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

Шумилов Иван Андреевич

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

202

1. **Цель домашней контрольной работы:**

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

**2. Формулировка задания**

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

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.

**3. Описание алгоритма**

**Данная программа на языке Pascal реализует структуру данных "очередь" с использованием связанного списка. Очередь представлена в виде односвязного списка, где каждый элемент содержит значение (inf) и указатель (link) на следующий элемент.**

**Основные операции, реализованные в программе:**

**1. \*\*addElem\*\*: Процедура добавления нового элемента в конец очереди. Пользователю предлагается ввести значение элемента. Если очередь пуста, добавляемый элемент становится началом очереди (begQ). Иначе, программа проходит по списку до конца и добавляет новый элемент в конец.**

**2. \*\*delElem\*\*: Процедура удаления первого элемента из начала очереди. Если очередь пуста, выводится сообщение об ошибке. Иначе, первый элемент удаляется, а начало очереди (begQ) сдвигается на следующий элемент.**

**3. \*\*printQ\*\*: Процедура вывода всех элементов очереди на экран в прямом порядке (от начала к концу).**

**4. \*\*recPrintQ\*\*: Процедура рекурсивного вывода всех элементов очереди на экран в обратном порядке (от конца к началу).**

**5. \*\*getCountElem\*\*: Функция подсчета количества элементов в очереди. Она проходит по всем элементам очереди и подсчитывает их количество.**

**Программа также содержит меню, в котором пользователь может выбрать одну из следующих операций:**

**1. Добавить новый элемент в очередь**

**2. Удалить первый элемент из очереди**

**3. Вывести все элементы очереди на экран в прямом порядке**

**4. Вывести все элементы очереди на экран в обратном порядке**

**5. Получить количество элементов в очереди**

**6. Выход из программы**

**Программа работает в цикле, пока пользователь не выберет пункт "Выход" в меню.**

**4. Схема алгоритма**

**Изображение выглядит как снимок экрана, черный, диаграмма, монохромный

Автоматически созданное описание**

**Изображение выглядит как снимок экрана, черно-белый, черный, дизайн

Автоматически созданное описание**

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

Автоматически созданное описание**

**Изображение выглядит как текст, снимок экрана, диаграмма

Автоматически созданное описание**

**5.Код программы**

**program** shestdkr;

**uses**

crt, dynqueue, staticqueue;

**function** mainMenu: integer;

**var**

sel: integer;

**begin**

**repeat**

clrscr;

writeln('1 - ДОБАВИТЬ ЭЛЕМЕНТ В ОЧЕРЕДЬ');

writeln('2 - УДАЛИТЬ ЭЛЕМЕНТ ИЗ ОЧЕРЕДИ');

writeln('3 - ПЕЧАТЬ ЭЛЕМЕНТОВ ОЧЕРЕДИ ОТ НАЧАЛА К КОНЦУ');

writeln('4 - ПЕЧАТЬ ПЕРВОГО ЭЛЕМЕНТА В ОЧЕРЕДИ');

writeln('5 - ПОЛУЧИТЬ КОЛИЧЕСТВО ЭЛЕМЕНТОВ ОЧЕРЕДИ');

writeln('6 - ВЫЙТИ В ГЛАВНОЕ МЕНЮ');

writeln('7 - ВЫХОД');

write('Выберите один из пунктов меню: ');

readln(sel);

**until** ((sel >= 1) **and** (sel <= 7));

writeln;

mainMenu := sel;

**end**;

**procedure** choiceMemory(**var** isDynamic: boolean);

**var**

choice: integer;

**begin**

**repeat**

clrscr;

writeln('Выберите способ реализации очереди:');

writeln('1 - Динамическая реализация');

writeln('2 - Статическая реализация');

write('Ввод: ');

readln(choice);

**case** choice **of**

1: isDynamic := true;

2: isDynamic := false;

**else**

writeln('Неверный выбор!');

**end**;

**until** (choice **in** [1, 2]);

**if** isDynamic **then**

dynqueue.begQ := NIL

**else**

staticqueue.initStaticQueue;

**end**;

**var**

sel: integer;

elem: char;

isDynamic: boolean;

**begin**

choiceMemory(isDynamic);

**repeat**

sel := mainMenu;

**case** sel **of**

1:

**begin**

**if** isDynamic **then**

**begin**

dynqueue.addDynamicElem;

writeln('Элемент успешно добавлен в конец очереди!');

**end**

**else**

**begin**

write('Введите значение добавляемого элемента: ');

readln(elem);

staticqueue.addStaticElem(elem);

writeln('Элемент успешно добавлен в конец очереди!');

**end**;

readkey;

**end**;

2:

**begin**

**if** isDynamic **then**

**begin**

**if** (dynqueue.begQ = NIL) **then**

writeln('В очереди нет ни одного элемента! Удаление невозможно!')

**else**

**begin**

dynqueue.delDynamicElem;

writeln('Элемент успешно удален из начала очереди!');

**end**;

**end**

**else**

**begin**

elem := staticqueue.delStaticElem;

**if** elem <> ' ' **then**

writeln('Элемент ', elem, ' успешно удален из начала очереди!')

**else**

readkey;

**end**;

**end**;

3:

**begin**

**if** isDynamic **then**

**begin**

**if** (dynqueue.begQ = NIL) **then**

writeln('В очереди нет ни одного элемента! Печать невозможна!')

**else**

**begin**

write('Элементы очереди имеют вид: ');

dynqueue.printDynamicQ;

**end**;

**end**

**else**

**begin**

staticqueue.printStaticQueue;

**end**;

readkey;

**end**;

4:

**begin**

**if** isDynamic **then**

**begin**

**if** (dynqueue.begQ = NIL) **then**

writeln('В очереди нет ни одного элемента! Печать невозможна!')

**else**

**begin**

write('Элементы очереди имеют вид: ');

dynqueue.recPrintDynamicQ(dynqueue.begQ);

**end**;

**end**

**else**

**begin**

**if** staticqueue.isStaticQueueEmpty **then**

writeln('В очереди нет ни одного элемента! Печать невозможна!')

**else**

**begin**

write('Элементы очереди имеют вид: ');

staticqueue.printStaticQueue;

**end**;

**end**;

readkey;

**end**;

5:

**begin**

**if** isDynamic **then**

writeln('Количество элементов в очереди: ', dynqueue.getCountDynamicElem)

**else**

**begin**

**var** i, p: integer;

i := 0;

p := staticqueue.front;

**while** p <> staticqueue.rear **do**

**begin**

inc(i);

p := (p + 1) **mod** MAX\_QUEUE\_SIZE;

**end**;

writeln('Количество элементов в очереди: ', i);

**end**;

readkey;

**end**;

6:

**begin**

choiceMemory(isDynamic);

**end**;

**end**;

**until** sel = 7;

**end**.

**unit** staticqueue;

**interface**

**const**

MAX\_QUEUE\_SIZE = 5;

**type**

Telem = **record**

inf: char;

**end**;

**var**

queue: **array**[0..MAX\_QUEUE\_SIZE-1] **of** Telem;

front, rear: integer;

**procedure** initStaticQueue;

**function** isStaticQueueEmpty: boolean;

**function** isStaticQueueFull: boolean;

**procedure** addStaticElem(x: char);

**function** delStaticElem: char;

**procedure** printStaticQueue;

**implementation**

**procedure** initStaticQueue;

**begin**

front := 0;

rear := 0;

**end**;

**function** isStaticQueueEmpty: boolean;

**begin**

isStaticQueueEmpty := front = rear;

**end**;

**function** isStaticQueueFull: boolean;

**begin**

isStaticQueueFull := (rear + 1) **mod** MAX\_QUEUE\_SIZE = front;

**end**;

**procedure** addStaticElem(x: char);

**begin**

**if not** isStaticQueueFull **then**

**begin**

queue[rear].inf := x;

rear := (rear + 1) **mod** MAX\_QUEUE\_SIZE;

**end**

**else**

writeln('Очередь переполнена!');

**end**;

**function** delStaticElem: char;

**var**

x: char;

**begin**

**if not** isStaticQueueEmpty **then**

**begin**

x := queue[front].inf;

front := (front + 1) **mod** MAX\_QUEUE\_SIZE;

delStaticElem := x;

**end**

**else**

**begin**

writeln('Очередь пуста!');

delStaticElem := ' ';

**end**;

**end**;

**procedure** printStaticQueue;

**var**

i: integer;

**begin**

**if** isStaticQueueEmpty **then**

writeln('Очередь пуста!')

**else**

**begin**

i := front;

**while** i <> rear **do**

**begin**

write(queue[i].inf, ' ');

i := (i + 1) **mod** MAX\_QUEUE\_SIZE;

**end**;

writeln;

**end**;

**end**;

**end**.

**unit** dynqueue;

**interface**

**type**

Tptr = ^Telem;

Telem = **record**

inf: char;

link: Tptr;

**end**;

**var**

begQ: Tptr;

**procedure** addDynamicElem;

**procedure** delDynamicElem;

**procedure** printDynamicQ;

**procedure** recPrintDynamicQ(pp: Tptr);

**function** getCountDynamicElem: integer;

**implementation**

**procedure** addDynamicElem;

**var**

add: Tptr;

p: Tptr;

**begin**

**new**(add);

add^.link := NIL;

write('Введите значение добавляемого элемента: ');

readln(add^.inf);

**if** (begQ = NIL) **then**

begQ := add

**else**

**begin**

p := begQ;

**while** (p^.link <> NIL) **do**

p := p^.link;

p^.link := add;

**end**;

**end**;

**procedure** delDynamicElem;

**var**

p: Tptr;

**begin**

p := begQ;

begQ := begQ^.link;

p^.link := NIL;

dispose(p);

**end**;

**procedure** printDynamicQ;

**var**

p: Tptr;

**begin**

p := begQ;

**while** (p <> NIL) **do**

**begin**

write(p^.inf:3);

p := p^.link;

**end**;

**end**;

**procedure** recPrintDynamicQ(pp: Tptr);

**begin**

**if** (pp <> NIL) **then**

**begin**

recPrintDynamicQ(pp^.link);

write(pp^.inf:3);

**end**;

**end**;

**function** getCountDynamicElem: integer;

**var**

p: Tptr;

k: integer;

**begin**

k := 0;

p := begQ;

**while** (p <> NIL) **do**

**begin**

k := k + 1;

p := p^.link;

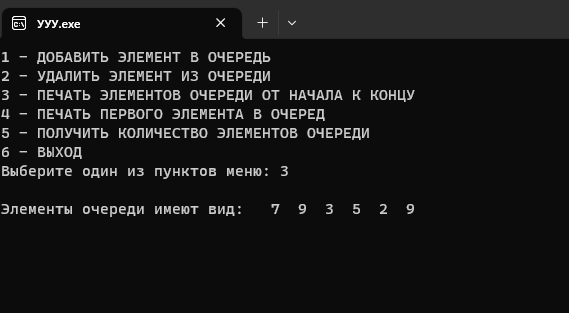
**end**;

getCountDynamicElem := k;

**end**;

**end**.

**5. Результат работы программы**



**6. Вывод**

В ходе выполнения домашней контрольной работы номер 6 были получены базовые навыки реализации консольных приложений на основе case-меню и работы со статическими структурами данных. Работа была разделена на несколько этапов: цель, формулировка задания, написание кода программы, результат выполнения программы, вывод. Код программы был написан на языке программирования Pascal. Делая вывод, стоит отметить, что в ходе работы все цели были достигнуты.