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 Казанцев Андрей Игоревич

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2024

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

**Задание:**

Вариант 10

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 и схема алгоритма:**

**TDeque** — тип записи, представляющий Deque. Он имеет массив Data для хранения элементов, Head Tail, а также для отслеживания передней и задней части Deque, а Count также для отслеживания количества элементов в Deque.

**Initialize** — это процедура, которая инициализирует пустой Deque.

**AddFront** и **AddRear** являются процедурами, которые добавляют элемент к передней и задней части Deque, соответственно.

**RemoveFront** и **RemoveRear** являются функциями, которые удаляют элемент из передней и задней части Deque соответственно и возвращают удаленный элемент.

**PrintDeque** — это процедура, которая печатает содержимое Deque.



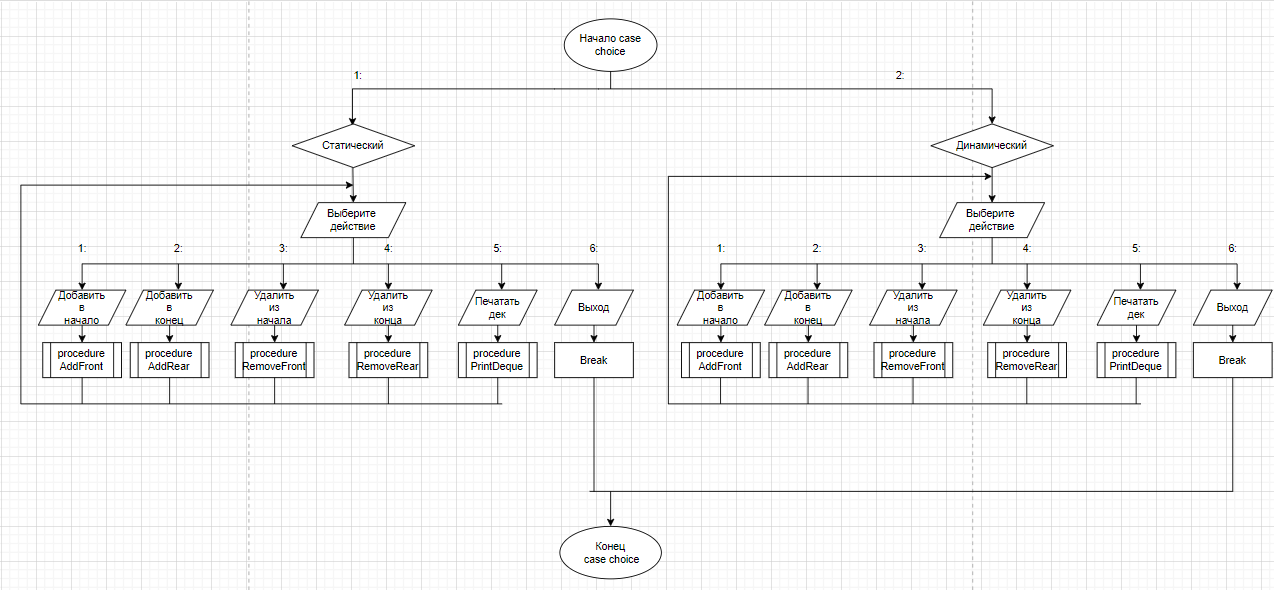


Рисунок 1 – общая схема

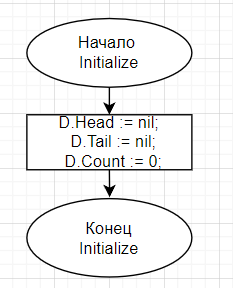


Рисунок 2 – процедура Initialize

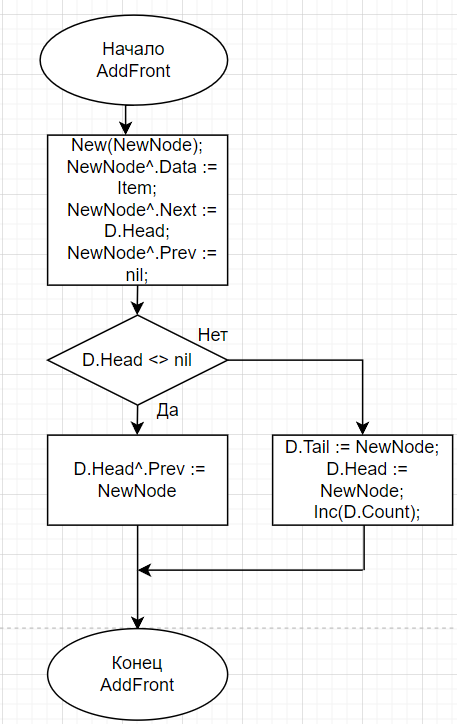


Рисунок 3 – процедура AddFront

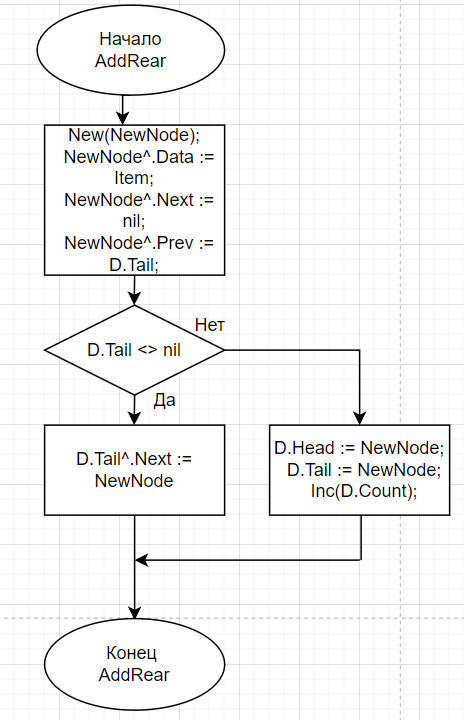


Рисунок 4 – процедура AddRear

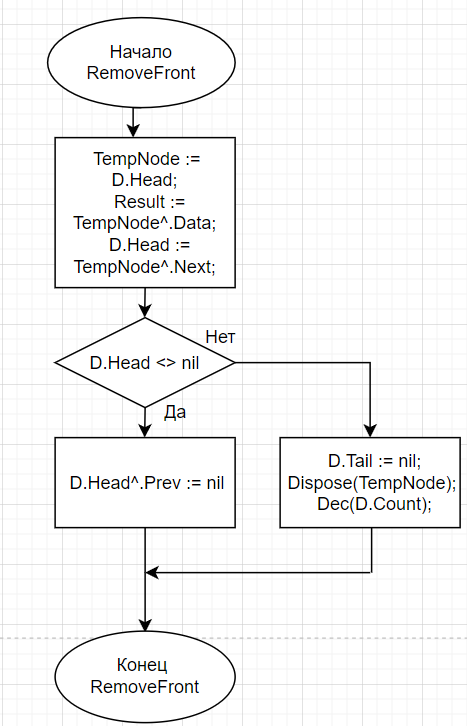


Рисунок 5 – процедура RemoveFront

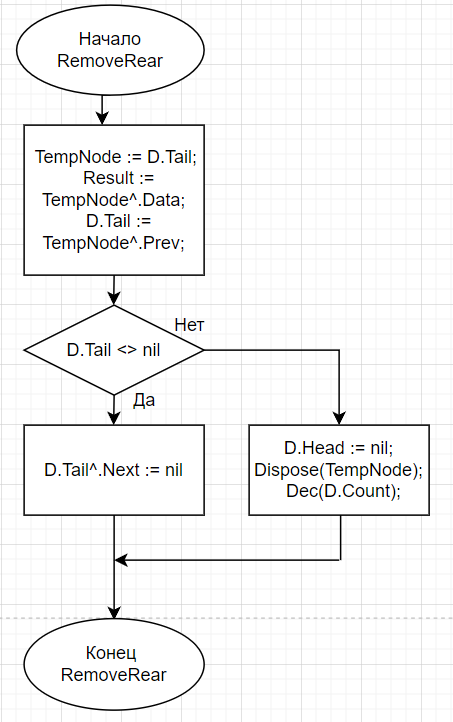


Рисунок 6 – процедура RemoveRear

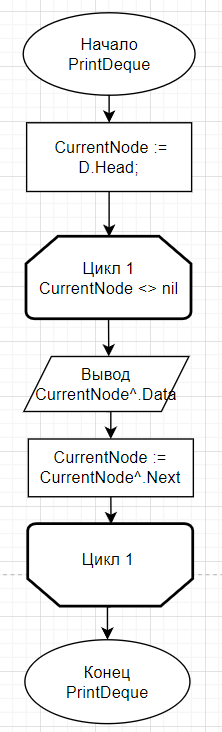


Рисунок 7 – процедура PrintDeque

**Код программы:**

**main**

**program** Main;

**uses**

CRT, StaticDeque, DynamicDeque;

**var**

choice, action, item: Integer;

StaticD: StaticDeque.TDeque;

DynamicD: DynamicDeque.TDeque;

**begin**

WriteLn('Выберите тип дека:');

WriteLn('1. Статический');

WriteLn('2. Динамический');

ReadLn(choice);

**case** choice **of**

1: **begin**

StaticDeque.Initialize(StaticD);

WriteLn('Вы выбрали статический дек.');

**while** True **do**

**begin**

WriteLn('Выберите действие:');

WriteLn('1. Добавить элемент в начало');

WriteLn('2. Добавить элемент в конец');

WriteLn('3. Удалить элемент из начала');

WriteLn('4. Удалить элемент из конца');

WriteLn('5. Печатать дек');

WriteLn('6. Выход');

ReadLn(action);

**case** action **of**

1: **begin**

Write('Введите элемент: ');

ReadLn(item);

StaticDeque.AddFront(StaticD, item);

**end**;

2: **begin**

Write('Введите элемент: ');

ReadLn(item);

StaticDeque.AddRear(StaticD, item);

**end**;

3: **begin**

WriteLn('Удаленный элемент из начала: ', StaticDeque.RemoveFront(StaticD));

**end**;

4: **begin**

WriteLn('Удаленный элемент из конца: ', StaticDeque.RemoveRear(StaticD));

**end**;

5: **begin**

Write('Дек: ');

StaticDeque.PrintDeque(StaticD);

**end**;

6: **begin**

**Break**;

**end**;

**end**;

**end**;

**end**;

2: **begin**

DynamicDeque.Initialize(DynamicD);

WriteLn('Вы выбрали динамический дек.');

**while** True **do**

**begin**

WriteLn('Выберите действие:');

WriteLn('1. Добавить элемент в начало');

WriteLn('2. Добавить элемент в конец');

WriteLn('3. Удалить элемент из начала');

WriteLn('4. Удалить элемент из конца');

WriteLn('5. Печатать дек');

WriteLn('6. Выход');

ReadLn(action);

**case** action **of**

1: **begin**

Write('Введите элемент: ');

ReadLn(item);

DynamicDeque.AddFront(DynamicD, item);

**end**;

2: **begin**

Write('Введите элемент: ');

ReadLn(item);

DynamicDeque.AddRear(DynamicD, item);

**end**;

3: **begin**

WriteLn('Удаленный элемент из начала: ', DynamicDeque.RemoveFront(DynamicD));

**end**;

4: **begin**

WriteLn('Удаленный элемент из конца: ', DynamicDeque.RemoveRear(DynamicD));

**end**;

5: **begin**

Write('Дек: ');

DynamicDeque.PrintDeque(DynamicD);

**end**;

6: **begin**

**Break**;

**end**;

**end**;

**end**;

**end**;

**end**;

**end**.

**Модуль StaticDeque**

**unit** StaticDeque;

**interface**

**const**

N = 100;

**type**

TDeque = **record**

Data: **array**[1..N] **of** Integer;

Head, Tail: Integer;

Count: Integer;

**end**;

**procedure** Initialize(**var** D: TDeque);

**procedure** AddFront(**var** D: TDeque; Item: Integer);

**procedure** AddRear(**var** D: TDeque; Item: Integer);

**function** RemoveFront(**var** D: TDeque): Integer;

**function** RemoveRear(**var** D: TDeque): Integer;

**procedure** PrintDeque(**var** D: TDeque);

**implementation**

**procedure** Initialize(**var** D: TDeque);

**begin**

D.Head := 1;

D.Tail := 0;

D.Count := 0;

**end**;

**procedure** AddFront(**var** D: TDeque; Item: Integer);

**begin**

**if** D.Count >= N **then**

**begin**

WriteLn('Дек полон');

**Exit**;

**end**;

**if** D.Head = 1 **then**

D.Head := N

**else**

Dec(D.Head);

D.Data[D.Head] := Item;

Inc(D.Count);

**end**;

**procedure** AddRear(**var** D: TDeque; Item: Integer);

**begin**

**if** D.Count >= N **then**

**begin**

WriteLn('Дек полон');

**Exit**;

**end**;

**if** D.Tail = N **then**

D.Tail := 1

**else**

Inc(D.Tail);

D.Data[D.Tail] := Item;

Inc(D.Count);

**end**;

**function** RemoveFront(**var** D: TDeque): Integer;

**begin**

**if** D.Count = 0 **then**

**begin**

WriteLn('Дек пуст');

**Exit**;

**end**;

Result := D.Data[D.Head];

**if** D.Head = N **then**

D.Head := 1

**else**

Inc(D.Head);

Dec(D.Count);

**end**;

**function** RemoveRear(**var** D: TDeque): Integer;

**begin**

**if** D.Count = 0 **then**

**begin**

WriteLn('Дек пуст');

**Exit**;

**end**;

Result := D.Data[D.Tail];

**if** D.Tail = 1 **then**

D.Tail := N

**else**

Dec(D.Tail);

Dec(D.Count);

**end**;

**procedure** PrintDeque(**var** D: TDeque);

**var**

i, idx: Integer;

**begin**

**if** D.Count = 0 **then**

**begin**

WriteLn('Дек пуст');

**end**;

idx := D.Head;

**for** i := 1 **to** D.Count **do**

**begin**

Write(D.Data[idx], ' ');

**if** idx = N **then**

idx := 1

**else**

Inc(idx);

**end**;

WriteLn;

**end**;

**end**.

**Модуль Dynamic Deque**

**unit** DynamicDeque;

**interface**

**type**

PNode = ^TNode;

TNode = **record**

Data: Integer;

Next, Prev: PNode;

**end**;

TDeque = **record**

Head, Tail: PNode;

Count: Integer;

**end**;

**procedure** Initialize(**var** D: TDeque);

**procedure** AddFront(**var** D: TDeque; Item: Integer);

**procedure** AddRear(**var** D: TDeque; Item: Integer);

**function** RemoveFront(**var** D: TDeque): Integer;

**function** RemoveRear(**var** D: TDeque): Integer;

**procedure** PrintDeque(**var** D: TDeque);

**implementation**

**procedure** Initialize(**var** D: TDeque);

**begin**

D.Head := nil;

D.Tail := nil;

D.Count := 0;

**end**;

**procedure** AddFront(**var** D: TDeque; Item: Integer);

**var**

NewNode: PNode;

**begin**

**New**(NewNode);

NewNode^.Data := Item;

NewNode^.Next := D.Head;

NewNode^.Prev := nil;

**if** D.Head <> nil **then**

D.Head^.Prev := NewNode

**else**

D.Tail := NewNode;

D.Head := NewNode;

Inc(D.Count);

**end**;

**procedure** AddRear(**var** D: TDeque; Item: Integer);

**var**

NewNode: PNode;

**begin**

**New**(NewNode);

NewNode^.Data := Item;

NewNode^.Next := nil;

NewNode^.Prev := D.Tail;

**if** D.Tail <> nil **then**

D.Tail^.Next := NewNode

**else**

D.Head := NewNode;

D.Tail := NewNode;

Inc(D.Count);

**end**;

**function** RemoveFront(**var** D: TDeque): Integer;

**var**

TempNode: PNode;

**begin**

**if** D.Count = 0 **then**

**begin**

WriteLn('Дек пуст');

**Exit**;

**end**;

TempNode := D.Head;

Result := TempNode^.Data;

D.Head := TempNode^.Next;

**if** D.Head <> nil **then**

D.Head^.Prev := nil

**else**

D.Tail := nil;

Dispose(TempNode);

Dec(D.Count);

**end**;

**function** RemoveRear(**var** D: TDeque): Integer;

**var**

TempNode: PNode;

**begin**

**if** D.Count = 0 **then**

**begin**

WriteLn('Дек пуст');

**Exit**;

**end**;

TempNode := D.Tail;

Result := TempNode^.Data;

D.Tail := TempNode^.Prev;

**if** D.Tail <> nil **then**

D.Tail^.Next := nil

**else**

D.Head := nil;

Dispose(TempNode);

Dec(D.Count);

**end**;

**procedure** PrintDeque(**var** D: TDeque);

**var**

CurrentNode: PNode;

**begin**

**if** D.Count = 0 **then**

**begin**

WriteLn('Дек пуст');

**Exit**;

**end**;

CurrentNode := D.Head;

**while** CurrentNode <> nil **do**

**begin**

Write(CurrentNode^.Data, ' ');

CurrentNode := CurrentNode^.Next;

**end**;

WriteLn;

**end**;

**end**.

**Результат выполнения программы:**

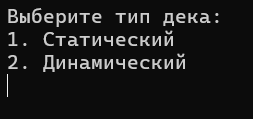
****

Рисунок 8 – меню выбора типа дека

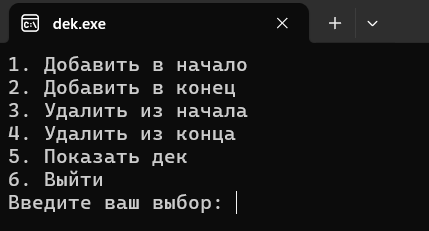
****

Рисунок 9 – меню выбора процедуры

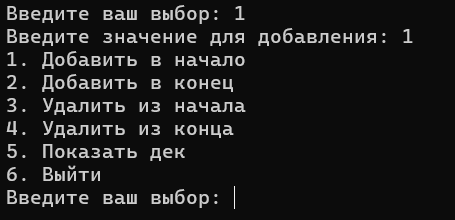


Рисунок 10 – добавление в начало

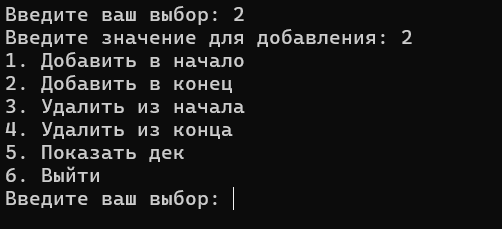


Рисунок 11 – добавление в конец

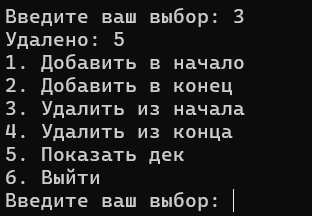


Рисунок 12 – удаление из начала

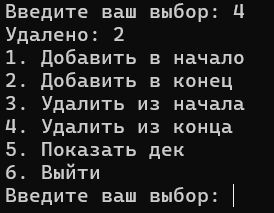


Рисунок 13 – удаление из конца

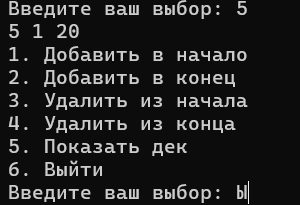


Рисунок 14 – показать дек

**Вывод**

Цель данной работы заключалась в том, чтобы изучить принципы работы с базовыми структурных данных, получить навыки организации case-меню. Эта цель была осуществима с помощью выполнения задания на написание программы для работы со структурой данных «Дек». Данное задание отличалось количеством используемых процедур и функций. Тем самым оно вызывало больший интерес, т.к., выполняя, эти задания можно было столкнуться с неизвестными ранее функциями и процедурами в языке программирования Паскаль, такими как: Initialize, AddFront, AddRear, RemoveFront, RemoveRear, PrintDeque, и др. Данные функции были изучены и применены в разработке программы для нахождения значения функции.

Не менее интересными были результаты выполнения программы. При запуске кода, программа, а при нажатии на определённые клавиши можно менять местами элементы строки, ставить любое число в начало или конец строки.

Говоря, о написании программы и анализа результата её выполнения, нельзя не упомянуть о составлении схемы алгоритма, благодаря которой в последствие была написана программа. Схема алгоритма была составлена на сайте www.drawio.com – это удобная платформа для создания подобных схем алгоритмов. Составить здесь схему алгоритмов сможет любой человек, который когда-либо составлял их на бумаге. Работы получаются очень аккуратными и красивыми.

Подводя итог, можно сказать, что в данной работе была достигнута цель и были освоены новые способы выполнения заданий, а именно создание электронных схем алгоритмов. Работа проведена успешно, получен новый интересный и приятный опыт.