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-204-52-00

Вейс Роман Михайлович

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2024

1. Цель работы

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

2. Формулировка задания.

Вариант 5.

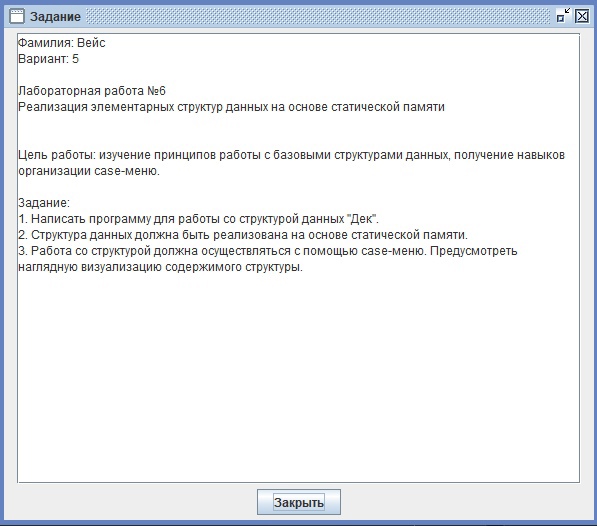


Рисунок 1 – Вариант работы

3. Описание алгоритма.

Дек может быть одного из двух типов: статическим или динамическим. Пользователь сначала выбирает тип дека.

Далее программа предоставляет пользователю следующие возможности для работы с деком:

* Добавление элемента в дек: пользователь указывает, куда добавить элемент (в начало или в конец) и вводит символ.
* Удаление элемента из дека: пользователь указывает, из какого конца удалить элемент.
* Проверка, пуст ли дек.
* Просмотр содержимого дека.

Пользователь может взаимодействовать с деком, выбирая одну из приведенных выше операций. Программа продолжает работу, пока пользователь не завершит ее, выбрав соответствующую опцию.

4. Схема.

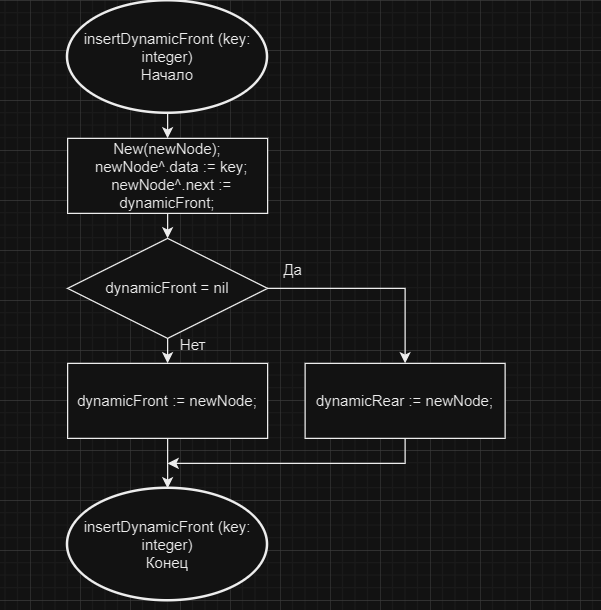


Рисунок 2 – Процедура insertDynamicFront

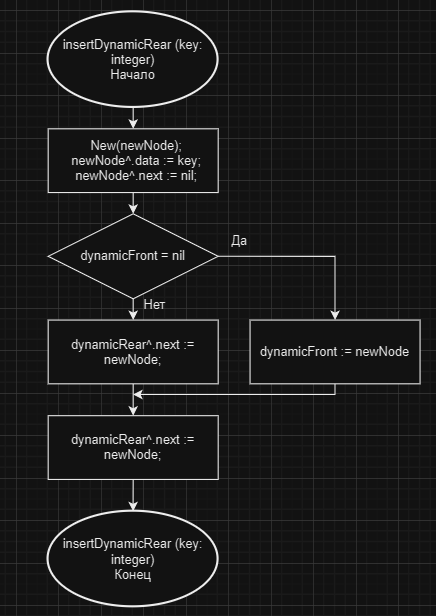


Рисунок 3 – Процедура InsertDynamicRear

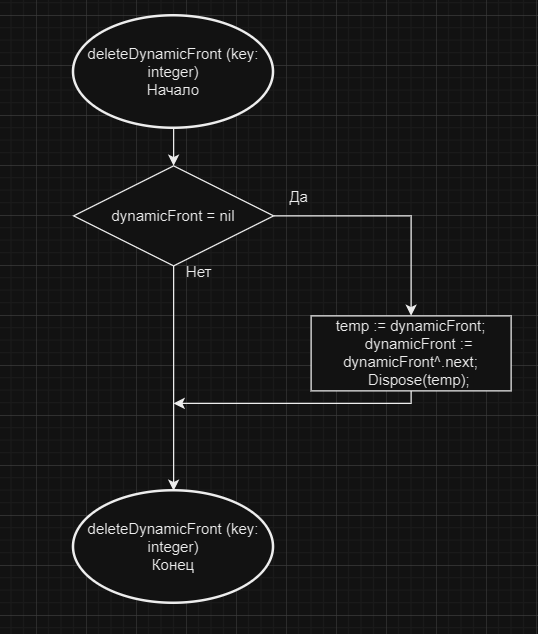


Рисунок 4 – Процедура deleteDynamicFront

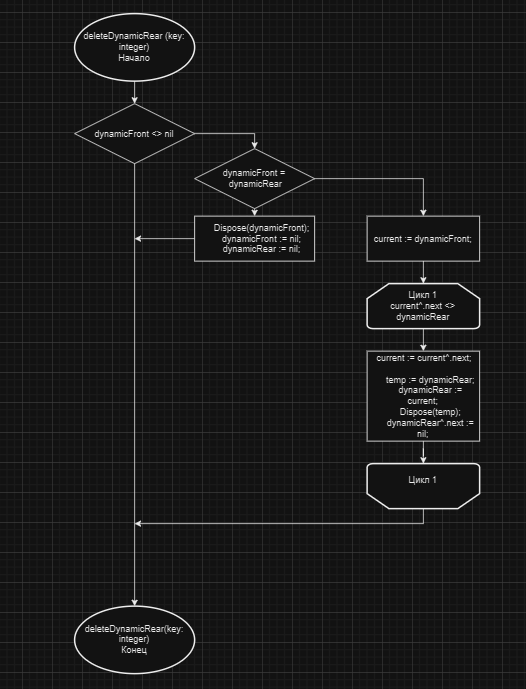
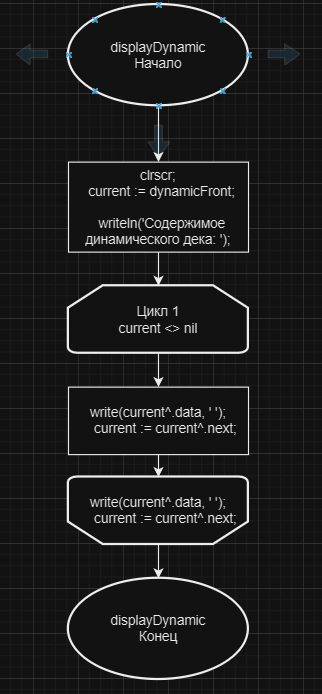


Рисунок 5 – Процедура deleteDynamicRear

  
Рисунок 6 – Процедура displayDynamic

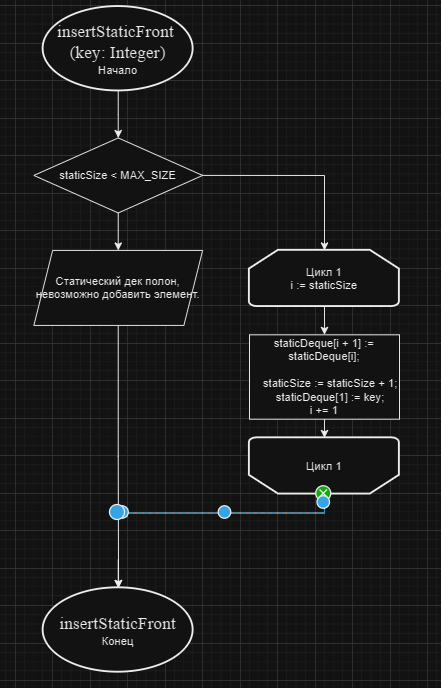


Рисунок 7 – Процедура insertStaticFront

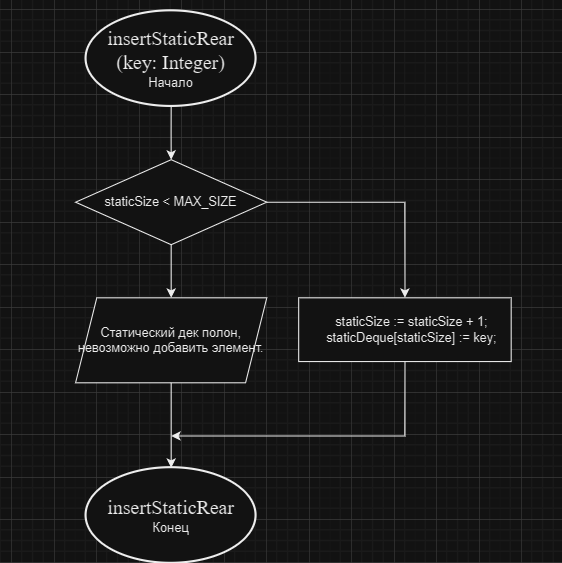


Рисунок 8 – Процедура insertStaticRear

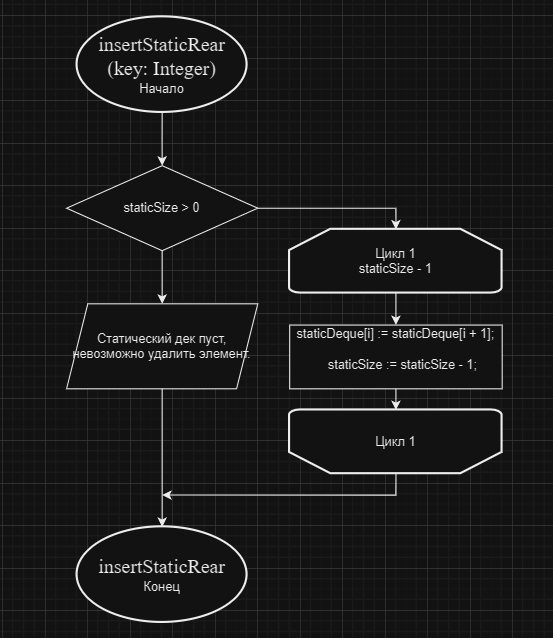


Рисунок 9 – Процедура deleteStaticFront

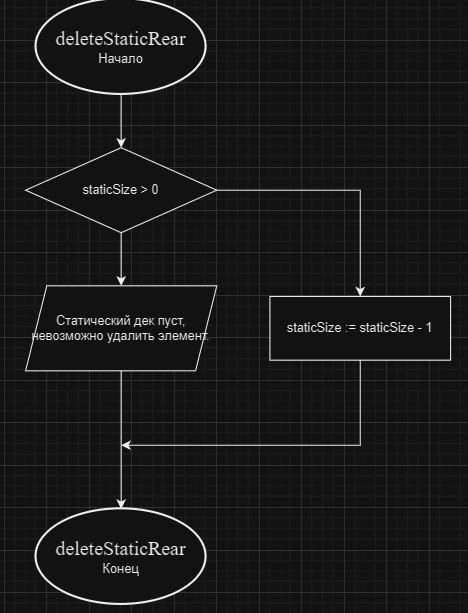


Рисунок 10 – Процедура deleteStaticRear

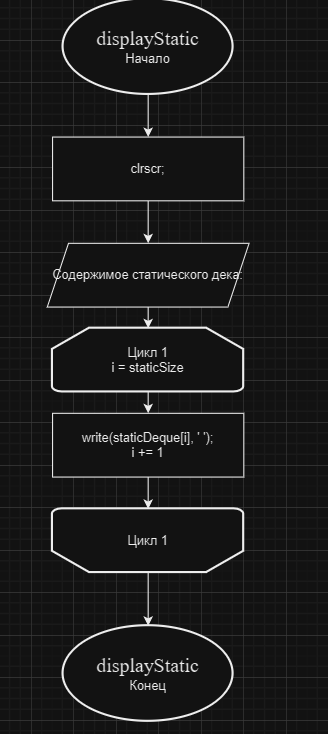


Рисунок 11 – Процедура displayStatic

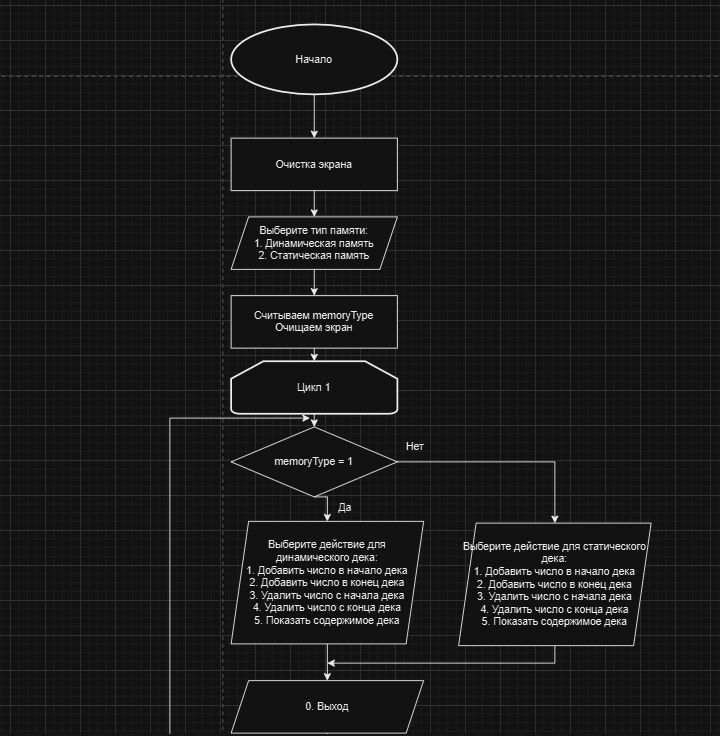


Рисунок 12.1 – Основная программа

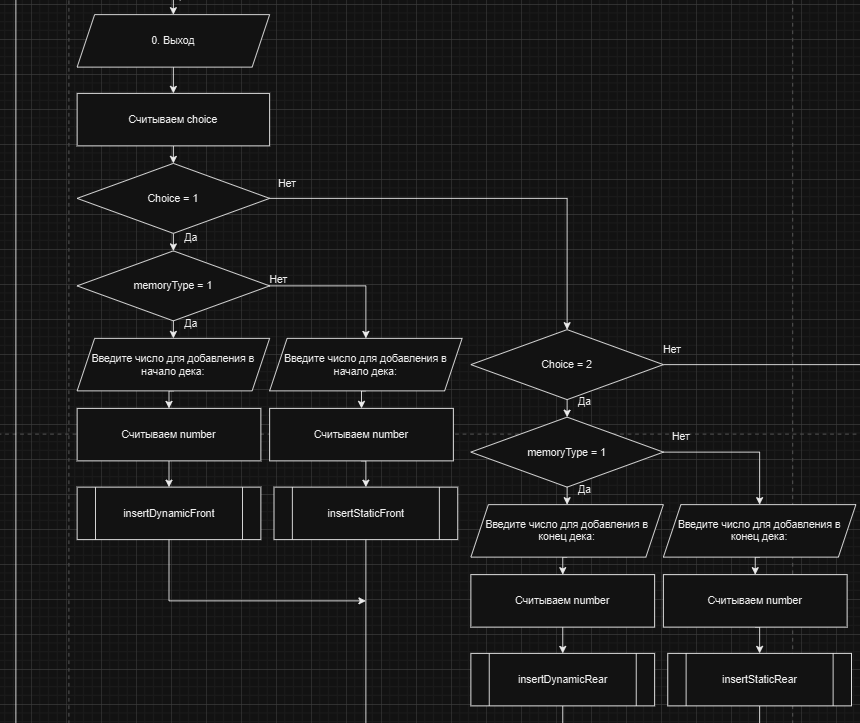


Рисунок 12.2 – Основная программа

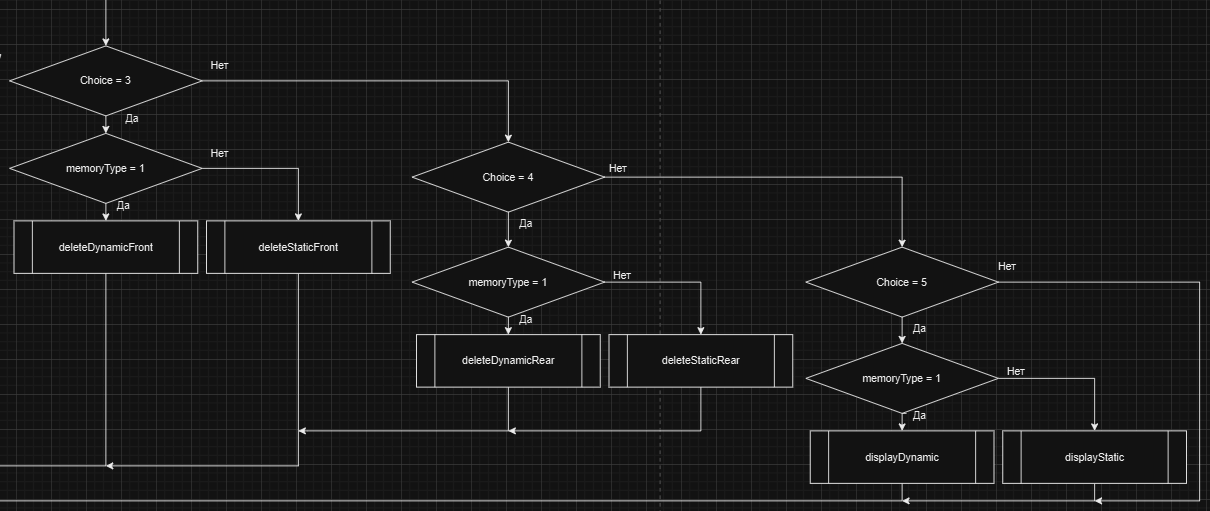


Рисунок 12.3 – Основная программа

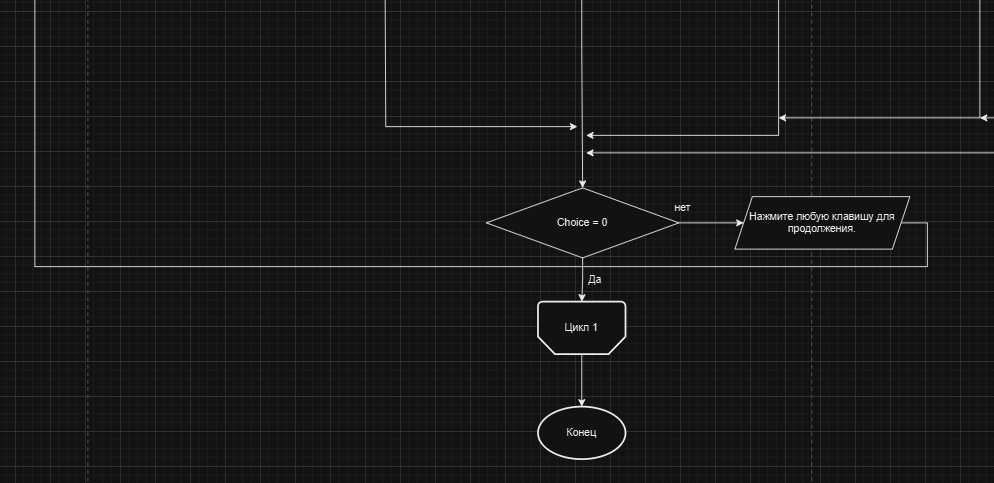


Рисунок 12.4 – Основная программа

5. Код программы.

**program** DequeMemoryType;

**uses**

crt;

**const**

MAX\_SIZE = 3;

**type**

Node = **record**

data: Integer;

next: ^Node;

**end**;

**var**

dynamicFront, dynamicRear: ^Node;

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

staticFront, staticRear, staticSize: Integer;

choice, number, memoryType: Integer;

**procedure** insertDynamicFront(key: Integer);

**var**

newNode: ^Node;

**begin**

**New**(newNode);

newNode^.data := key;

newNode^.next := dynamicFront;

**if** dynamicFront = nil **then**

dynamicRear := newNode;

dynamicFront := newNode;

**end**;

**procedure** insertDynamicRear(key: Integer);

**var**

newNode: ^Node;

**begin**

**New**(newNode);

newNode^.data := key;

newNode^.next := nil;

**if** dynamicFront = nil **then**

dynamicFront := newNode

**else**

dynamicRear^.next := newNode;

dynamicRear := newNode;

**end**;

**procedure** deleteDynamicFront;

**var**

temp: ^Node;

**begin**

**if** dynamicFront <> nil **then**

**begin**

temp := dynamicFront;

dynamicFront := dynamicFront^.next;

Dispose(temp);

**end**;

**end**;

**procedure** deleteDynamicRear;

**var**

current, temp: ^Node;

**begin**

**if** dynamicFront <> nil **then**

**begin**

**if** dynamicFront = dynamicRear **then**

**begin**

Dispose(dynamicFront);

dynamicFront := nil;

dynamicRear := nil;

**end**

**else**

**begin**

current := dynamicFront;

**while** current^.next <> dynamicRear **do**

current := current^.next;

temp := dynamicRear;

dynamicRear := current;

Dispose(temp);

dynamicRear^.next := nil;

**end**;

**end**;

**end**;

**procedure** displayDynamic;

**var**

current: ^Node;

**begin**

clrscr;

current := dynamicFront;

writeln('Содержимое динамического дека: ');

**while** current <> nil **do**

**begin**

write(current^.data, ' ');

current := current^.next;

**end**;

writeln;

**end**;

**procedure** insertStaticFront(key: Integer);

**begin**

**if** staticSize < MAX\_SIZE **then**

**begin**

**for var** i := staticSize **downto** 1 **do**

staticDeque[i + 1] := staticDeque[i];

staticSize := staticSize + 1;

staticDeque[1] := key;

**end**

**else**

writeln('Статический дек полон, невозможно добавить элемент.');

**end**;

**procedure** insertStaticRear(key: Integer);

**begin**

**if** staticSize < MAX\_SIZE **then**

**begin**

staticSize := staticSize + 1;

staticDeque[staticSize] := key;

**end**

**else**

writeln('Статический дек полон, невозможно добавить элемент.');

**end**;

**procedure** deleteStaticFront;

**begin**

**if** staticSize > 0 **then**

**begin**

**for var** i := 1 **to** staticSize - 1 **do**

staticDeque[i] := staticDeque[i + 1];

staticSize := staticSize - 1;

**end**

**else**

writeln('Статический дек пуст, невозможно удалить элемент.');

**end**;

**procedure** deleteStaticRear;

**begin**

**if** staticSize > 0 **then**

staticSize := staticSize - 1

**else**

writeln('Статический дек пуст, невозможно удалить элемент.');

**end**;

**procedure** displayStatic;

**begin**

clrscr;

writeln('Содержимое статического дека: ');

**for var** i := 1 **to** staticSize **do**

write(staticDeque[i], ' ');

writeln;

**end**;

**begin**

clrscr;

dynamicFront := nil;

dynamicRear := nil;

staticFront := 1;

staticRear := 1;

staticSize := 0;

writeln('Выберите тип памяти:');

writeln('1. Динамическая память');

writeln('2. Статическая память');

readln(memoryType);

**repeat**

clrscr;

**if** memoryType = 1 **then**

**begin**

writeln('Выберите действие для динамического дека:');

writeln('1. Добавить число в начало дека');

writeln('2. Добавить число в конец дека');

writeln('3. Удалить число с начала дека');

writeln('4. Удалить число с конца дека');

writeln('5. Показать содержимое дека');

**end**

**else if** memoryType = 2 **then**

**begin**

writeln('Выберите действие для статического дека:');

writeln('1. Добавить число в начало дека');

writeln('2. Добавить число в конец дека');

writeln('3. Удалить число с начала дека');

writeln('4. Удалить число с конца дека');

writeln('5. Показать содержимое дека');

**end**;

writeln('0. Выход');

readln(choice);

clrscr;

**case** choice **of**

1: **begin**

**if** memoryType = 1 **then**

**begin**

writeln('Введите число для добавления в начало дека: ');

readln(number);

insertDynamicFront(number);

**end**

**else if** memoryType = 2 **then**

**begin**

writeln('Введите число для добавления в начало дека: ');

readln(number);

insertStaticFront(number);

**end**;

**end**;

2: **begin**

**if** memoryType = 1 **then**

**begin**

writeln('Введите число для добавления в конец дека: ');

readln(number);

insertDynamicRear(number);

**end**

**else if** memoryType = 2 **then**

**begin**

writeln('Введите число для добавления в конец дека: ');

readln(number);

insertStaticRear(number);

**end**;

**end**;

3: **begin**

**if** memoryType = 1 **then**

deleteDynamicFront

**else if** memoryType = 2 **then**

deleteStaticFront;

**end**;

4: **begin**

**if** memoryType = 1 **then**

deleteDynamicRear

**else if** memoryType = 2 **then**

deleteStaticRear;

**end**;

5: **begin**

**if** memoryType = 1 **then**

displayDynamic

**else if** memoryType = 2 **then**

displayStatic;

**end**;

**end**;

**if** (choice >= 1) **and** (choice <= 5) **then**

**begin**

writeln;

writeln('Нажмите любую клавишу для продолжения.');

readkey;

**end**;

**until** choice = 0;

**end**.

6. Результат выполнения программы.

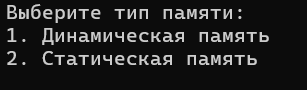


Рисунок 13

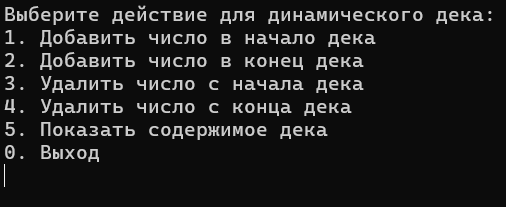


Рисунок 14

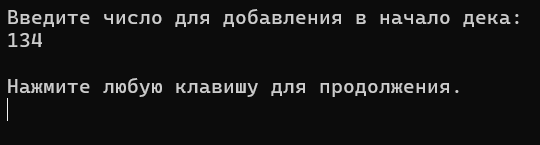


Рисунок 15 – добавление в начало дека

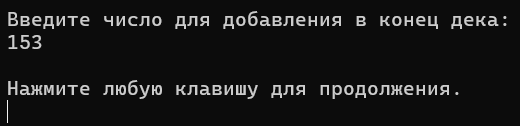


Рисунок 16

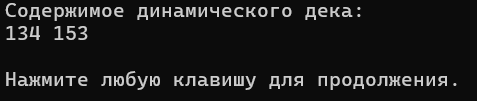


Рисунок 17

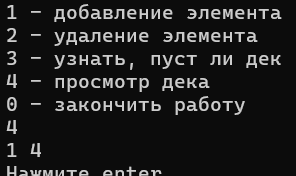


Рисунок 14

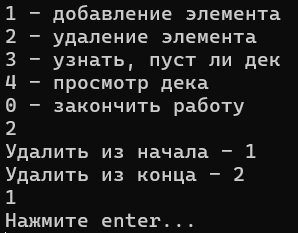


Рисунок 15

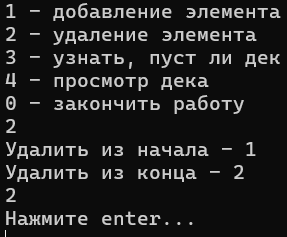


Рисунок 16

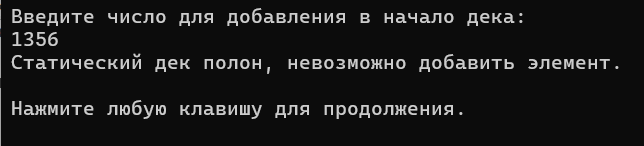


Рисунок 17

7. Вывод.

Выполняя домашнюю контрольную работу, мы смогли достичь поставленной цели: мы изучили принципы работы с базовыми структурами данных, а также получили навыки организации case-меню.

В ходе работы над задание мы вспомнили и поработали с процедурами и функциями внутри Pascal, вспомнили как работает case-меню.