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, ПОЛУЧЕНИЕ НАВЫКОВ ОРГАНИЗАЦИИ CASE-МЕНЮ»**

**ПО «МДК 05.02 РАЗРАБОТКА КОДА ИНФОРМАЦИОННЫХ СИСТЕМ»**

Выполнил: студент учебной группы

ИСПк-205-52-00

Веселов Алексей Денисович

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2024

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

2. Формулировка задания (с вариантом)

Вариант: 19

1) Написать программу для работы со структурой данных “Двусвязный список”.

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

Для выполнения задания была написана программа и несколько подпрограмм.

Основная программа является кейсом-меню, для действий пользователя.

Была реализована возможность добавить элементы с начала, с конца, после и до элемента. Реализация была достигнута при помощи подпрограмм

Так же в данной программе предусмотрена возможность удаления и очистки списка.

Данная структура использует статическую память и реализована при помощи массива.

Для добавления элементов с начала или в середину использовался «сдвиг».

При добавлении в конец элемент просто ставился в следующую не заполненную ячейку.

4. Схема алгоритма с комментариями

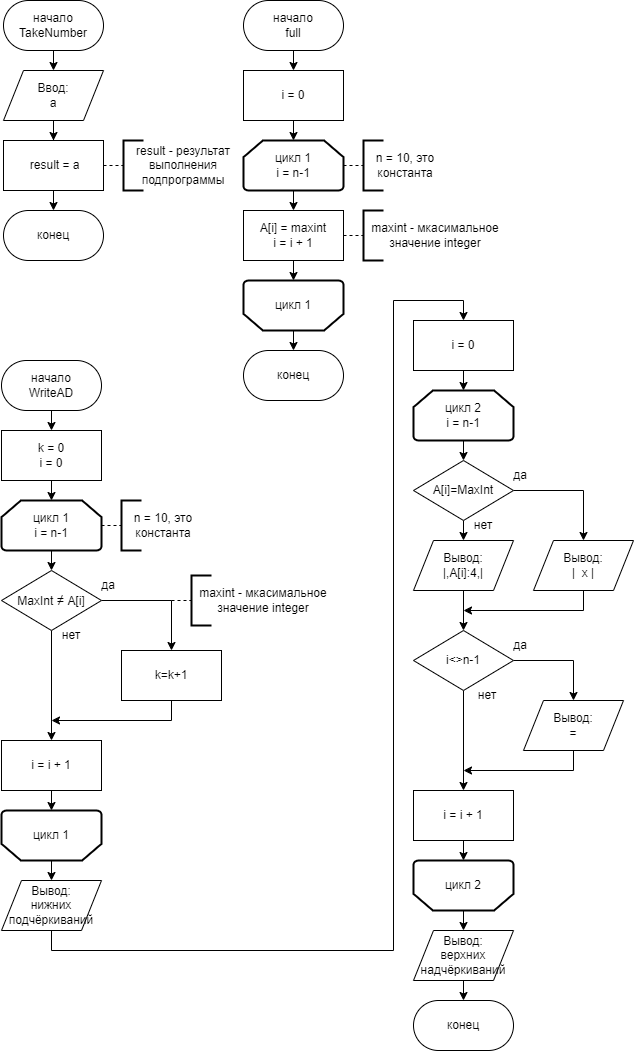


Рисунок 1 − Подпрограммы TakeNumber full WriteAD

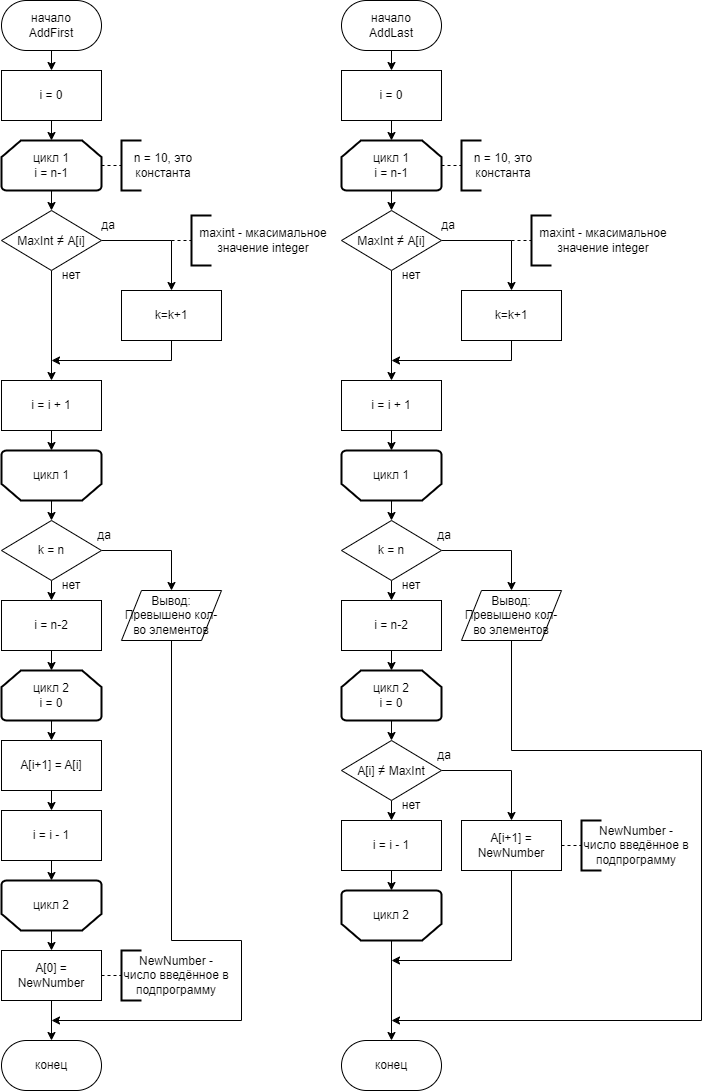


Рисунок 2 − Подпрограммы AddFirst AddLast

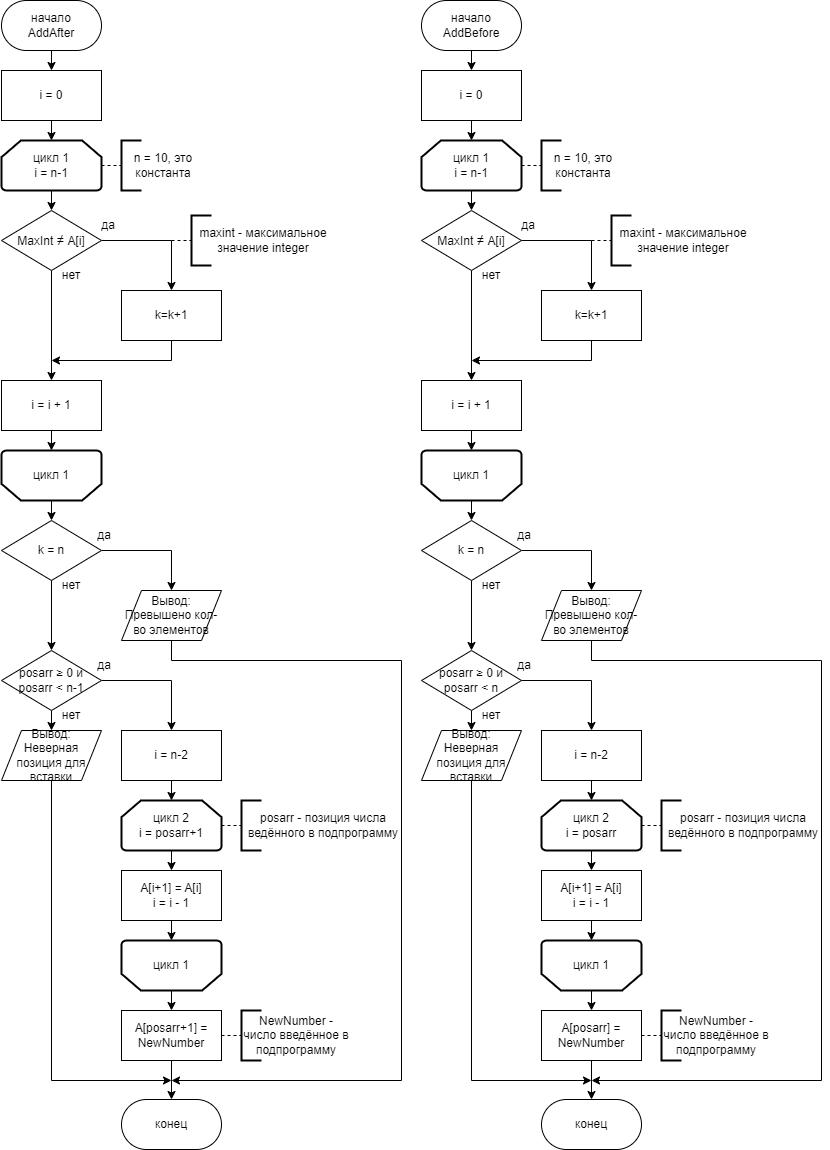


Рисунок 3 − Подпрограммы AddAfter AddBefore

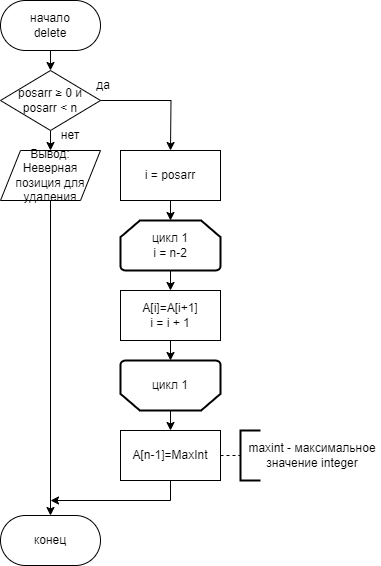


Рисунок 4 − Подпрограмма delete

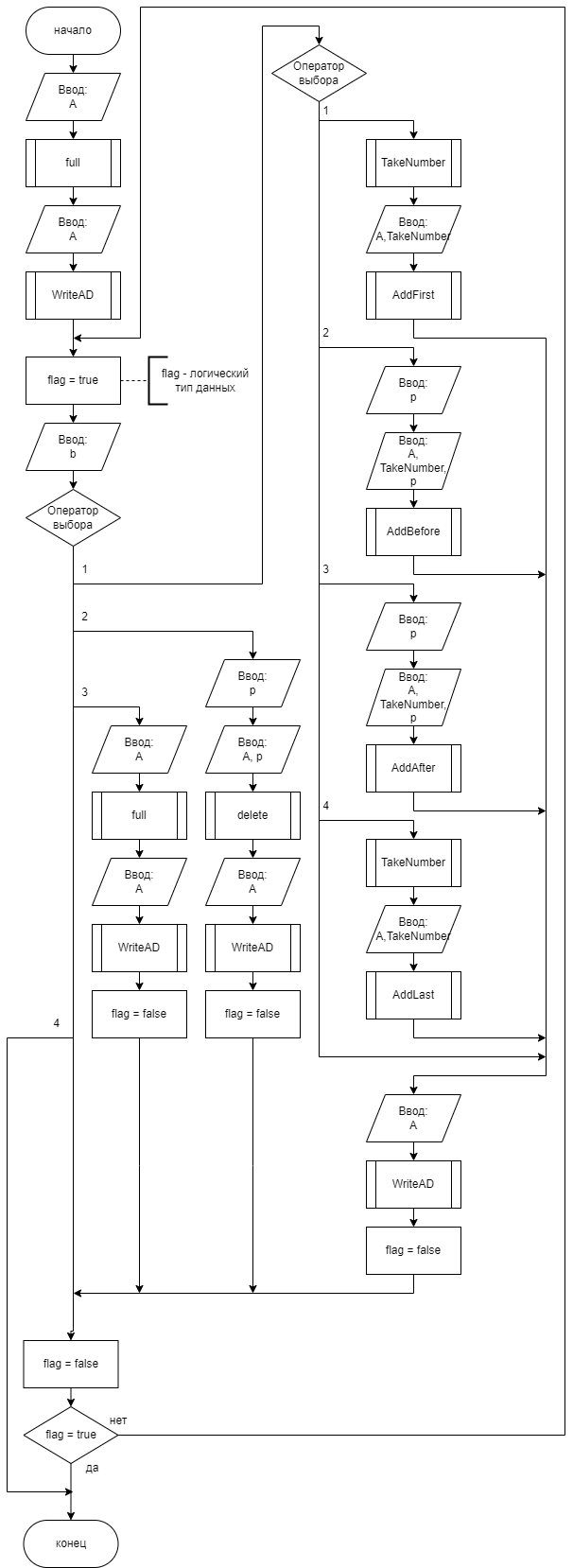


Рисунок 5 − Тело программы

5. Код программы

**program** zad2;

**uses** crt;

**const**

n = 10;

**type** arr = **array** [0..n-1] **of** integer;

**var** b,p: integer;

A:arr;flag: boolean;

**function** TakeNumber:integer;

**var** a:integer;

**begin**

read(a);

result:=a;

**end**;

**procedure** full (**var** A:arr);

**var** i:integer;

**begin**

**for** i:=0 **to** n-1 **do** A[i]:=maxint;

**end**;

**procedure** WriteAD (**var** A:arr);

**var** i,k:integer;

**begin**

k:=0;

**for** i:=0 **to** n-1 **do if** MaxInt<>A[i] **then** k+=1;

**if** (k<>n) **then** Write(' \_\_\_\_ ');

Write(' \_\_\_\_ \_\_\_\_ \_\_\_\_ \_\_\_\_ \_\_\_\_ \_\_\_\_ \_\_\_\_ \_\_\_\_ \_\_\_\_ \_\_\_\_ ');

**if** (k<>n) **then** Write('\_\_\_\_');

Writeln;

**if** (k<>n) **then** Write('| x | = ');

**for** i:=0 **to** n-1 **do begin**

**if** A[i]=MaxInt **then** Write('| x |')

**else** Write('|',A[i]:4,'|');

**if** i<>n-1 **then** Write(' = ');

**end**;

**if** (k<>n) **then** Write(' = | x |');

Writeln;

**if** (k<>n) **then** Write(' ‾‾‾‾ ');

Write(' ‾‾‾‾ ‾‾‾‾ ‾‾‾‾ ‾‾‾‾ ‾‾‾‾ ‾‾‾‾ ‾‾‾‾ ‾‾‾‾ ‾‾‾‾ ‾‾‾‾ ');

**if** (k<>n) **then** Write('‾‾‾‾');

Writeln;

**end**;

**procedure** AddFirst ( **var** A:arr; NewNumber: integer);

**var** i, k :integer;

**begin**

**for** i:=0 **to** n-1 **do if** MaxInt<>A[i] **then** k+=1;

**if** k=n **then begin**

Writeln('Превышено кол-во элементов');

**exit**;

**end**

**else begin**

**for** i:=n-2 **downto** 0 **do** A[i+1]:=A[i]; /// 1 2 3 4 5 6 7 8

A[0]:=NewNumber;

**end**;

**end**;

**procedure** AddLast ( **var** A:arr; NewNumber: integer);

**var** i, k :integer;

**begin**

**for** i:=0 **to** n-1 **do if** MaxInt<>A[i] **then** k+=1;

**if** k=n **then begin**

Writeln('Превышено кол-во элементов');

**exit**;

**end**

**else begin**

**for** i:=n-2 **downto** 0 **do**

**if** A[i]<>MaxInt **then**

**begin**

A[i+1]:=NewNumber;

**Break**;

**end**;

**end**;

**end**;

**procedure** AddBefore ( **var** A:arr; NewNumber: integer; posarr:integer);

**var** k,i:integer;

**begin**

**for** i:=0 **to** n-1 **do if** MaxInt<>A[i] **then** k+=1;

**if** k=n **then begin**

Writeln('Превышено кол-во элементов');

**exit**;

**end**

**else begin**

**if** (posarr >= 0) **and** (posarr < n) **then**

**begin**

**for** i:=n-2 **downto** posarr **do** A[i+1]:=A[i];

A[posarr]:=NewNumber;

**end**

**else** Writeln('Неверная позиция для вставки');

**end**;

**end**;

**procedure** AddAfter ( **var** A:arr; NewNumber: integer; posarr:integer);

**var** k,i:integer;

**begin**

**for** i:=0 **to** n-1 **do if** MaxInt<>A[i] **then** k+=1;

**if** k=n **then begin**

Writeln('Превышено кол-во элементов');

**exit**;

**end**

**else begin**

**if** (posarr>=0) **and** (posarr<n-1) **then**

**begin**

**for** i:=n-2 **downto** posarr+1 **do** A[i+1] := A[i];

A[posarr + 1] := NewNumber;

**end**

**else** Writeln('Неверная позиция для вставки');

**end**;

**end**;

**procedure** delete(**var** A: arr; posarr: integer);

**var** i: integer;

**begin**

**if** (posarr >= 0) **and** (posarr < n) **then**

**begin**

**for** i:=posarr **to** n-2 **do** A[i]:=A[i+1];

A[n-1]:=MaxInt;

**end**

**else** Writeln('Неверная позиция для удаления');

**end**;

**begin**

full(A);

Writeln('Двусвязный список');

WriteAD(A);

**repeat**

flag:=true;

writeln('Что вы хотите сделать?');

writeln('1: Добавить элемент');

writeln('2: Удалить элемент');

writeln('3: Очистить список');

writeln('4: Закрыть приложение');

readln(b);

**case** b **of**

1:**begin**

writeln('Куда вы хотите добавить элемент?');

writeln('1: В начало');

writeln('2: Перед...');

writeln('3: После...');

writeln('4: В конец ');

readln(b);

**case** b **of**

1: **begin**

writeln('Введите число');

AddFirst(A,TakeNumber);

**end**;

2: **begin**

writeln('Введите номер элемента перед которым нужно добавить новый элемент');

read(p);

writeln('Введите число');

AddBefore(A,TakeNumber,p)

**end**;

3: **begin**

writeln('Введите номер элемента после которого нужно добавить новый элемент');

read(p);

writeln('Введите число');

AddAfter(A,TakeNumber,p)

**end**;

4: **begin**

writeln('Введите число');

AddLast(A,TakeNumber);

**end**;

**end**;

writeln('Элемент успешно добавлено');

WriteAD(A);

flag:=false;

**end**;

2:**begin**

writeln('Введите номер элемента который нужно удалить');

read(p);

delete(A,p);

writeln('Элемент успешно удалено');

WriteAD(A);

flag:=false;

**end**;

3:**begin**

full(A);

WriteAD(A);

flag:=false;

**end**;

4:**Exit**;

**end**;

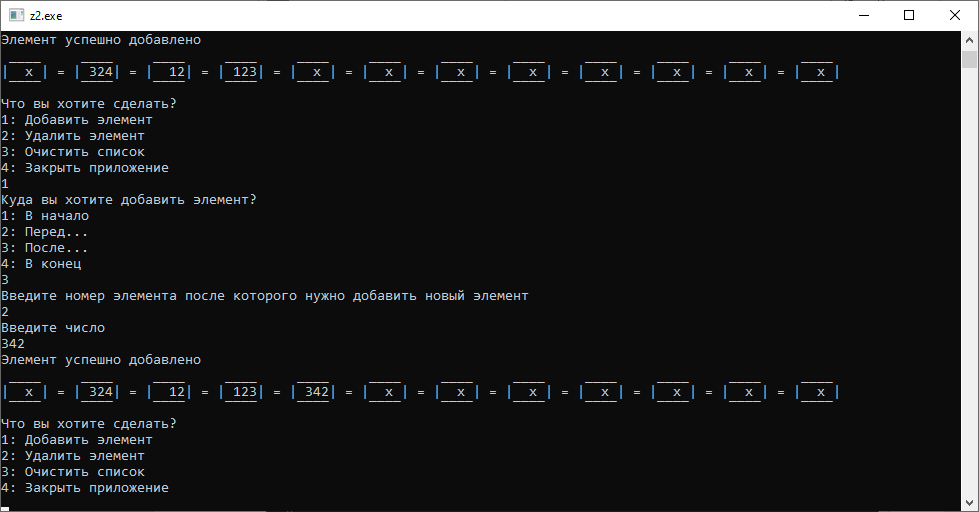
flag:=false;

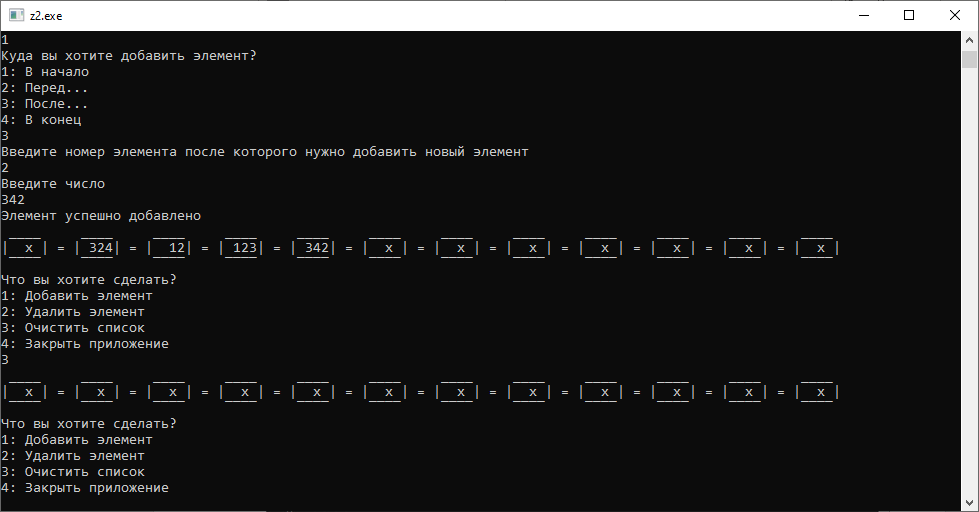
**until** flag=true;

**end**.

6. Результат выполнения программы







6. Вывод

Цель работы была достигнута.

В процессе возникали сложности с реализацией на статической памяти

Было реализовано наглядное отображение списка

Осталась проблема с вставлением элементов до или после какого-то элемента. Элемент будет «подвешен в воздухе».

Для имитации параметра nil был использован maxint.

Так же была реализована полная очистка списка в случае возникновения ошибки.

Для использования массивов в процедурах был создан тип arr.

Более о работе сказать нечего.