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-101-51-00

Бекмансуров Рустам Мхадтисович

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2024

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

**Формулировка задания:**

**Вариант 22**

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.

**Описание алгоритма:** Задание реализовано на основе статической и динамической памяти. Для облегчения чтения кода, реализации были вынесены в отдельные модули. Также реализован их выбор при помощи case-меню.

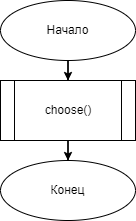


Рисунок 1 – алгоритм программы

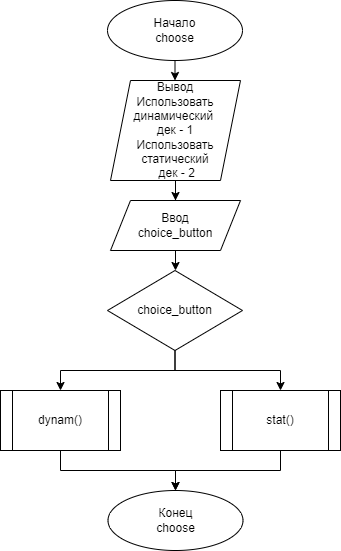


Рисунок 2 – алгоритм процедуры choose

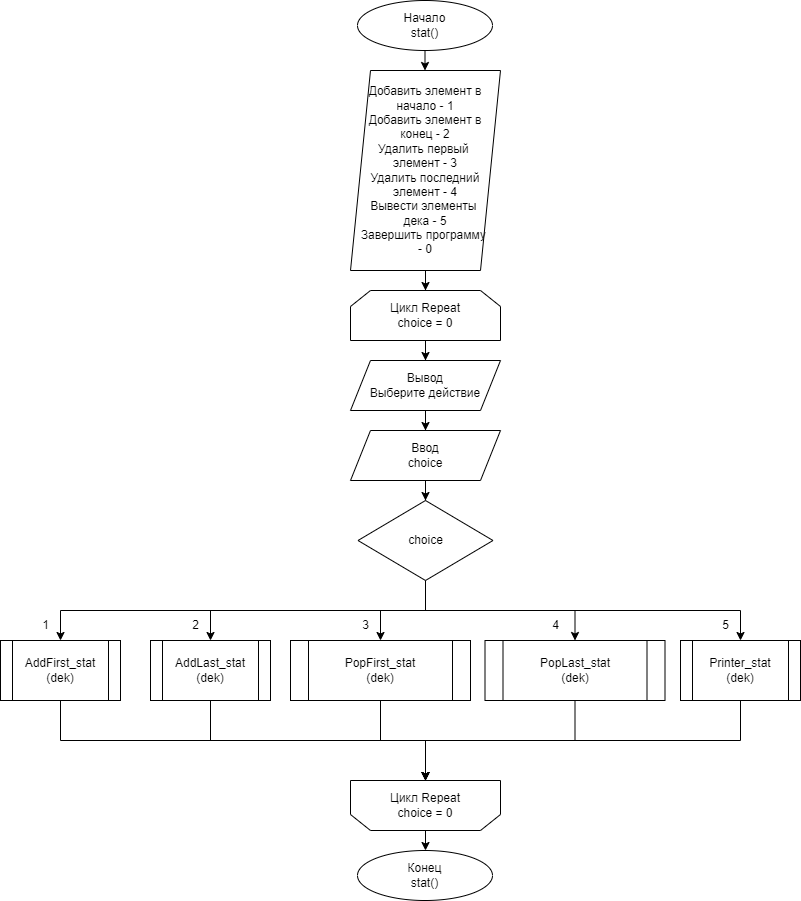


Рисунок 3 – алгоритм процедуры stat

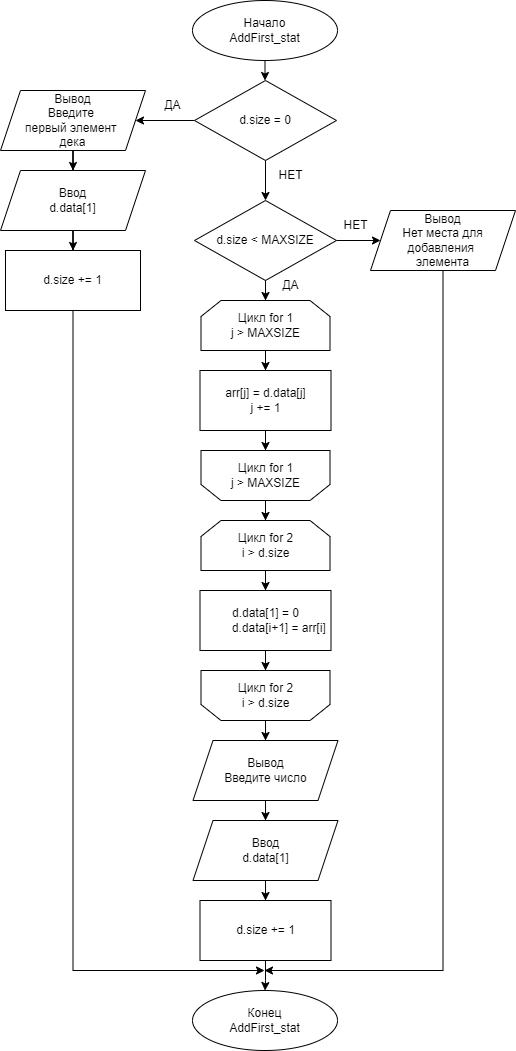


Рисунок 4 – алгоритм процедуры AddFirst\_stat

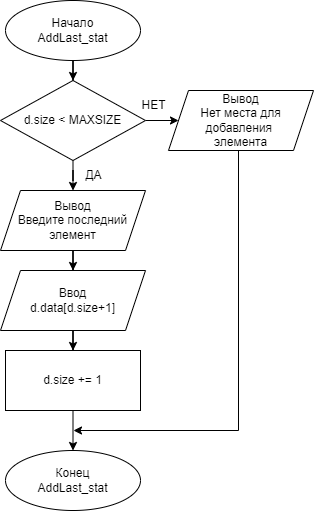


Рисунок 5 – алгоритм процедуры AddLast\_stat

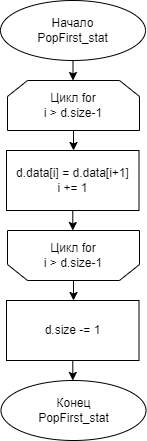


Рисунок 6 – алгоритм процедуры PopFirst\_stat

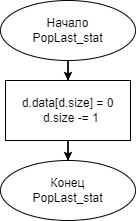


Рисунок 7 – алгоритм процедуры PopLast\_stat

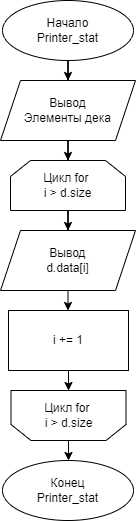


Рисунок 8 – алгоритм процедуры Printer\_stat

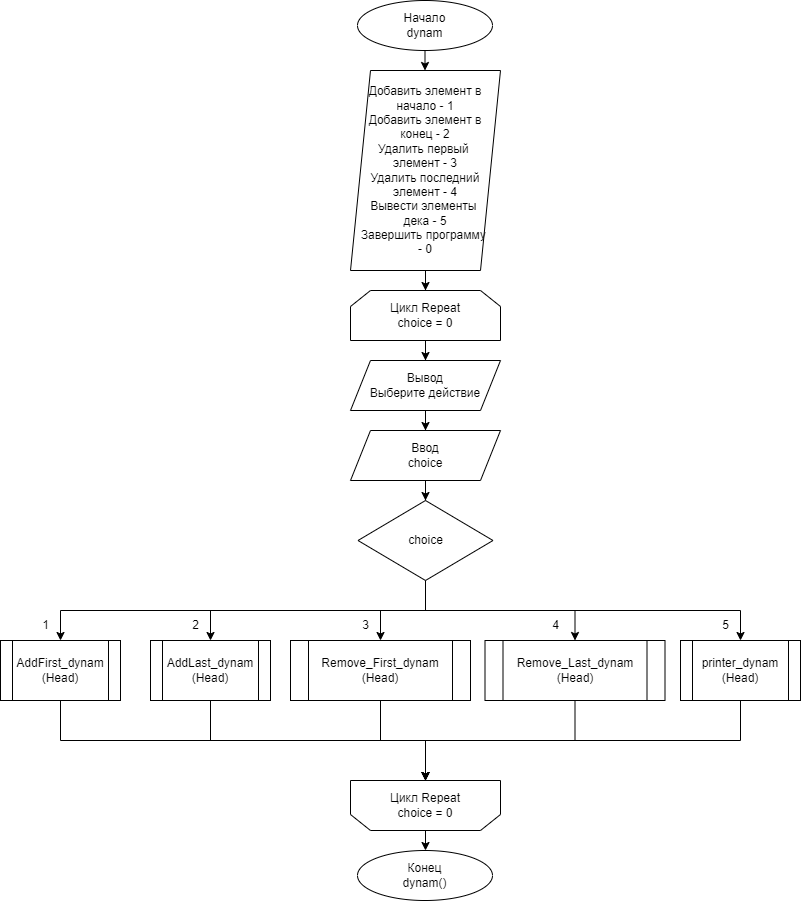


Рисунок 9 – алгоритм процедуры dynam

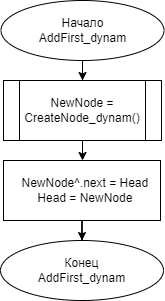


Рисунок 10 – алгоритм процедуры AddFirst\_dynam

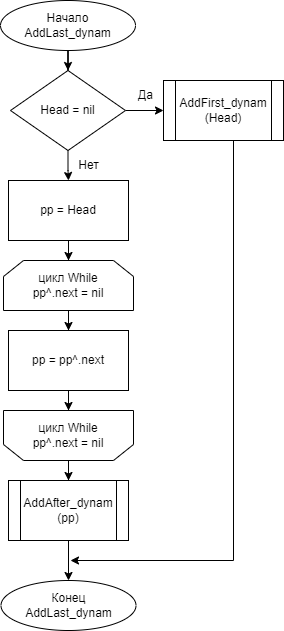


Рисунок 11 – алгоритм процедуры AddLast\_dynam

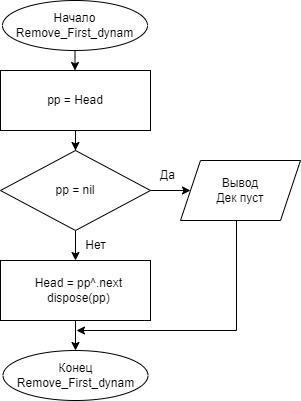


Рисунок 12 – алгоритм процедуры Remove\_First\_dynam

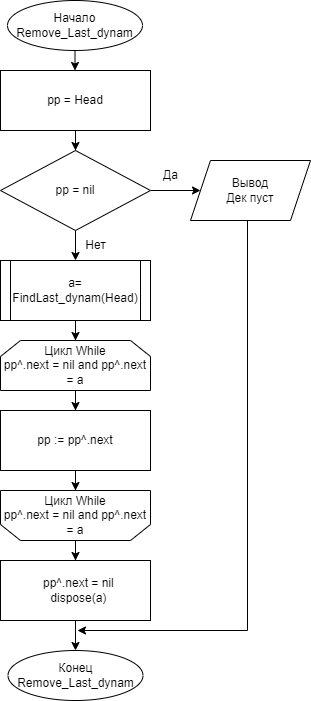


Рисунок 13 – алгоритм процедуры Remove\_Last\_dynam

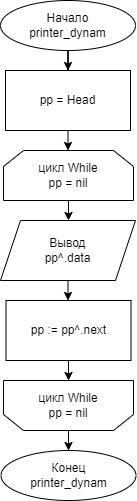


Рисунок 14 – алгоритм процедуры printer\_dynam

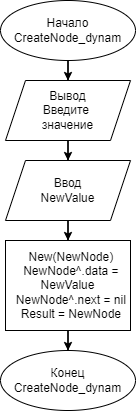


Рисунок 15 – алгоритм функции CreateNode\_dynam

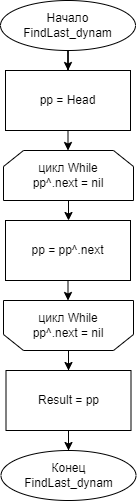


Рисунок 16 – алгоритм функции FindLast\_dynam

**Кода программы:**

**program** dkr6;

**Uses** CRT, statistic, dynamic;

**var** choice\_button: integer;

**procedure** choose();

**begin**

writeln('Использовать динамический дек - 1');

writeln('Использовать статический дек - 2');

choice\_button := ReadInteger;

**case** choice\_button **of**

1:dynam();

2:stat();

**end**;

**end**;

**begin**

choose();

**end**.

**Unit** statistic;

**const** MAXSIZE = 50;

**type**

deque = **record** {дек рассчитан на MAXSIZE символов}

data: **array**[1..MAXSIZE] **of** integer;

size: integer; {число элементов }

**end**;

**var** dek: deque;

choice: integer;

**procedure** AddFirst\_stat(**var** d: deque);

**var** arr:**array**[1..MAXSIZE] **of** integer;

**begin**

**if** (d.size = 0) **then**

**begin**

writeln('Введите первый элемент дека');

read(d.data[1]);

d.size += 1;

**end**

**else**

**begin**

**if** (d.size < MAXSIZE) **then**

**begin**

**for var** j := 1 **to** MAXSIZE **do**

**begin**

arr[j] := d.data[j];

**end**;

**for var** i := 1 **to** d.size **do**

**begin**

d.data[1] := 0;

d.data[i+1] := arr[i];

**end**;

writeln('Введите число');

read(d.data[1]);

d.size += 1

**end**

**else**

**begin**

writeln('Нет места для добавления элемента');

**end**;

**end**;

**end**;

**procedure** AddLast\_stat(**var** d:deque);

**begin**

**if** (d.size < MAXSIZE) **then**

**begin**

writeln('Введите последний элемент');

read(d.data[d.size+1]);

d.size += 1;

**end**

**else**

**begin**

writeln('Нет места для добавления элемента');

**end**;

**end**;

**procedure** PopFirst\_stat(**var** d:deque);

**begin**

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

**begin**

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

**end**;

d.size -= 1;

**end**;

**procedure** PopLast\_stat (**var** d:deque);

**begin**

d.data[d.size] := 0;

d.size -= 1;

**end**;

**procedure** Printer\_stat (d:deque);

**begin**

Writeln('Элементы дека');

**for var** i:= 1 **to** d.size **do**

**begin**

write(d.data[i], ' ');

**end**;

**end**;

**procedure** stat();

**begin**

writeln('Добавить элемент в начало - 1');

writeln('Добавить элемент в конец - 2');

writeln('Удалить первый элемент - 3');

writeln('Удалить последний элемент - 4');

writeln('Вывести элементы дека - 5');

writeln('Завершить программу - 0');

**repeat**

choice:=readInteger('Выберите действие');

**case** choice **of**

1:AddFirst\_stat(dek);

2:AddLast\_stat(dek);

3:PopFirst\_stat(dek);

4:PopLast\_stat(dek);

5:Printer\_stat(dek);

**end**;

**until** choice = 0;

**end**;

**end**.

**Unit** dynamic;

**Uses** CRT;

**type** PNode = ^deque;

deque = **record**

data: integer;

next: PNode;

**end**;

**function** CreateNode\_dynam (): PNode;

**var** NewNode: PNode;

NewValue: integer;

**begin**

writeln('Введите значение');

Readln(NewValue);

**New**(NewNode);

NewNode^.data := NewValue;

NewNode^.next := nil;

Result := NewNode;

**end**;

**procedure** AddFirst\_dynam ( **var** Head: PNode );

**var** NewNode: PNode;

**begin**

NewNode := CreateNode\_dynam();

NewNode^.next := Head;

Head := NewNode;

**end**;

**procedure** AddAfter\_dynam ( **var** Head: PNode );

**var** p, temp, NewNode: PNode;

**begin**

NewNode := CreateNode\_dynam();

p := Head;

temp := p^.next;

p^.next := NewNode;

NewNode^.next := temp;

**end**;

**procedure** AddLast\_dynam ( **var** Head: PNode);

**var** pp: PNode;

**begin**

**if** Head = nil **then**

AddFirst\_dynam ( Head )

**else begin**

pp := Head;

**while** pp^.next <> nil **do**

pp := pp^.next;

AddAfter\_dynam ( pp );

**end**;

**end**;

**procedure** printer\_dynam(Head: PNode);

**var**

pp: PNode;

**begin**

pp:=Head;

**while** pp <> nil **do**

**begin**

write(pp^.data, ' ');

pp := pp^.next;

**end**;

**end**;

**function** FindLast\_dynam(Head: PNode): PNode;

**var** pp: PNode;

**begin**

pp := Head;

**while** (pp^.next <> nil) **do**

**begin**

pp := pp^.next;

**end**;

Result := pp;

**end**;

**Procedure** Remove\_First\_dynam (**var** Head: PNode);

**var** pp: PNode;

**begin**

pp := Head;

**if** (pp = nil) **then** writeln('Дек пуст')

**else**

**begin**

Head := pp^.next;

dispose(pp);

**end**;

**end**;

**Procedure** Remove\_Last\_dynam (Head: PNode);

**var** a, pp: PNode;

**begin**

pp := Head;

**if** (pp = nil) **then** writeln('Дек пуст')

**else**

**begin**

a := FindLast\_dynam(Head);

**while** (pp^.next <> nil) **and** (pp^.next <> a) **do**

**begin**

pp := pp^.next;

**end**;

pp^.next := nil;

dispose(a);

**end**;

**end**;

**var** Head: PNode;

choice :integer;

**procedure** dynam();

**begin**

writeln('Добавить элемент в начало - 1');

writeln('Добавить элемент в конец - 2');

writeln('Удалить первый элемент - 3');

writeln('Удалить последний элемент - 4');

writeln('Вывести элементы дека - 5');

writeln('Завершить программу - 0');

**repeat**

choice:=readInteger('Выберите действие');

**case** choice **of**

1:AddFirst\_dynam(Head);

2:AddLast\_dynam(Head);

3:Remove\_First\_dynam(Head);

4:Remove\_Last\_dynam(Head);

5:printer\_dynam(Head);

**end**;

**until** choice = 0;

**end**;

**end**.

Результаты выполнения задания.



Рисунок 17 – начало работы программы

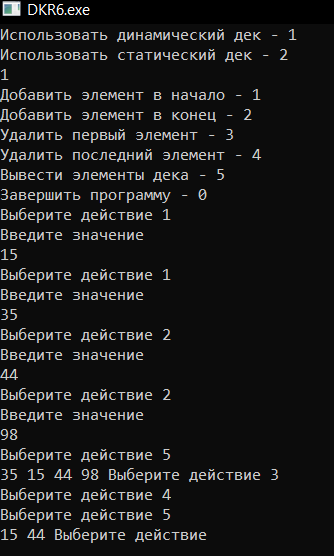


Рисунок 18 – работа программы на динамическом деке

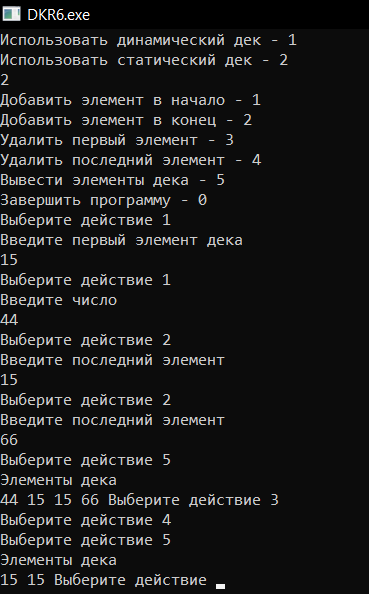


Рисунок 19 – работа программы на статическом деке

**Вывод:** в процессе выполнения задания проблем не возникло, была реализована структура данных «Дек» на основе статической памяти и динамической с возможностью добавления и удаления элемента. Так же в ходе работы взаимодействия со структурой данных были выполнены в форме case-меню. Были изучены принципы работы с базовыми структурами данных. Полученны навыки организации case-меню.