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 НА ОСНОВЕ СТАТИЧЕСКОЙ ПАМЯТИ»**

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

Выполнила: студентка учебной группы

ИСПк-203-52-00

Бочкарёва Виктория Дмитриевна

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2024

**Цель работы**

Изучение принципов работы с базовыми структурами данных, получение навыков организации case-меню.

**Формулировка задания**

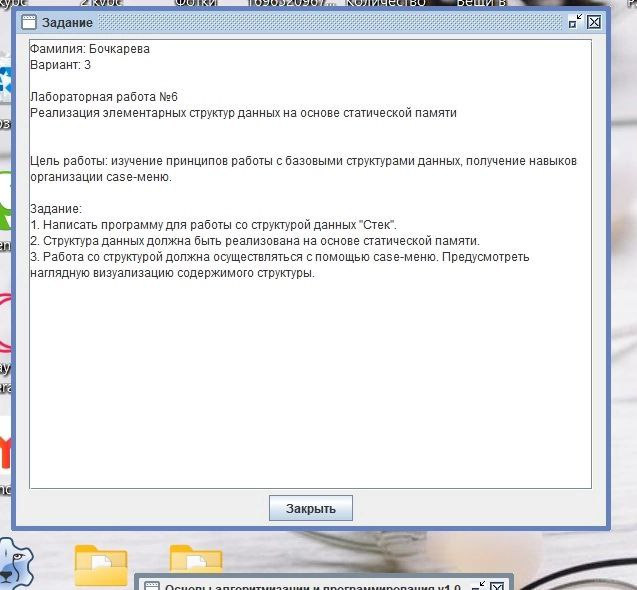


Рисунок 1 – Формулировка задания

**Описание алгоритма**

Алгоритм представляет собой case-меню с возможностью выбора реализации стека: динамическая/статическая. После выбора реализации дается выбор совершаемых действий над стеком: добавить элемент, взять элемент, вывести стек, очистить стек. В случае с динамической реализацией стек основывается на списках с указателями. В статической – на массиве.

Для работы со стеком созданы функции и процедуры Pop, Push, AddToStack, Print, ClearStack для статической и для динамической реализаций.

Во избежание громоздкости кода, все подпрограммы хранятся в модулях: Dynamic\_Stack для динамической и Static\_Stack для статической.

**Схема алгоритма**

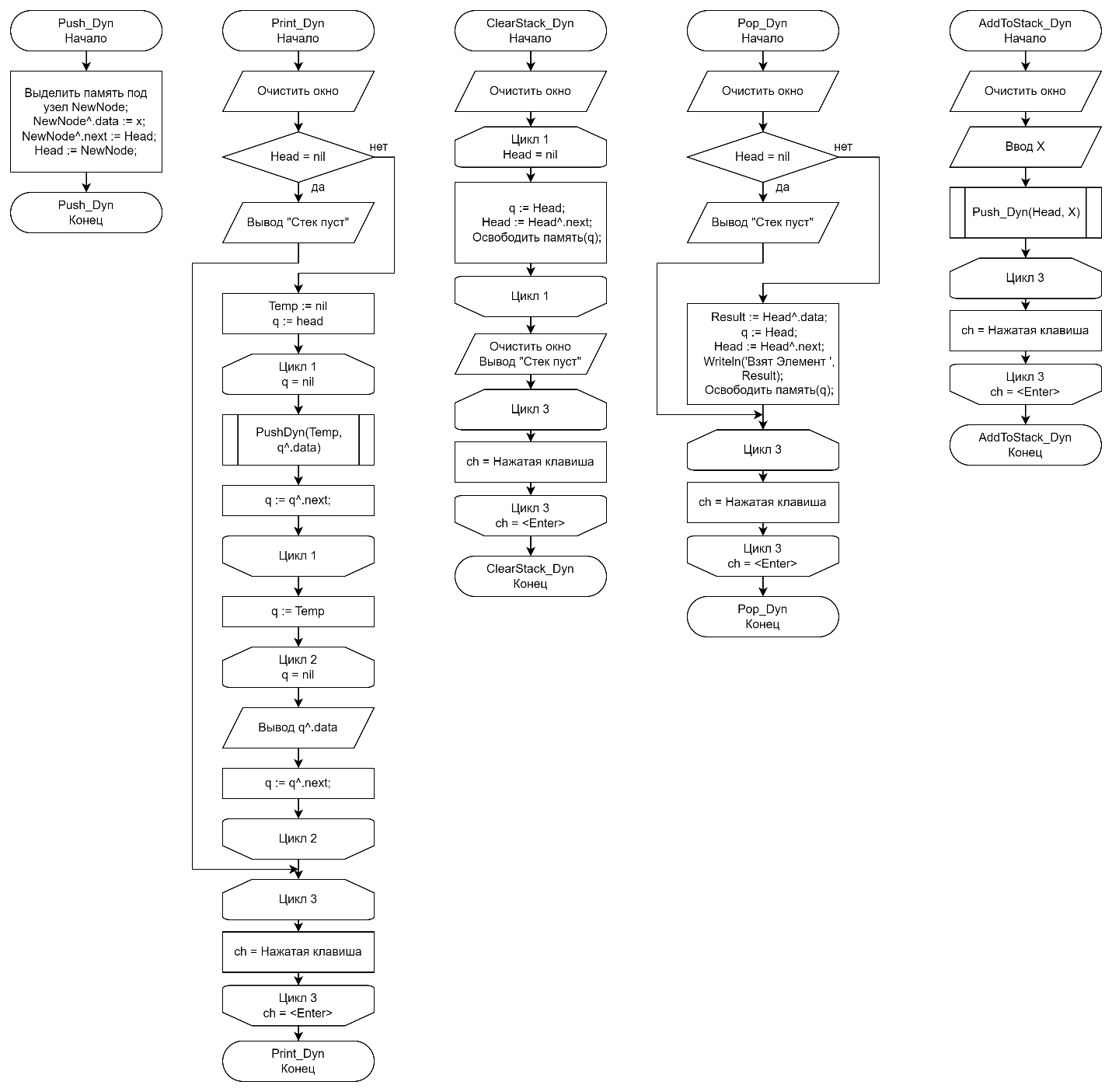


Рисунок 2 – Подпрограммы модуля Dynamic\_Stack

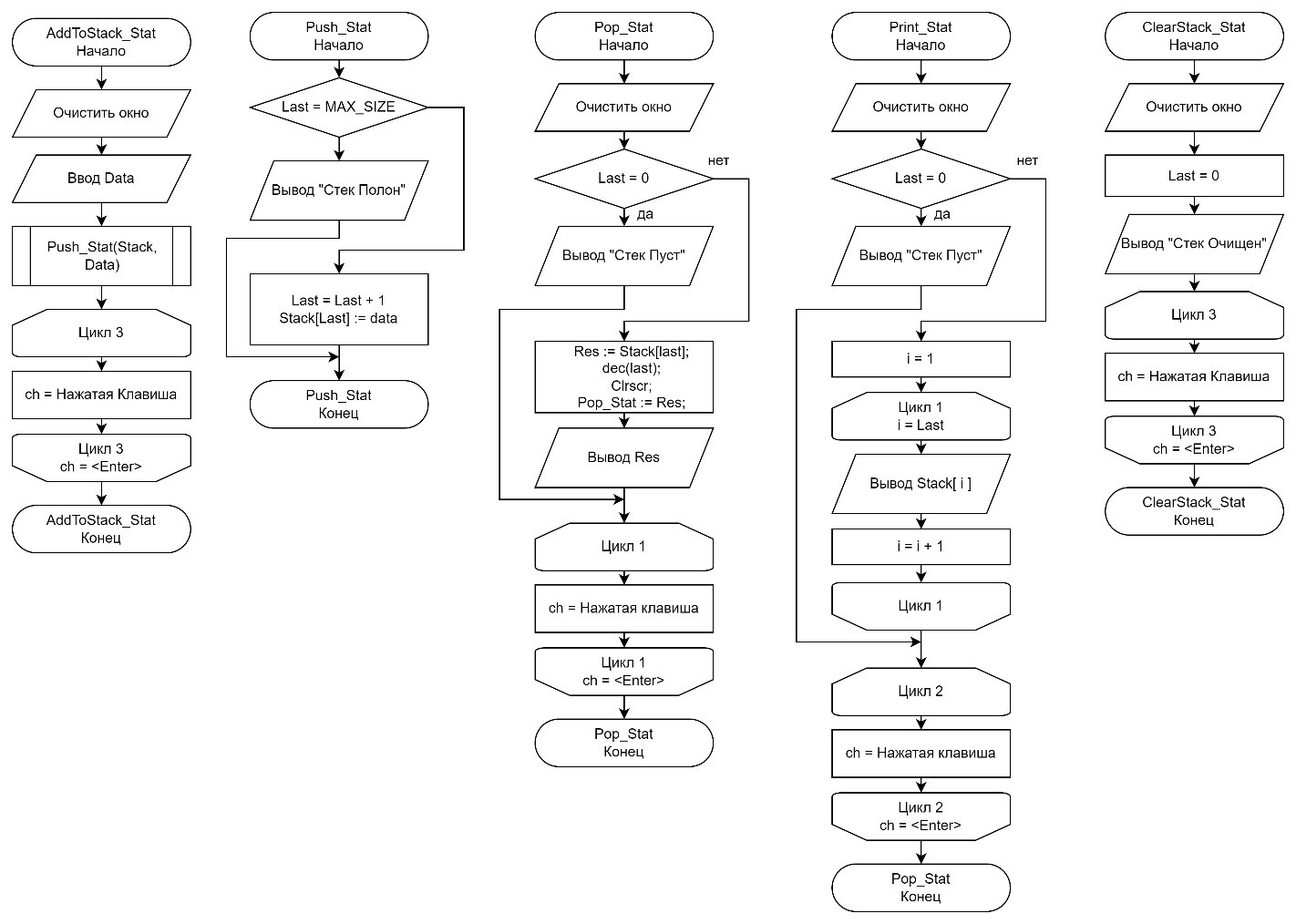


Рисунок 3 – Подпрограммы модуля Static\_Stack

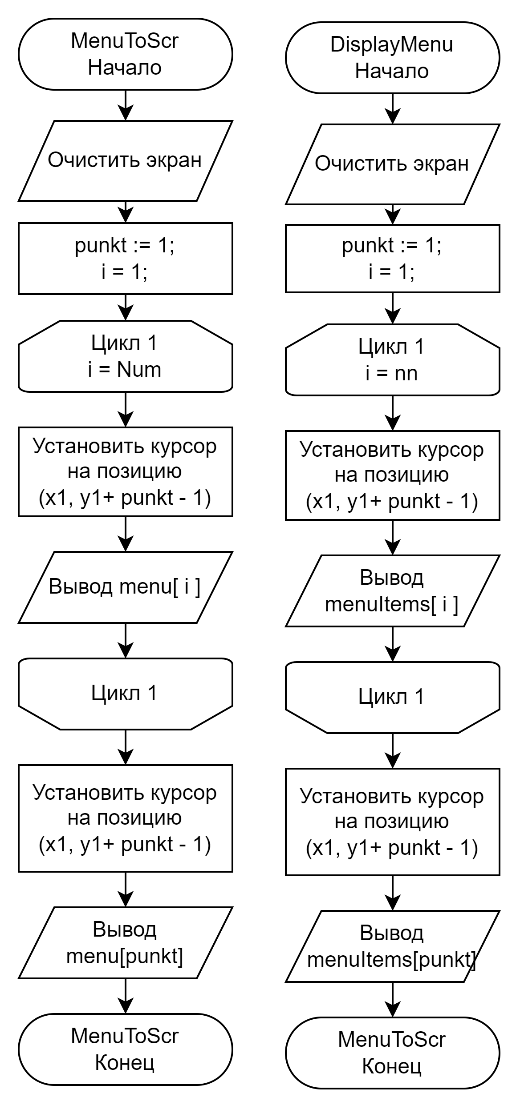


Рисунок 4 – Процедуры для вывода пунктов на экран

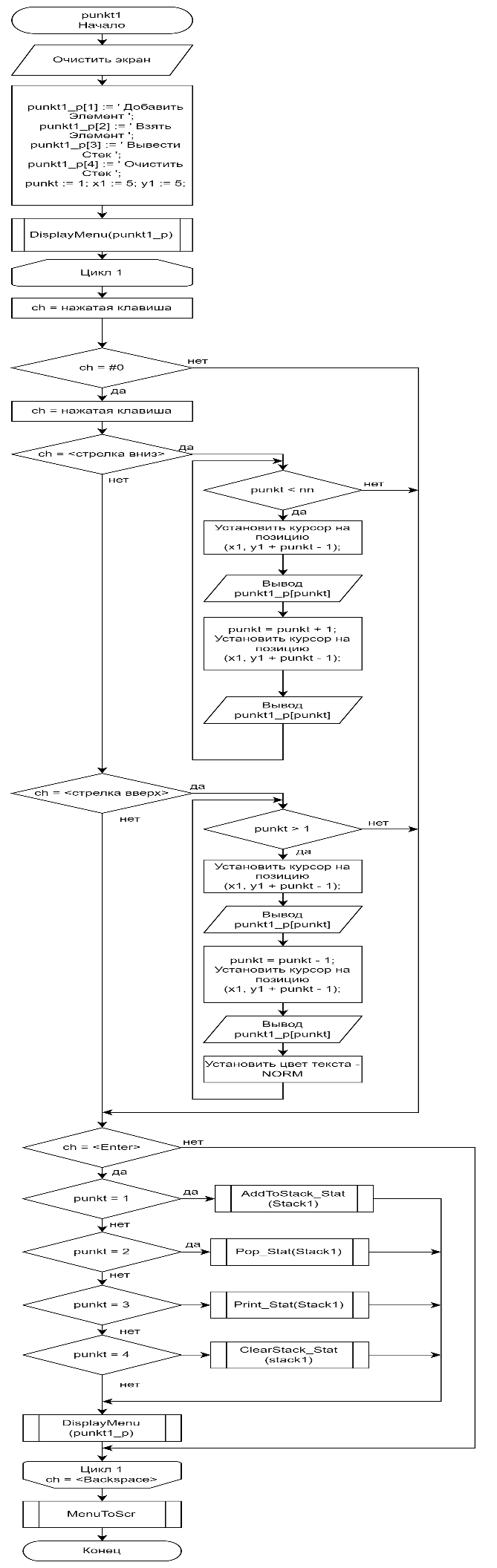


Рисунок 5 – Процедура для пункта 1

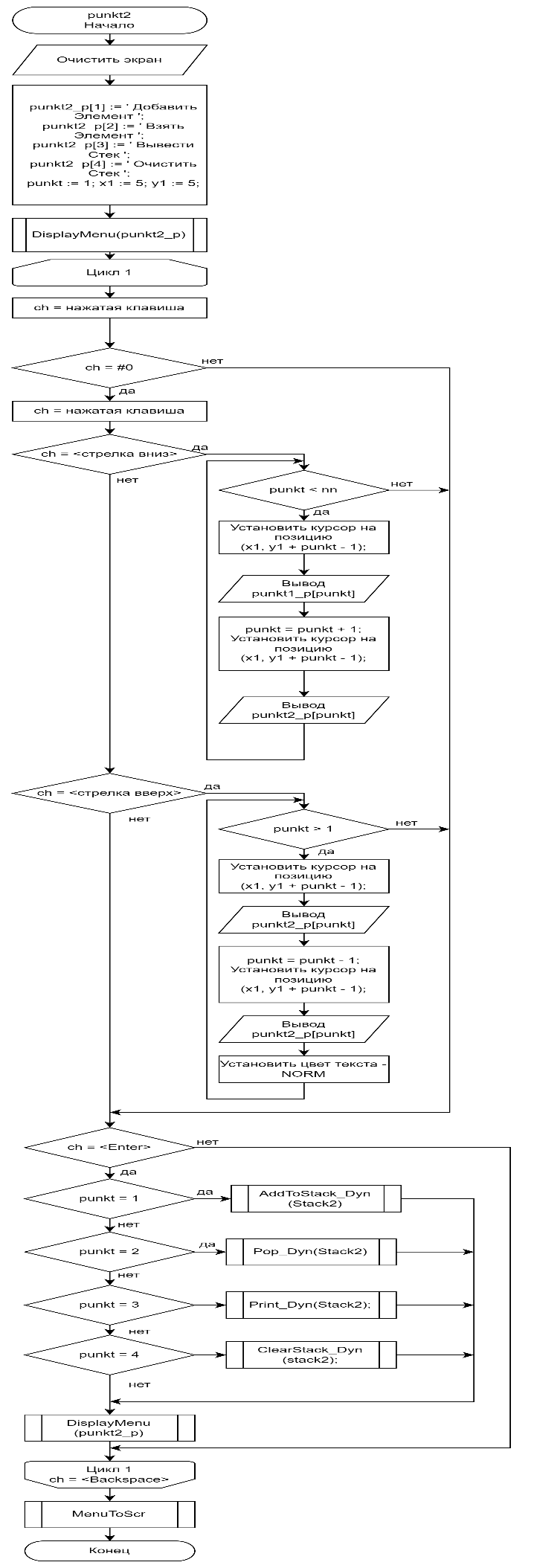


Рисунок 6 – Процедура для пункта 2

**Код программы**

Модуль Static\_Stack:

**unit** Static\_Stack;

**interface**

**Uses** Crt;

**const**

MAX\_SIZE = 100;

NORM = 8;

SEL = 13;

**type**

tStack = **array** [1..MAX\_SIZE] **of** Integer;

**var**

last: Integer;

Stack: tStack;

ch: char;

**procedure** Push\_Stat(**var** Stack: tStack; data: integer);

**procedure** Print\_Stat(Stack: tStack);

**procedure** addToStack\_Stat(**var** Stack: tStack);

**procedure** clearStack\_Stat(**var** Stack: tStack);

**function** Pop\_Stat(**var** Stack: tStack): integer;

**implementation**

**procedure** Push\_Stat(**var** Stack: tStack; data: integer);

**begin**

**if** last = MAX\_SIZE **then**

**begin**

textcolor(red);

Writeln('Стек Полон');

textcolor(norm);

**end**

**else begin**

inc(last);

Stack[last] := data;

**end**;

**end**;

**function** Pop\_Stat(**var** Stack: tStack): integer;

**var**

Res: integer;

**begin**

ClrScr;

**if** last = 0 **then**

**begin**

textcolor(red);

Writeln('Стек Пуст');

textcolor(norm);

**end**

**else begin**

Res := Stack[last];

dec(last);

Clrscr;

Pop\_Stat := Res;

Writeln('Взят Элемент ', Res);

**end**;

**repeat**

ch := readkey

**until** ch = #13;

**end**;

**procedure** Print\_Stat(Stack: tStack);

**var**

i: Integer;

**begin**

clrscr;

**if** last = 0 **then**

**begin**

textcolor(red);

WriteLn('Стек Пуст');

textcolor(norm);

**end**

**else begin**

**for** i := 1 **to** last **do**

Write(Stack[i], ' ');

**end**;

**repeat**

ch := readkey;

**until** ch = #13;

**end**;

**procedure** addToStack\_Stat(**var** Stack: tStack);

**var**

data: integer;

**begin**

Clrscr;

textcolor(green);

Write('Введите значение добавляемого элемента: ');

ReadLn(data);

Push\_Stat(Stack, data);

textcolor(norm);

**repeat**

ch := ReadKey;

**until** ch = #13;

**end**;

**procedure** clearStack\_Stat(**var** Stack: tStack);

**begin**

last := 0;

Clrscr;

textcolor(red);

WriteLn('Стек очищен');

textcolor(norm);

**repeat**

ch := ReadKey;

**until** ch = #13;

**end**;

**end.**

Модуль Dynamic\_Stack:

**unit** Dynamic\_Stack;

**interface**

**Uses** Crt;

**const**

NORM = 8;

SEL = 13;

**type**

PNode = ^Node;{ указатель на узел }

Node = **record**

data: integer;

next: PNode; { указатель на след. элемент }

**end**;

**var**

S: PNode;

x: integer;

ch: char;

**procedure** Push\_Dyn(**var** Head: PNode; x: integer);

**procedure** Print\_Dyn(**var** Head: PNode);

**procedure** addToStack\_Dyn(**var** Head: PNode);

**procedure** ClearStack\_Dyn(**var** Head: PNode);

**function** Pop\_Dyn(**var** Head: PNode): integer;

**implementation**

**procedure** Push\_Dyn(**var** Head: PNode; x: integer);

**var**

NewNode: PNode;

**begin**

**New**(NewNode); { выделение памяти }

NewNode^.data := x; { запись символа }

NewNode^.next := Head; { сделать первым узлом }

Head := NewNode;

**end**;

**procedure** Print\_Dyn(**var** Head: PNode);

**var**

Temp, q: PNode;

**begin**

ClrScr;

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

**begin**

textcolor(red);

Writeln('Стек Пуст');

textcolor(norm);

**end**

**else**

**begin**

Temp := nil;

// Переносим элементы из исходного стека во временный стек

q := Head;

**while** q <> nil **do**

**begin**

Push\_Dyn(Temp, q^.data);

q := q^.next;

**end**;

// Выводим элементы из временного стека

q := Temp;

**while** q <> nil **do**

**begin**

write(q^.data, ' ');

q := q^.next;

**end**;

**end**;

**repeat**

ch := readkey;

**until** ch = #13;

**end**;

**procedure** ClearStack\_Dyn(**var** Head: PNode);

**var**

q: PNode;

**begin**

**while** Head <> nil **do**

**begin**

q := Head;

Head := Head^.next;

Dispose(q);

**end**;

Clrscr;

textcolor(red);

WriteLn('Стек очищен');

textcolor(norm);

**repeat**

ch := ReadKey;

**until** ch = #13;

**end**;

**function** Pop\_Dyn(**var** Head: PNode): integer;

**begin**

ClrScr;

**var** q: PNode;

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

**begin**

textcolor(red);

Writeln('Стек Пуст');

textcolor(norm);

**end**

**else**

**begin**

Result := Head^.data; { берем верхний символ }

q := Head; { запоминаем вершину }

Head := Head^.next; { удаляем вершину }

Writeln('Взят Элемент ', Result);

Dispose(q); { удаление из памяти }

**end**;

**repeat**

ch := readkey

**until** ch = #13;

**end**;

**procedure** AddToStack\_Dyn(**var** Head: PNode);

**begin**

Clrscr;

textcolor(green);

Write('Введите значение добавляемого элемента: ');

ReadLn(x);

Push\_Dyn(Head, x);

textcolor(norm);

**repeat**

ch := ReadKey;

**until** ch = #13;

**end**;

**end**.

Код программы:

**program** DKR\_6;

**uses** Crt, Static\_Stack, Dynamic\_Stack;

**const**

MAX\_SIZE = 100;

NORM = 8; {серый}

SEL = 13; {розовый}

Num = 3;

nn = 4;

**type**

omg = **array**[1..nn] **of** string;

**var**

menu: **array**[1..Num] **of** string;{ названия пунктов меню }

punkt1\_p, punkt2\_p: omg;

Stack1: tStack;

Stack2: PNode;

punkt: integer;

ch: char;

x1, y1: integer;

**procedure** MenuToScr;

**var**

i: integer;

**begin**

ClrScr;

punkt := 1;

**for** i := 1 **to** Num **do**

**begin**

GoToXY(x1, y1 + i - 1);

write(menu[i]);

**end**;

TextColor(SEL);

GoToXY(x1, y1 + punkt - 1);

write(menu[punkt]);

TextColor(NORM);

**end**;

**procedure** DisplayMenu(**var** menuItems: omg; title: string);

**var**

i: integer;

**begin**

ClrScr;

punkt := 1;

textcolor(11);

writeln(title);

textcolor(norm);

**for** i := 1 **to** nn **do**

**begin**

GoToXY(x1, y1 + i - 1);

write(menuItems[i]);

**end**;

TextColor(SEL);

GoToXY(x1, y1 + punkt - 1);

write(menuItems[punkt]);

TextColor(NORM);

**end**;

**procedure** punkt1;

**begin**

ClrScr;

punkt1\_p[1] := ' Добавить Элемент ';

punkt1\_p[2] := ' Взять Элемент ';

punkt1\_p[3] := ' Вывести Стек ';

punkt1\_p[4] := ' Очистить Стек ';

punkt := 1; x1 := 5; y1 := 5;

TextColor(NORM);

DisplayMenu(punkt1\_p, 'Реализация СТЕКА с помощью статической структуры');

**repeat**

ch := ReadKey;

**if** ch = #0 **then begin**

ch := ReadKey;

**case** ch **of**

#40:{ стрелка вниз }

**if** punkt < 4 **then begin**

GoToXY(x1, y1 + punkt - 1); write(punkt1\_p[punkt]);

punkt := punkt + 1;

TextColor(SEl);

GoToXY(x1, y1 + punkt - 1); write(punkt1\_p[punkt]);

TextColor(NORM);

**end**;

#38:{ стрелка вверх }

**if** punkt > 1 **then begin**

GoToXY(x1, y1 + punkt - 1); write(punkt1\_p[punkt]);

punkt := punkt - 1;

TextColor(SEl);

GoToXY(x1, y1 + punkt - 1); write(punkt1\_p[punkt]);

TextColor(NORM);

**end**;

**end**;

**end**

**else**

**if** ch = #13 **then begin**{ нажата клавиша <Enter> }

**case** punkt **of**

1: AddToStack\_Stat(Stack1);

2: Pop\_Stat(Stack1);

3: Print\_Stat(Stack1);

4: ClearStack\_Stat(stack1);

**end**;

DisplayMenu(punkt1\_p, 'Реализация СТЕКА с помощью статической структуры')

**end**;

**until** ch = #8;

menutoscr;

**end**;

**procedure** punkt2;

**begin**

clrscr;

punkt2\_p[1] := ' Добавить Элемент ';

punkt2\_p[2] := ' Взять Элемент ';

punkt2\_p[3] := ' Вывести Стек ';

punkt2\_p[4] := ' Очистить Стек ';

punkt := 1; x1 := 5; y1 := 5;

TextColor(NORM);

DisplayMenu(punkt2\_p, 'Реализация СТЕКА с помощью ДиНаМиЧеСкОй структуры');

**repeat**

ch := ReadKey;

**if** ch = #0 **then begin**

ch := ReadKey;

**case** ch **of**

#40:{ стрелка вниз }

**if** punkt < 4 **then begin**

GoToXY(x1, y1 + punkt - 1); write(punkt2\_p[punkt]);

punkt := punkt + 1;

TextColor(SEl);

GoToXY(x1, y1 + punkt - 1); write(punkt2\_p[punkt]);

TextColor(NORM);

**end**;

#38:{ стрелка вверх }

**if** punkt > 1 **then begin**

GoToXY(x1, y1 + punkt - 1); write(punkt2\_p[punkt]);

punkt := punkt - 1;

TextColor(SEl);

GoToXY(x1, y1 + punkt - 1); write(punkt2\_p[punkt]);

TextColor(NORM);

**end**;

**end**;

**end**

**else**

**if** ch = #13 **then begin**{ нажата клавиша <Enter> }

**case** punkt **of**

1: AddToStack\_Dyn(Stack2);

2: Pop\_Dyn(Stack2);

3: Print\_Dyn(Stack2);

4: ClearStack\_Dyn(stack2);

**end**;

DisplayMenu(punkt2\_p, 'Реализация СТЕКА с помощью ДиНаМиЧеСкОй структуры');

**end**;

**until** ch = #8;

menutoscr;

**end**;

**begin**

clrscr;

Writeln('STACK');

menu[1] := ' Статическая структура ';

menu[2] := ' Динамическая структура ';

menu[3] := ' Выход ';

punkt := 1; x1 := 5; y1 := 5;

TextColor(NORM);

MenuToScr;

**repeat**

ch := ReadKey;

**if** ch = #0 **then begin**

ch := ReadKey;

**case** ch **of**

#40:{ стрелка вниз }

**if** punkt < Num **then begin**

GoToXY(x1, y1 + punkt - 1); write(menu[punkt]);

punkt := punkt + 1;

TextColor(SEl);

GoToXY(x1, y1 + punkt - 1); write(menu[punkt]);

TextColor(NORM);

**end**;

#38:{ стрелка вверх }

**if** punkt > 1 **then begin**

GoToXY(x1, y1 + punkt - 1); write(menu[punkt]);

punkt := punkt - 1;

TextColor(SEl);

GoToXY(x1, y1 + punkt - 1); write(menu[punkt]);

TextColor(NORM);

**end**;

**end**;

**end**

**else**

**if** ch = #13 **then begin**{ нажата клавиша <Enter> }

**case** punkt **of**

1: punkt1;

2: punkt2;

3: ch := #27;{ выход }

**end**;

**end**;

**until** ch = #27;{ 27 - код <Esc> }

**end**.

**Результат выполнения программы**

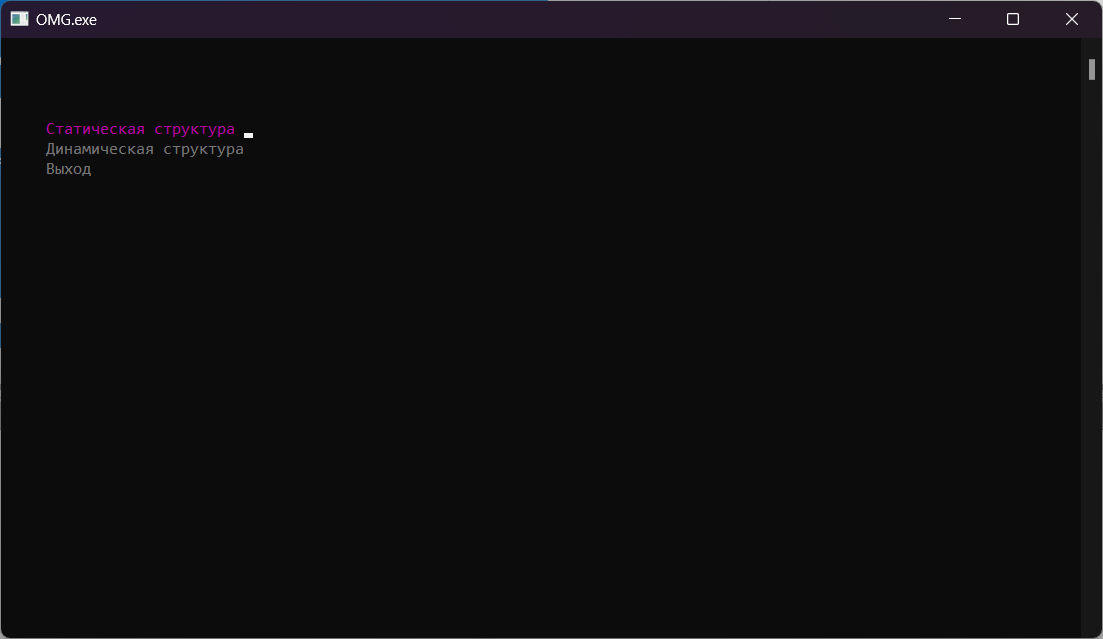


Рисунок 7 – Результат выполнения программы

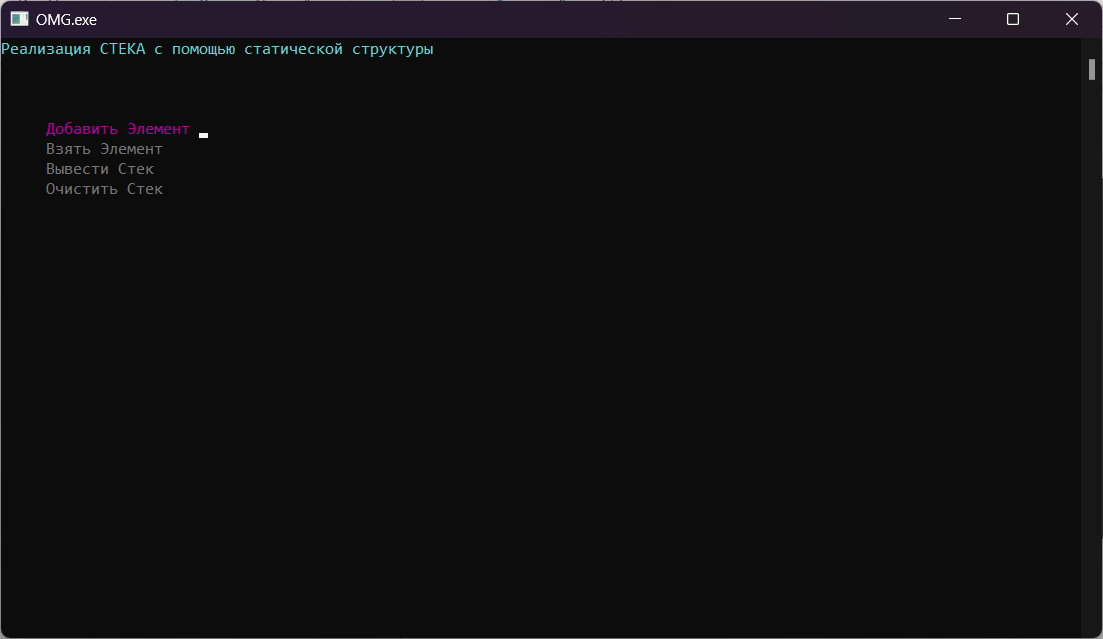


Рисунок 8 – Результат выполнения программы

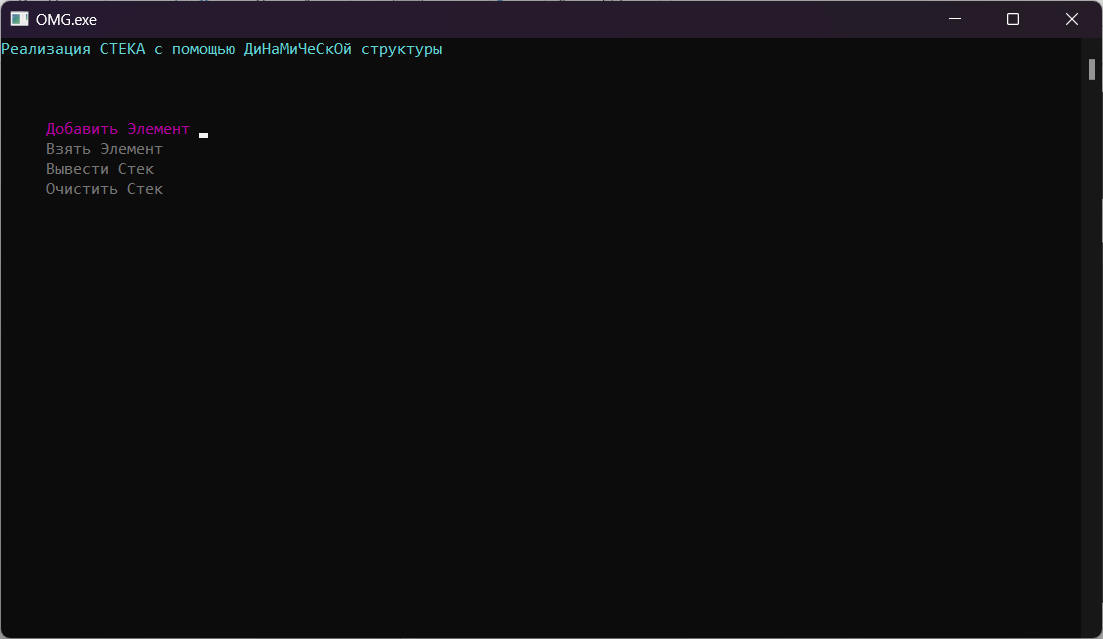


Рисунок 9 – Результат выполнения программы

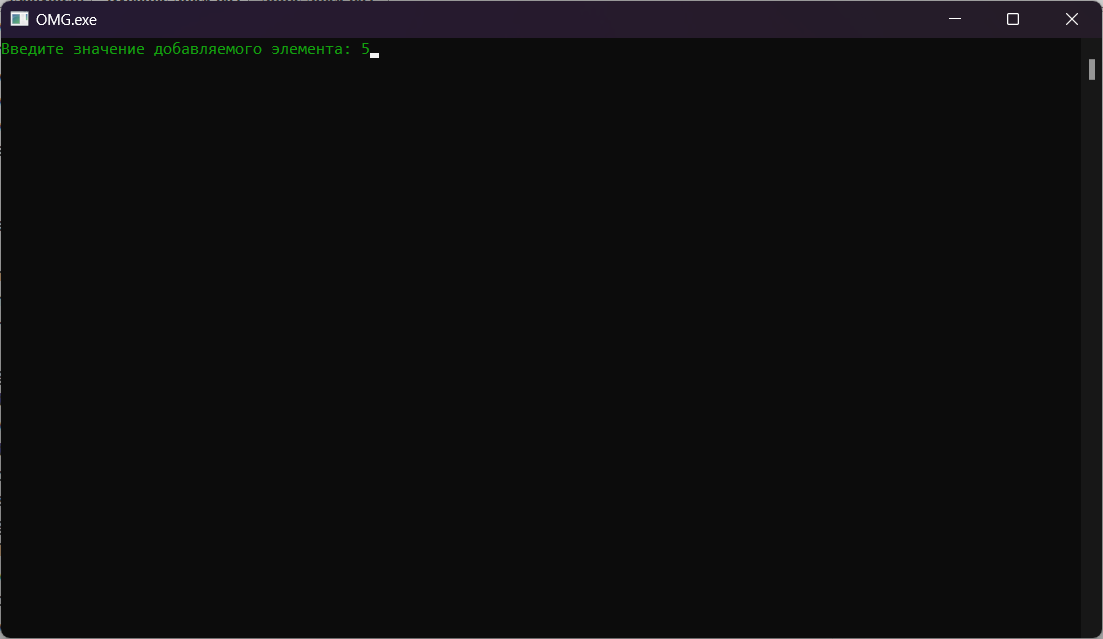


Рисунок 10 – Ввод элемента в стек

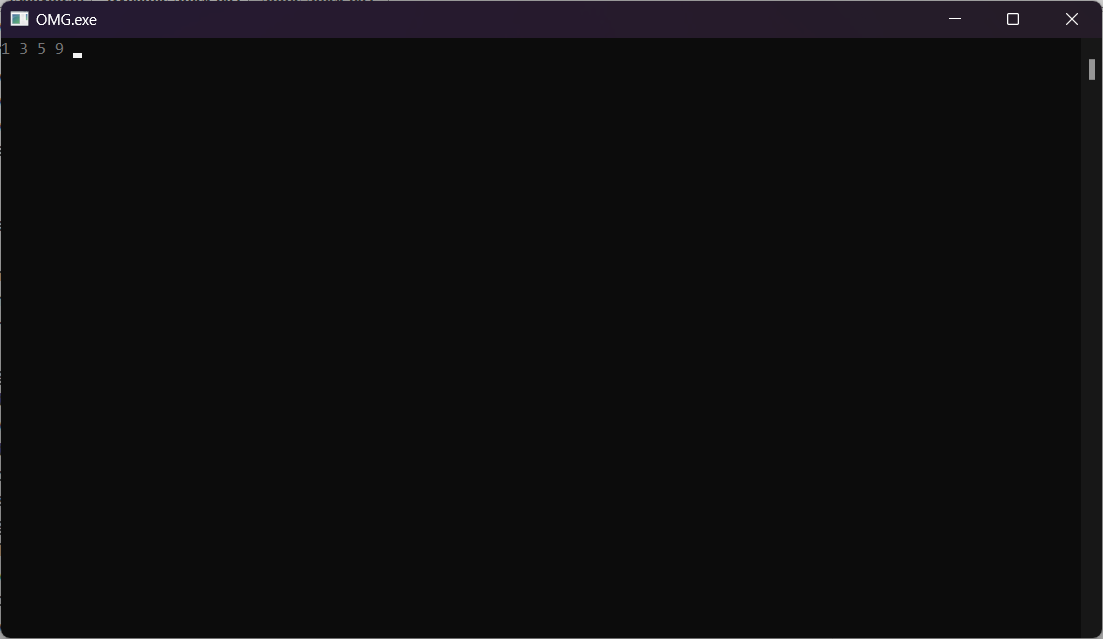


Рисунок 11 – Вывод стека

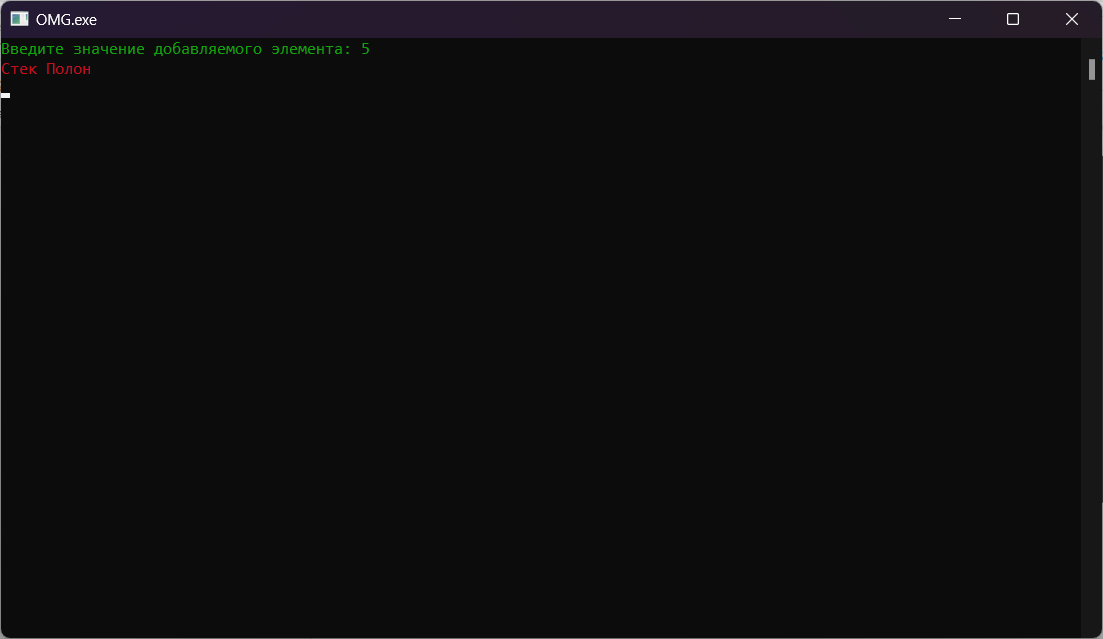


Рисунок 12 – Переполнение стека

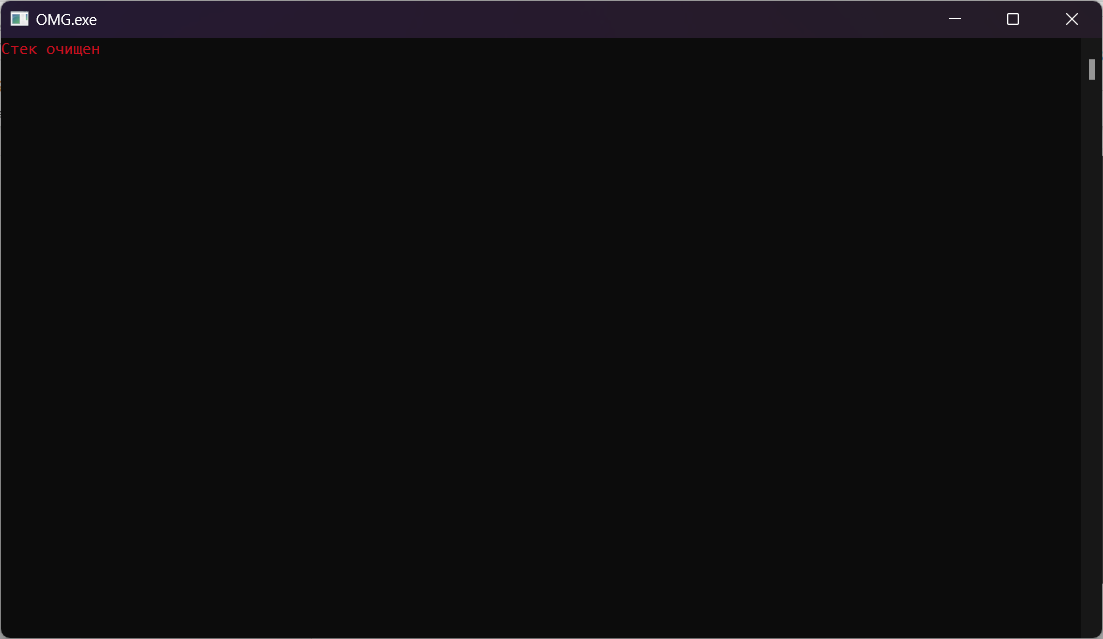


Рисунок 13 – Очистка стека

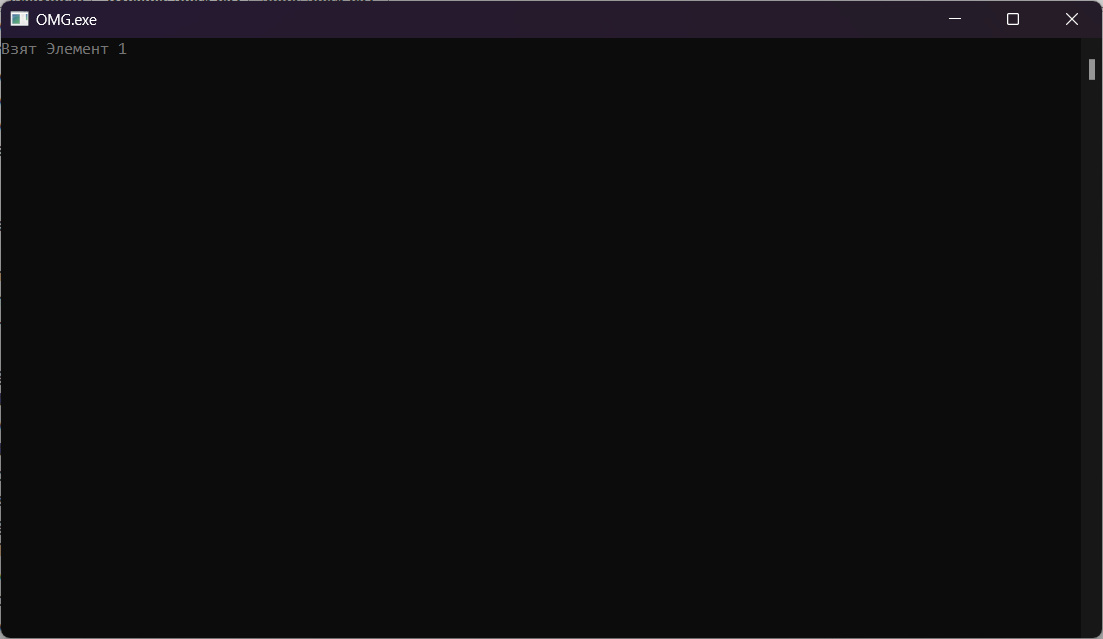


Рисунок 14 – Взять элемент из стека



Рисунок 15 – Вывод и взятие элемента из стека, если стек пуст

**Вывод**

В результате выполнения домашней контрольной работы поставленная цель была достигнута. Были изучены принципы работы с базовыми структурами данных, получены навыки организации case-меню.

В ходе ДКР была разработана программа для реализации структуры данных – стек – с помощью статических и динамических структур. Были также развиты навыки работы с собственными модулями. Программа реализует два способа работы со стеками. Пользователь может выбрать какую структуру он хочет использовать для реализации стека и какие действия он хочет над ним провести.

Таким образом, все поставленные цели и задачи были успешно достигнуты и выполнены. Получены удовлетворяющие результаты.