всем узлам динамического стека, начиная с вершины, и выводит значение данных из каждого узла на экран.

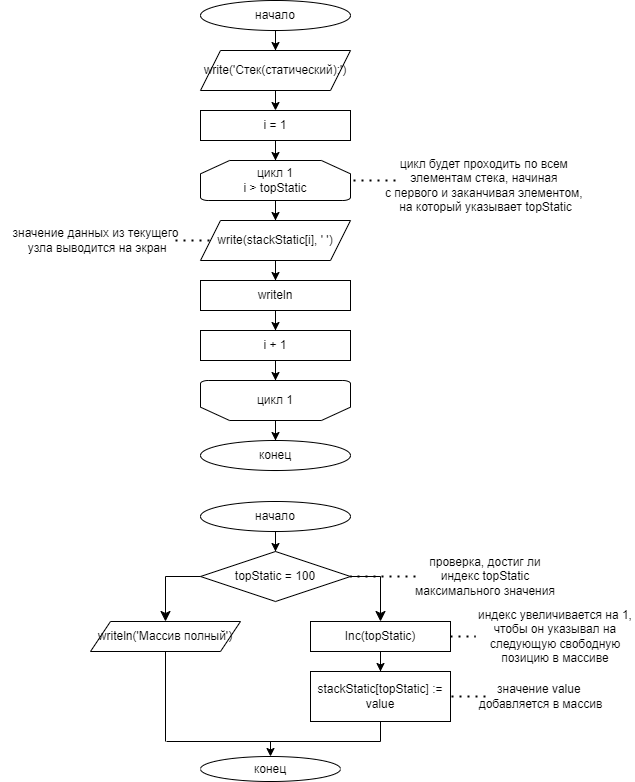


Рис. 4 – процедура printStackStatic

Процедура проходит по всем элементам статического стека, начиная с первого элемента, и выводит значение каждого элемента на экран.

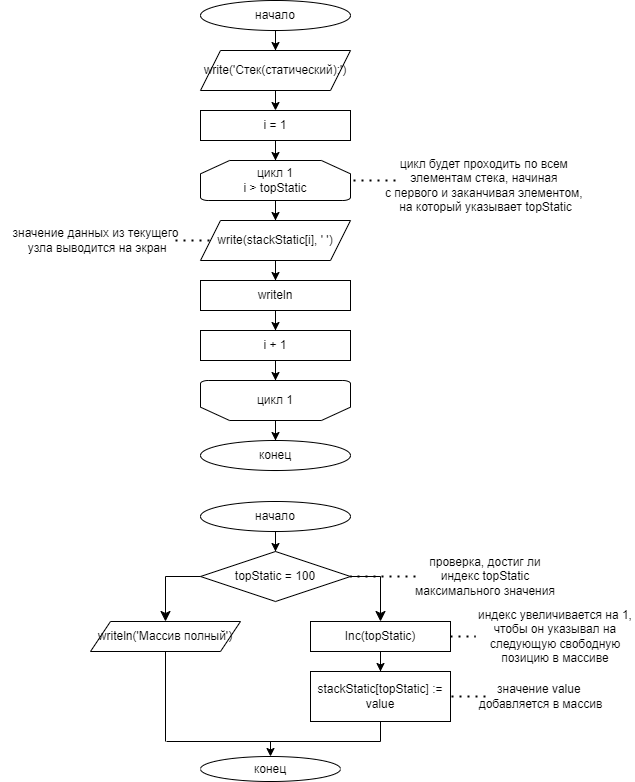


Рис. 5 – процедура pushStatic

Если в стеке есть место, процедура увеличивает индекс вершины стека и добавляет новый элемент value в массив по этому индексу.

Если стек полон, процедура выводит сообщение об ошибке и не добавляет новый элемент.

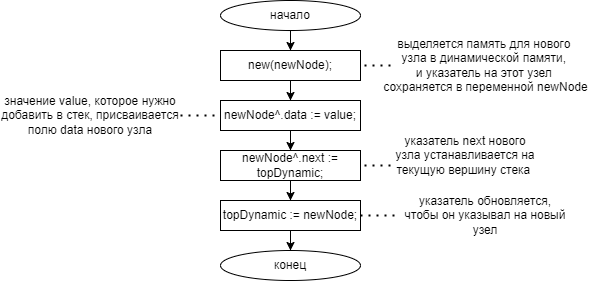


Рис. 6 – процедура pushDynamic

Процедура создает новый узел, заполняет его данными (value), связывает его с предыдущей вершиной стека и делает новый узел новой вершиной стека.

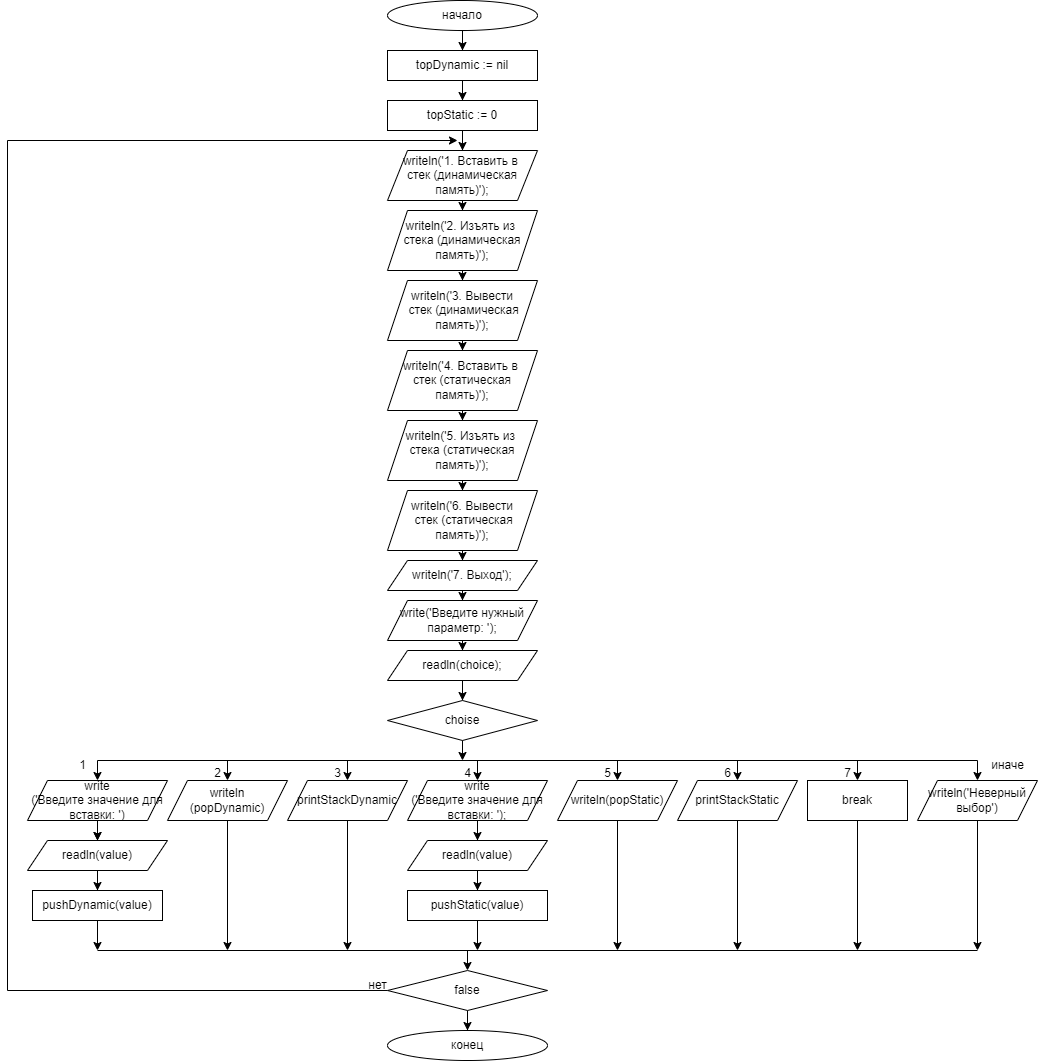


Рис. 7 – основной блок

**4. Код программы**

**program** dkr\_6;

**uses** crt;

**type**

StackNode = ^Node;

Node = **record**

data: integer;

next: StackNode;

**end**;

**var**

topDynamic: StackNode;

topStatic: integer;

stackStatic: **array**[1..100] **of** integer;

**procedure** pushDynamic(value: integer);

**var**

newNode: StackNode;

**begin**

**new**(newNode);

newNode^.data := value;

newNode^.next := topDynamic;

topDynamic := newNode;

**end**;

**function** popDynamic: integer;

**var**

temp: StackNode;

value: integer;

**begin**

**if** topDynamic <> nil **then**

**begin**

temp := topDynamic;

value := temp^.data;

topDynamic := temp^.next;

dispose(temp);

popDynamic := value;

**end**

**else**

**begin**

writeln('Стек пуст');

popDynamic := -1;

**end**;

**end**;

**procedure** printStackDynamic;

**var**

currentNode: StackNode;

**begin**

writeln('Стек (динамический):');

currentNode := topDynamic;

**while** currentNode <> nil **do**

**begin**

write(currentNode^.data, ' ');

currentNode := currentNode^.next;

**end**;

writeln;

**end**;

**procedure** pushStatic(value: integer);

**begin**

**if** topStatic = 100 **then**

writeln('Массив полный')

**else**

**begin**

Inc(topStatic);

stackStatic[topStatic] := value;

**end**;

**end**;

**function** popStatic: integer;

**var**

value: integer;

**begin**

**if** topStatic > 0 **then**

**begin**

value := stackStatic[topStatic];

stackStatic[topStatic] := 0;

Dec(topStatic);

popStatic := value;

**end**

**else**

**begin**

writeln('Массив пустой');

popStatic := -1;

**end**;

**end**;

**procedure** printStackStatic;

**var**

i: integer;

**begin**

writeln('Стек (статический):');

**for** i := 1 **to** topStatic **do**

write(stackStatic[i], ' ');

writeln;

**end**;

**var**

choice, value: integer;

**begin**

topDynamic := nil;

topStatic := 0;

**repeat**

writeln('1. Вставить в стек (динамическая память)');

writeln('2. Изъять из стека (динамическая память)');

writeln('3. Вывести стек (динамическая память)');

writeln('4. Вставить в стек (статическая память)');

writeln('5. Изъять из стека (статическая память)');

writeln('6. Вывести стек (статическая память)');

writeln('7. Выход');

write('Введите нужный параметр: ');

readln(choice);

**case** choice **of**

1:

**begin**

write('Введите значение для вставки: ');

readln(value);

pushDynamic(value);

**end**;

2:

writeln(popDynamic);

3:

printStackDynamic;

4:

**begin**

write('Введите значение для вставки: ');

readln(value);

pushStatic(value);

**end**;

5:

writeln(popStatic);

6:

printStackStatic;

7:

**break**;

**else**

writeln('Неверный выбор');

**end**;

**until** false;

**end**.

**5. Результаты выполнения программы**

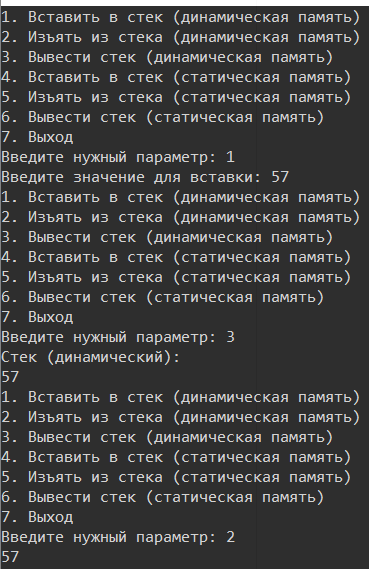
****

Рис. 8 – вставить в стек, вывести стек и изъять из стека (динамическая память)

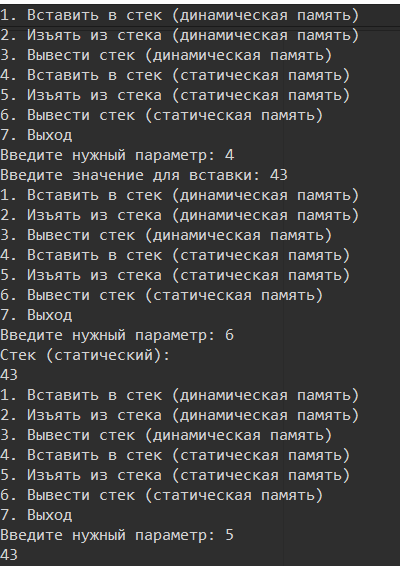


Рис. 9 – вставить в стек, вывести стек и изъять из стека (статическая память)

**6. Вывод**

В ходе выполнения лабораторной работы по реализации элементарных структур данных на основе статической и динамической памяти были изучены принципы работы с различными типами памяти. Статическая память выделяется на этапе компиляции программы и имеет фиксированный размер, что обеспечивает быстрый доступ к данным, но ограничивает их количество и гибкость. Динамическая память выделяется во время выполнения программы и позволяет эффективно управлять ресурсами, однако требует более сложной работы с указателями и управлением памятью.

В процессе выполнения отчета, для создания блок схемы мы работали с программой draw.io. Draw.io — это бесплатный онлайн-сервис, который помогает создавать алгоритмы, прототипы, инфографику и диаграммы любого вида. Чаще всего его используют именно для построения диаграмм. Сервисом пользуются разные IT-специалисты — от бизнес-аналитиков до разработчиков и дизайнеров.