МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

**«Вятский государственный университет»**

**(ФГБОУ ВО «ВятГУ»)**

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

Реализация элементарных структур данных на основе динамической памяти

Отчет

Лабораторная работа №6 по дисциплине

«Программирование»

Выполнил студент группы ИВТ-11 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Кудяшев Я.Ю./

Проверил преподаватель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Чистяков Г.А./

Киров 2020

**Цель работы**

Изучение структуры и принципов организации программных модулей, закрепление навыков работы с динамической памятью, получение базовых навыков организации работы в режиме командной строки.

**Задание**

1. Написать программу для работы со структурой данных «Кольцевой список».
2. Структура данных должна быть реализована на основе динамической памяти.
3. Структура данных (поля и методы) должна быть описана в отдельном модуле.
4. Работа сос структурой должна осуществляться в режиме командной строки (с реализацией автодополнения и истории команд). Предусмотреть наглядную визуализацию содержимого структуры.

**Программа**

**unit** book;

**interface**

**uses** crt;

**type**

Tinf=integer;

List=^TList;

TList=**record**

data:TInf;

next:List;

**end**;

**var**

SpisNach,

tmpl:List;

znach,a,b:int64;

ch:char;

x,j,i:integer;

s:string;

c:char;

m,k:tlist;

up,dn,au:boolean;

comand:**array of** string;

perAupn,updn,com:integer;

**procedure** AddElem(**var** nach:List;znach1:TInf);

**procedure** Print(spis1:List);

**Procedure** DelElem(**var** spis1:List;tmp:List);

**Procedure** Free(spis1:List);

**procedure** DelElemZnach(**var** Spis1:List;znach1:TInf);

**procedure** IncSort(nach:list);

**Procedure** DelElemPos(**var** spis1:List;posi:integer);

**procedure** DecSort(nach:list);

**implementation**

**procedure** AddElem(**var** nach:List;znach1:TInf);

**var**

tmp,tmp1:List;

**begin**

**if** nach=nil **then**

**begin**

Getmem(nach,**SizeOf**(TList));

nach^.next:=nach;

tmp:=nach;

**end**

**else**

**begin**

tmp:=nach;

**while** tmp^.next<>nach **do**

tmp:=tmp^.next;

GetMem(tmp1,**SizeOf**(Tlist));

tmp1^.next:=nach;

tmp^.next:=tmp1;

tmp:=tmp1;

**end**;

**else**

tmp^.data:=znach1;

**end**;

**procedure** Print(spis1:List);

**var**

nach:List;

**begin**

**if** spis1=nil **then**

**begin**

writeln('List is empty');

readLn;

**exit**;

**end**;

nach:=spis1;

Write(spis1^.data, ' ');

spis1:=spis1^.next;

**while** spis1<>nach **do**

**begin**

Write(spis1^.data, ' ');

spis1:=spis1^.next;

**end**;

readln;

**end**;

**Procedure** DelElem(**var** spis1:List;tmp:List);

**var**

tmpi:List;

**begin**

**if** tmp=spis1 **then**

**begin**

tmpi:=tmp;

**while** tmpi^.next<>spis1 **do**

tmpi:=tmpi^.next;

**if** tmpi=spis1 **then**

**begin**

spis1^.next:=nil;

dispose(spis1);

spis1:=nil

**end**

**else**

**begin**

tmpi^.next:=tmp^.next;

spis1:=spis1^.next;

dispose(tmp)

**end**;

**end**

**else**

**begin**

tmpi:=spis1;

**while** tmpi^.next<>tmp **do**

tmpi:=tmpi^.next;

tmpi^.next:=tmp^.next;

dispose(tmp);

**end**;

**end**;

**procedure** DelElemZnach(**var** Spis1:List;znach1:TInf);

**var**

tmp:List;

**begin**

tmp:=spis1;

**if** znach1=tmp^.data **then**

**begin**

DelElem(spis1,tmp);

**exit**;

**end**;

tmp:=tmp^.next;

**while** tmp<>Spis1 **do**

**begin**

**if** tmp^.data=znach1 **then**

**begin**

DelElem(spis1,tmp);

**exit**

**end**;

tmp:=tmp^.next;

**end**;

**end**;

**Procedure** Free(spis1:List);

**var**

tmp,nach:List;

**begin**

**if** spis1=nil **then**

**exit**;

nach:=spis1;

tmp:=spis1;

spis1:=spis1^.next;

dispose(tmp);

**while** spis1<>nach **do**

**begin**

tmp:=spis1;

spis1:=spis1^.next;

FreeMem(tmp,**SizeOf**(Tlist));

**end**;

**end**;

**Procedure** DelElemPos(**var** spis1:List;posi:integer);

**var**

i:integer;

tmp:List;

**begin**

**if** spis1=nil **then exit**;

tmp:=spis1^.next;

i:=1;

**while** tmp<>spis1 **do**

**begin**

tmp:=tmp^.next;

inc(i)

**end**;

**begin**

i:=1;

tmp:=spis1;

**while** i<posi **do**

**begin**

tmp:=tmp^.next;

inc(i)

**end**;

DelElem(spis1,tmp);

**end**;

**end**;

**procedure** IncSort(nach:list);

**var**

tmp,rab:List;

tmps:Tinf;

**begin**

GetMem(tmp,**SizeOf**(Tlist));

rab:=nach;

**while** rab^.next<>nach **do**

**begin**

tmp:=rab^.next;

**while** tmp<>nach **do**

**begin**

**if** tmp^.data<rab^.data **then**

**begin**

tmps:=tmp^.data;

tmp^.data:=rab^.data;

rab^.data:=tmps

**end**;

tmp:=tmp^.next

**end**;

rab:=rab^.next

**end**

**end**;

**procedure** DecSort(nach:list);

**var**

tmp,rab:List;

tmps:Tinf;

**begin**

GetMem(tmp,**SizeOf**(Tlist));

rab:=nach;

**while** rab^.next<>nach **do**

**begin**

tmp:=rab^.next;

**while** tmp<>nach **do**

**begin**

**if** tmp^.data>rab^.data **then**

**begin**

tmps:=tmp^.data;

tmp^.data:=rab^.data;

rab^.data:=tmps

**end**;

tmp:=tmp^.next

**end**;

rab:=rab^.next

**end**

**end**;

**end**.

**Program** spisok;

**uses**

crt,book;

**var**

kk,aa,gg:string;

nn:real;

tmpl:List;

pp,a,b:integer;

ch:char;

x,j,i,n,code:integer;

s:string;

c:char;

m,k:tlist;

up,dn,au:boolean;

comand:**array of** string;

perAupn,updn,com:integer;

**procedure** commands;

**begin**

writeLn('Commands for working with the list: ');

writeLn('1. Add (Add a new list of items in the range [-10000,10000])');

writeLn('2. Show (Show list elements)');

writeLn('3. KickElement (Delete an element with the entered value)');

writeLn('4. Clear (Delete the entire list)');

writeLn('5. IncSort (Sorting elements in ascending order)');

writeLn('6. DecSort (Sorting elements in descending order)');

writeLn('7. Exit(You can also press button "Esc")');

readLn;

**end**;

**procedure** tab;

**var**

i:integer;

**begin**

i:=1;

**while** i<=length(s) **do begin**

**if** s[i] = ' ' **then begin** delete(s,i,1);

i:=i-1;

**end**

**else** i:=i+1;

**if** s[i] <>' ' **then break**;

**end**;

au:= false;

**case** s **of**

'a','ad': **begin** s:='add';i:=4; au:= true; **end**;

's','sh','sho': **begin** s:='show';i:=5; au:= true; **end**;

'h','he','hel': **begin** s:='help';i:=5; au:= true; **end**;

'k','ki','kic','kick','kicke','kickel','kickele','kickelem','kickeleme','kickelemen': **begin** s:='kickelement';i:=12; au:= true; **end**;

'c','cl','cle','clea': **begin** s:='clear';i:=6; au:= true; **end**;

'i','in','inc','incs','incso','incsor': **begin** s:='incsort';i:=8; au:= true; **end**;

'd','de','dec','decs','decso','decsor': **begin** s:='decsort';i:=8; au:= true; **end**;

'e','ex','exi': **begin** s:='exit';i:=5; au:= true; **end**;

**end**;

**if** au = true **then begin**

gotoxy(1,wherey);

write(s);

**end**;

**end**;

**procedure** menu;

**begin**

clrscr;

**case** s **of**

'add': **begin**

**repeat**

write('Enter the number of elements: ');

readLn(kk);

val(kk,nn,code);

**if** (code<>0) **or** (nn>10000) **or** (nn<1) **then begin**

writeLn('Incorrect input. The number is out of range or you entered a letter. Please press "Enter" and try again.');

readLn;

clrscr;

**end**

**until** (code=0) **and** (nn<10001) **and** (nn>0);

val(kk,pp,n);

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

**begin**

**repeat**

**begin**

writeLn('Enter the ',i,' element: ');

readln(aa);

val(aa,znach,code);

**if** (code<>0) **or** (znach>10000) **or** (znach<-10000) **then begin**

writeLn('Incorrect input. The number is out of range or you entered a letter. Please press "Enter" and try again.');

readLn;

**end**;

**end**;

**until** (code=0) **and** (znach<10001) **and** (znach>-10001);

AddElem(SpisNach,znach);

**end**;

**end**;

'help':commands;

'kickelement':**begin**

**repeat**

Write('Enter the value of the item to delete: ');

readln(gg);

val(gg,znach,code);

**if** (code<>0) **or** (znach>10000) **or** (znach<-10000) **then begin**

writeLn('Incorrect input. The number is out of range or you entered a letter. Please press "Enter" and try again.');

readLn;

**end**

**until** (code=0) **and** (znach<10001) **and** (znach>-10001);

DelElemZnach(SpisNach,znach);

**end**;

'show':Print(SpisNach);

'clear':**begin**

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

DelElemPos(SpisNach,i);

**end**;

'incsort':**begin if** spisNach=nil **then**

**begin**

writeLn('List is empty. Add some elements.');

readLn;

**end**

**else**

IncSort(SpisNach);

**end**;

'decsort':**begin if** spisNach=nil **then**

**begin**

writeLn('List is empty. Add some elements.');

readLn;

**end**

**else**

DecSort(SpisNach);

**end**;

'exit':**begin**

Free(SpisNach);

halt;

**end**;

**end**;

**while** keypressed = false **do**

clrscr;

**if** length(s)>0 **then begin**

com:=com+1;

setlength(comand,com);

comand[com-1]:=s;

**end**;

setlength(s,0);

i:=0;

textcolor(10);

writeln('Enter the command: ');

**end**;

**procedure** point;

**begin**

**if** com>0 **then begin**

**if** up = true **then** perAupn:=1;

**if** dn = false **then** perAupn:=-1;

**if** s = '' **then** s:=comand[com-1];

**if** (up = true) **or** (dn = true) **then begin**

updn:=updn + perAupn;

**if** updn>com **then** updn:=1;

**if** updn = 0 **then** updn:=com;

s:=comand[updn-1];

**end**;

**end**;

up:=false;

dn:=false;

**end**;

**begin**

SpisNach:=nil;

textcolor(10);

clrscr;

writeLn('Program for working with a ring list');

write('Enter ');

textcolor(red);

write('"Help"');

textcolor(10);

writeLn(' to view the list of the commands');

write('Use button ');

textcolor(red);

write('"Tab" ');

textcolor(10);

writeLn('for autocomplete commands');

writeLn;

writeln('Enter the command: ');

i:=0;

com:=0;

**repeat**

c:=readkey;

**case** c **of**

#97..#122:**if** i<100 **then begin**

i:=i+1;

setlength(s,i);

**if** i>wherex **then begin**

j:=1;

x:=wherex;

**while** i-x>=j **do begin**

s[i-j+1]:=s[i-j];

gotoxy(i-j+1,wherey);

write(s[i-j]);

j:=j+1;

**end**;

gotoxy(wherex-1,wherey);

s[wherex-1]:=c;

gotoxy(1,wherey);

write(s);

**end**

**else begin** s[i]:=c;

write(c);

**end**;

**end**;

#65..#90 : **if** i<100 **then begin** setlength(s,i); c:=chr(ord(c)+32);

i:=i+1;

setlength(s,i);

**if** i>wherex **then begin**

j:=1;

x:=wherex;

**while** i-x>=j **do begin**

s[i-j+1]:=s[i-j];

gotoxy(i-j+1,wherey);

write(s[i-j]);

j:=j+1;

**end**;

gotoxy(wherex-1,wherey);

s[wherex-1]:=c;

gotoxy(1,wherey);

write(s);

**end**

**else begin** s[i]:=c;

write(c);

**end**;

**end**;

#9: tab;

#8: **begin** x:=wherex; **if** x>1 **then begin** delete(s,wherex-1,1);i:=i-1;setlength(s,i);delline;gotoxy(1,wherey);write(s);gotoxy(x-1,wherey);**end**;**end**;

#13: menu;

**end**;

**if** c=#0 **then begin**

c:=readkey;

**case** c **of**

#72: **begin if** com>0 **then begin** up:=true; point;**end**;gotoxy(1,2);delline;write(s);**end**;

#75: **begin if** wherex>1 **then begin** gotoxy(wherex-1,wherey);**end**; **end**;

#77: **begin if** wherex<=length(s) **then begin** gotoxy(wherex+1,wherey);**end**; **end**;

#80: **begin if** com>0 **then begin** dn:=true; point;**end**;gotoxy(1,2);delline;write(s);**end**;

**end**;

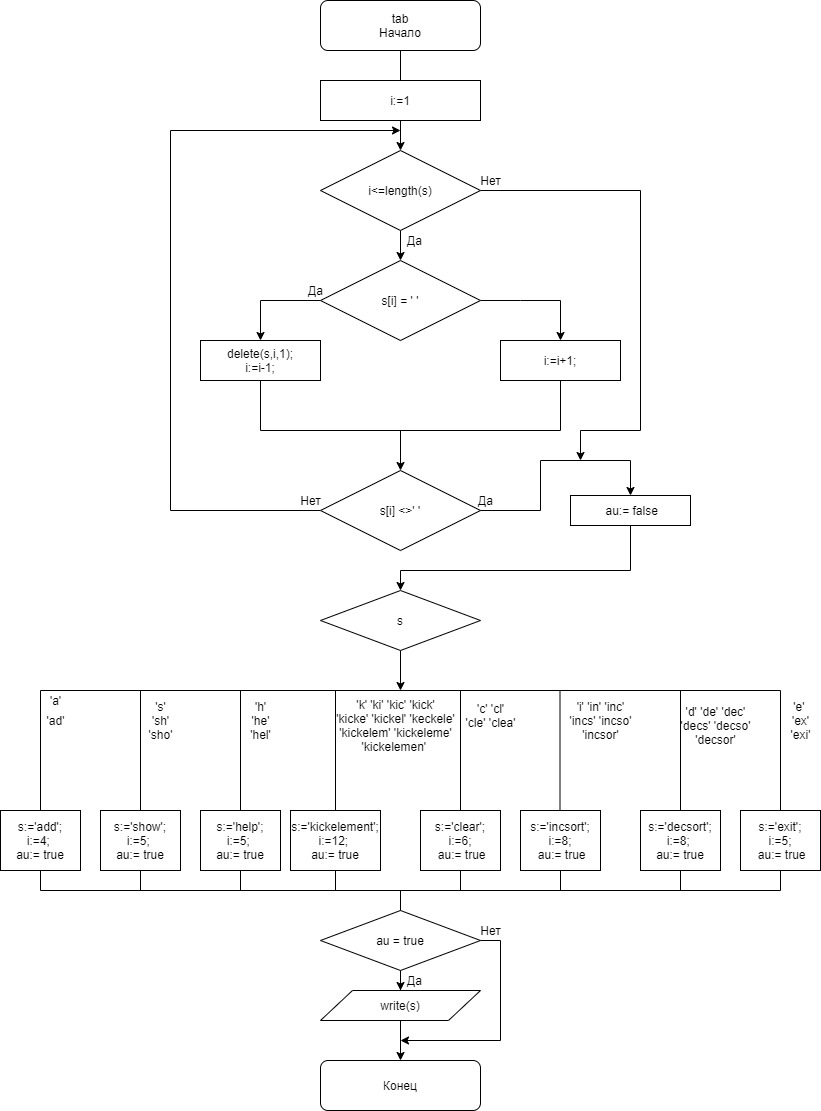
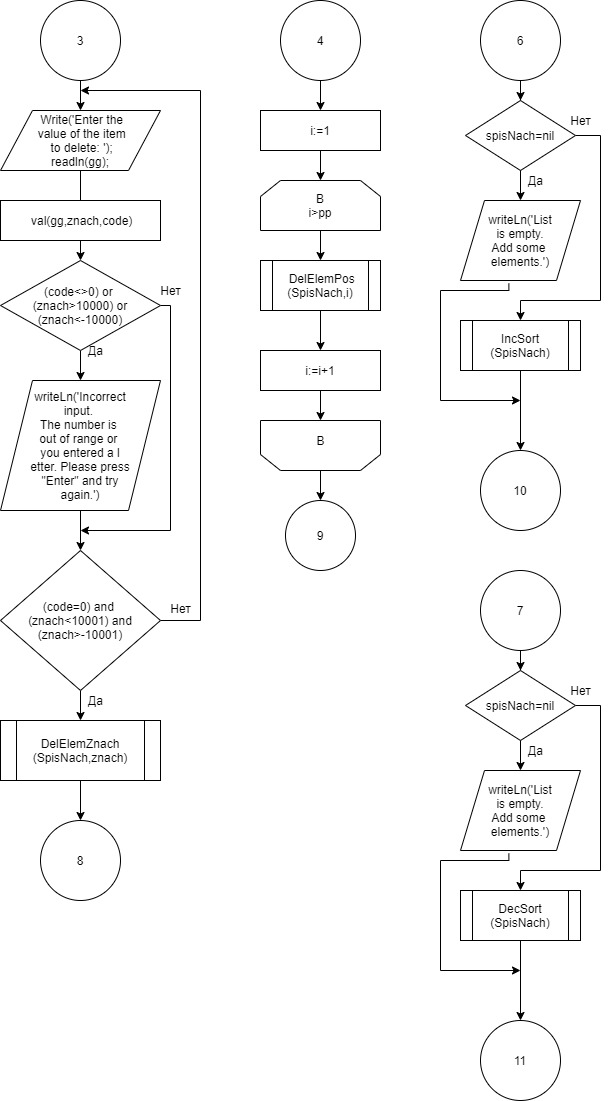
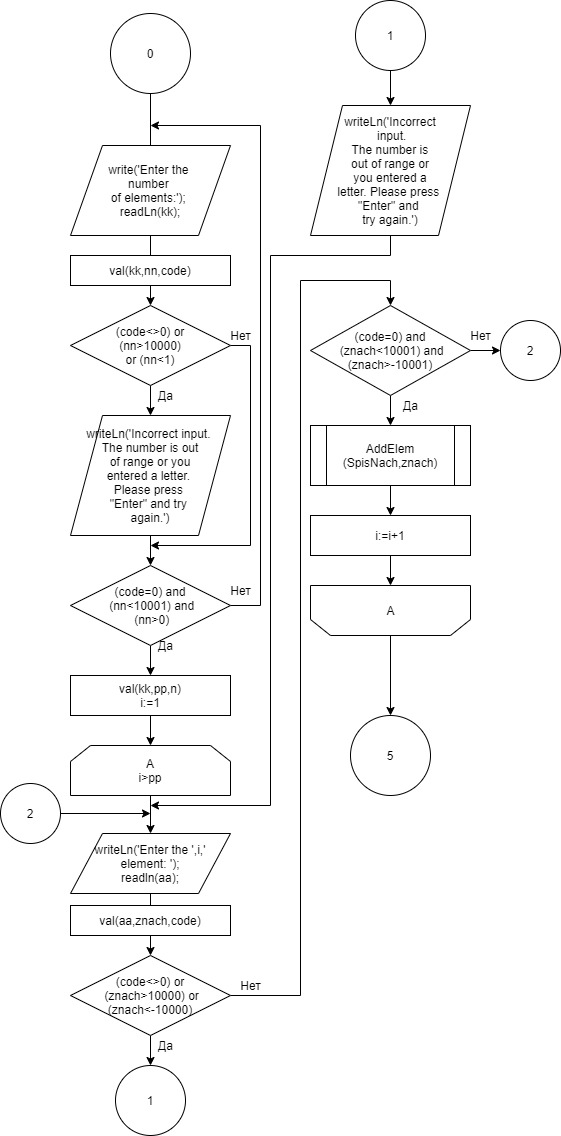
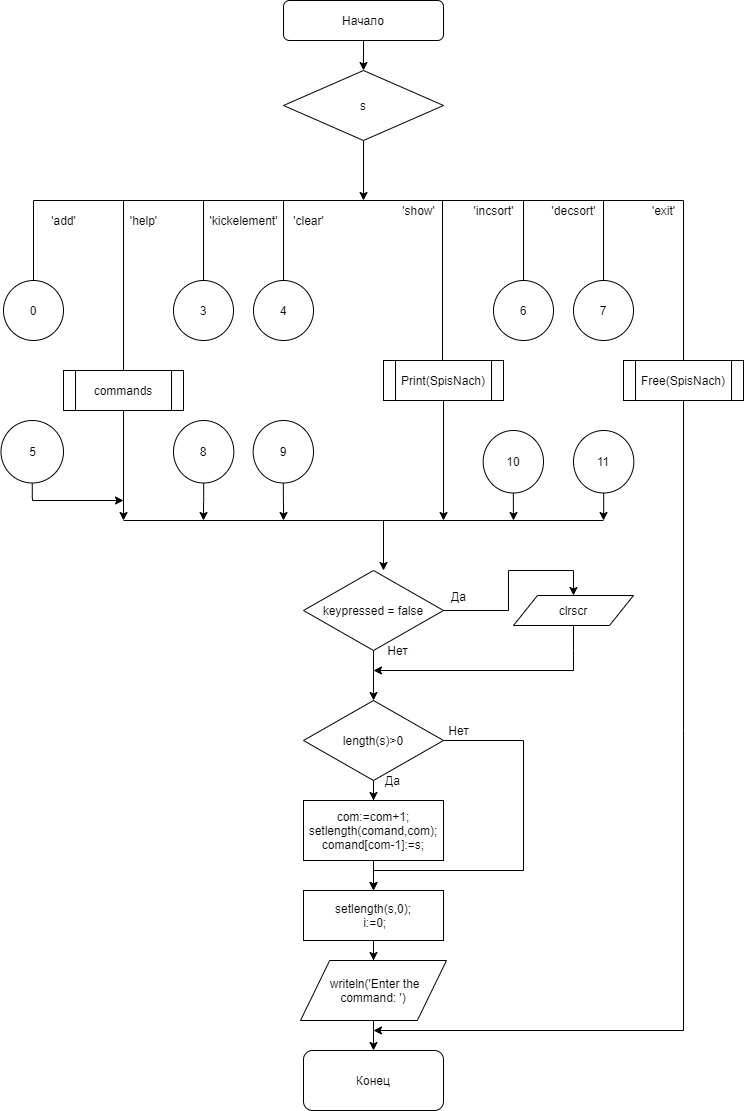
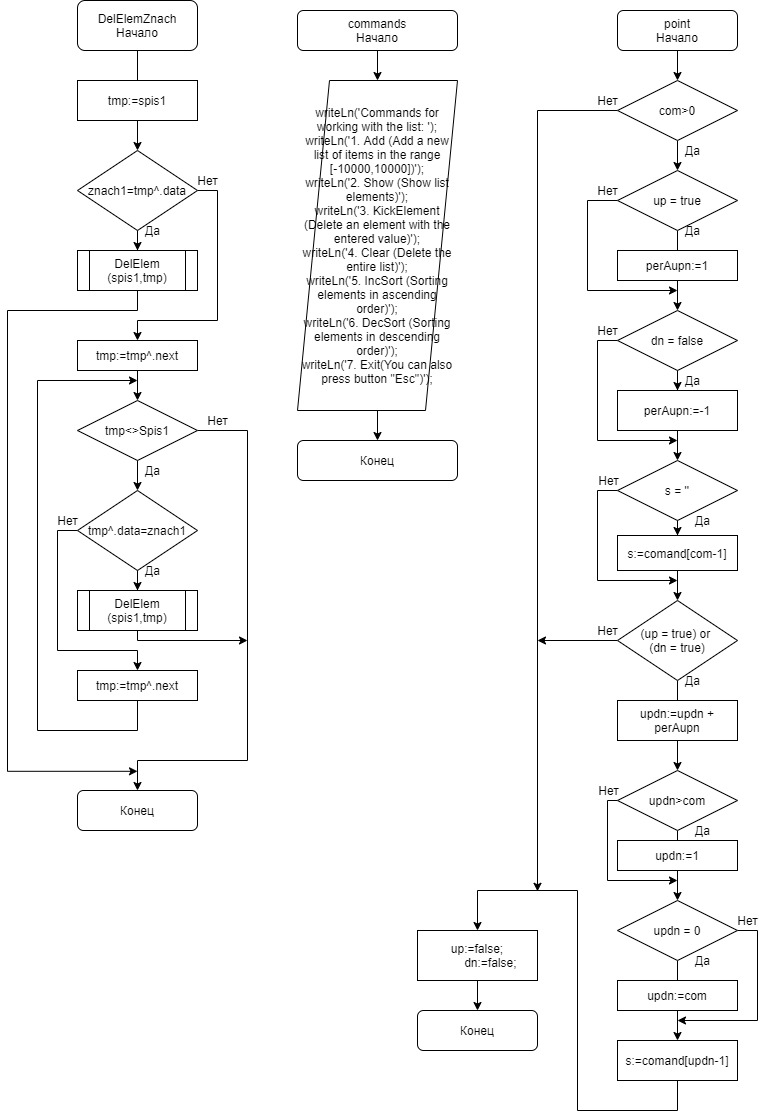
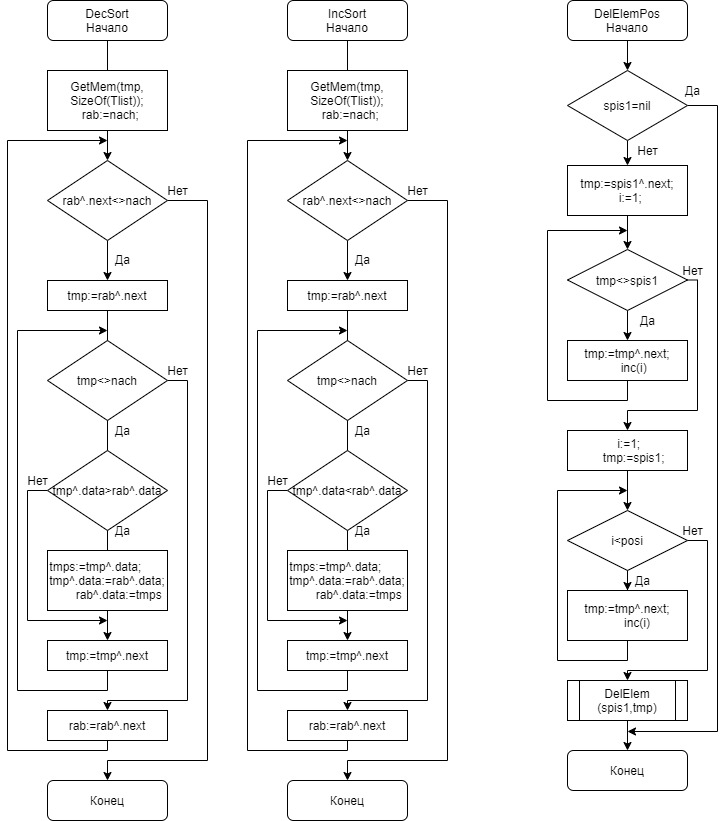
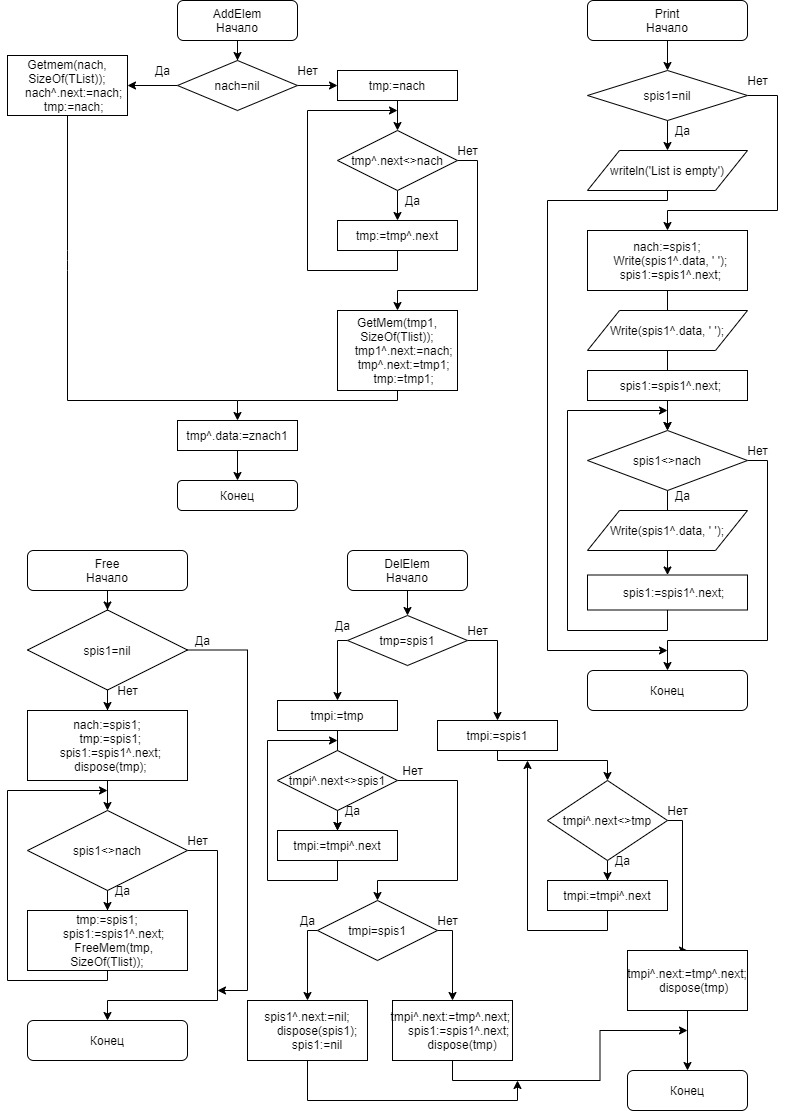
**end**;

**until** (c=#27);

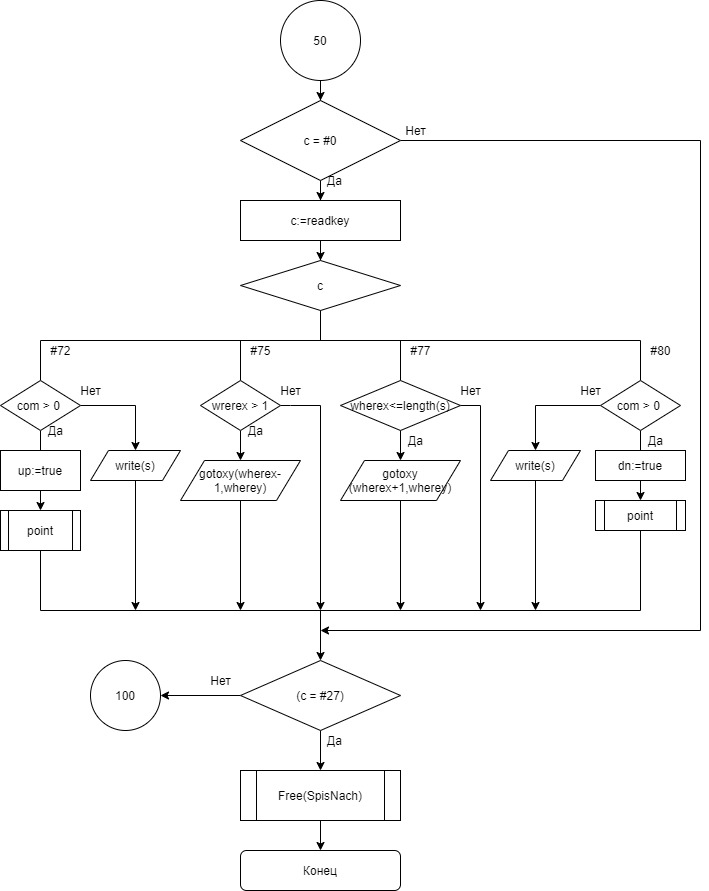
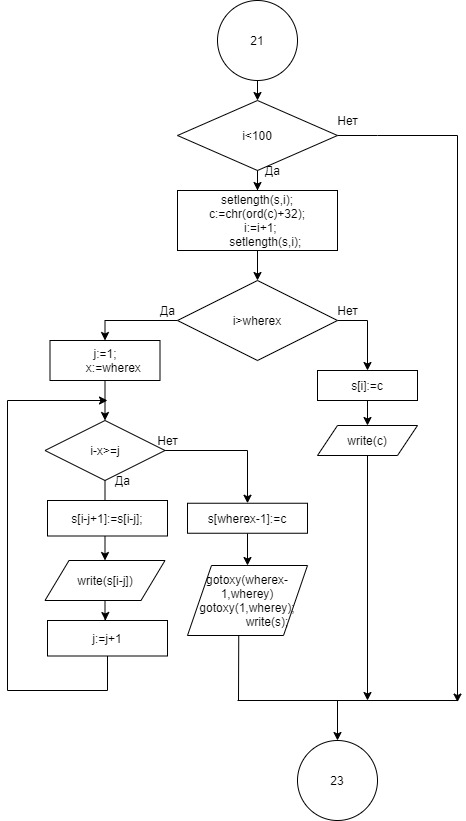
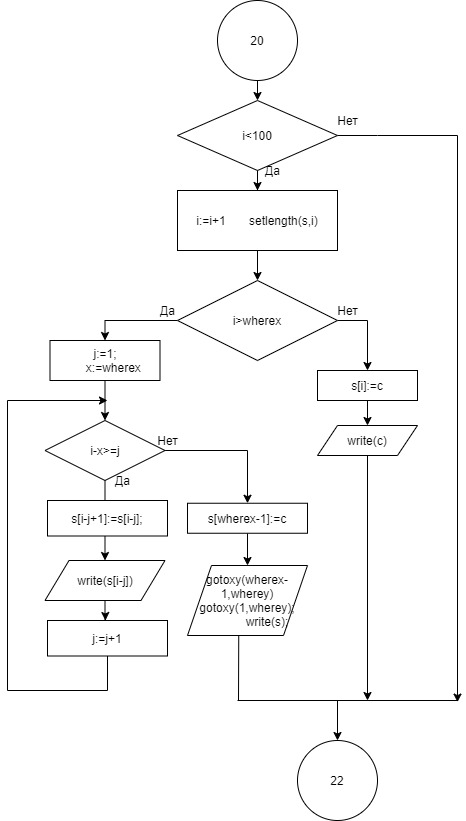
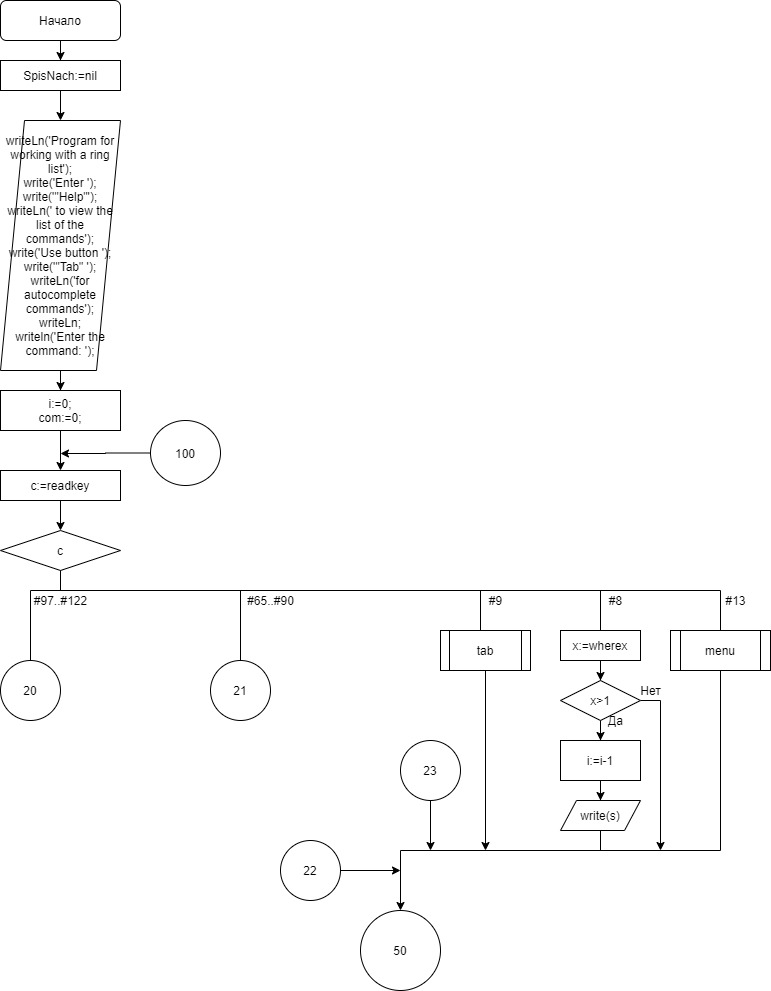
Free(SpisNach);

**end**.

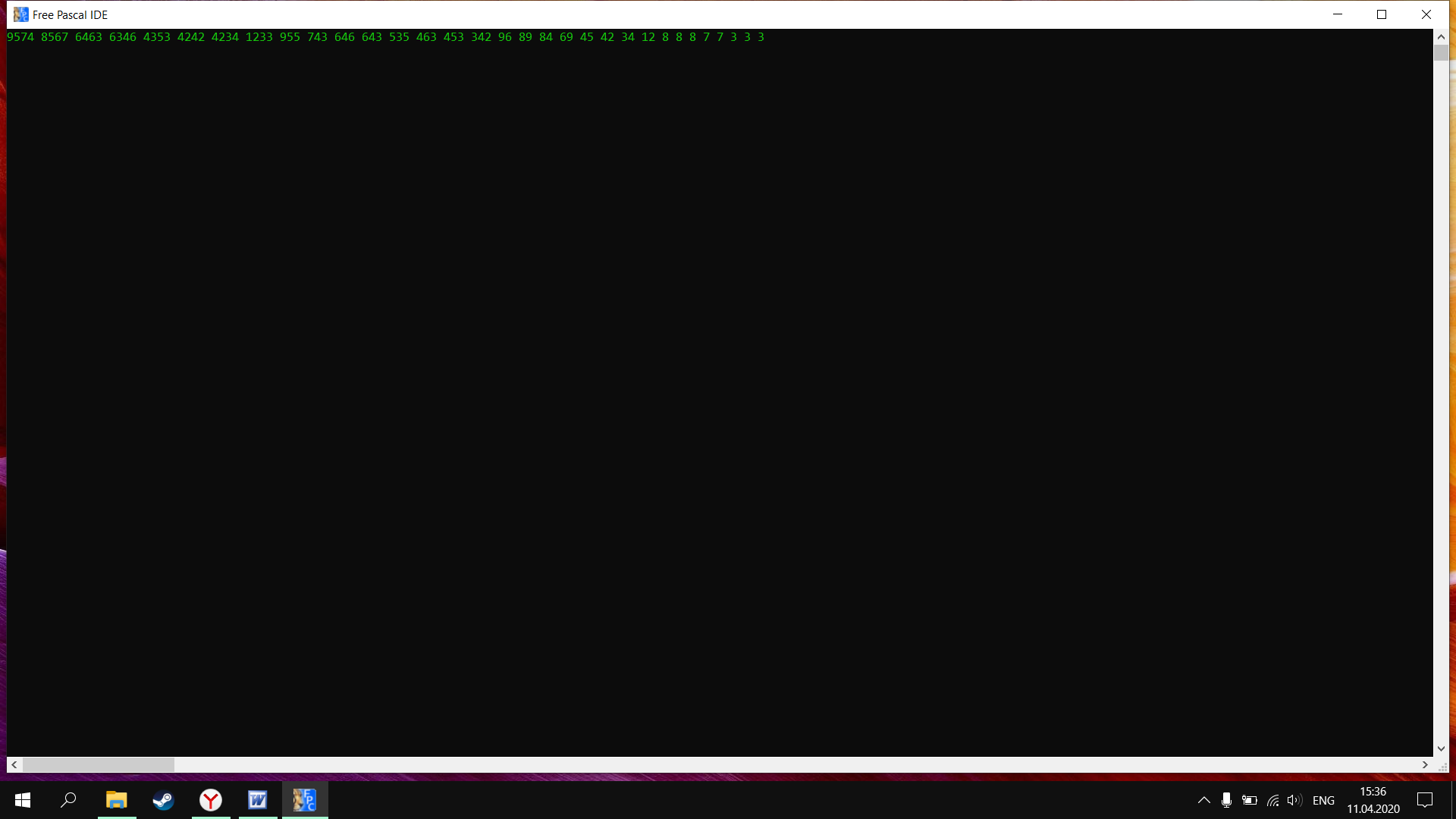
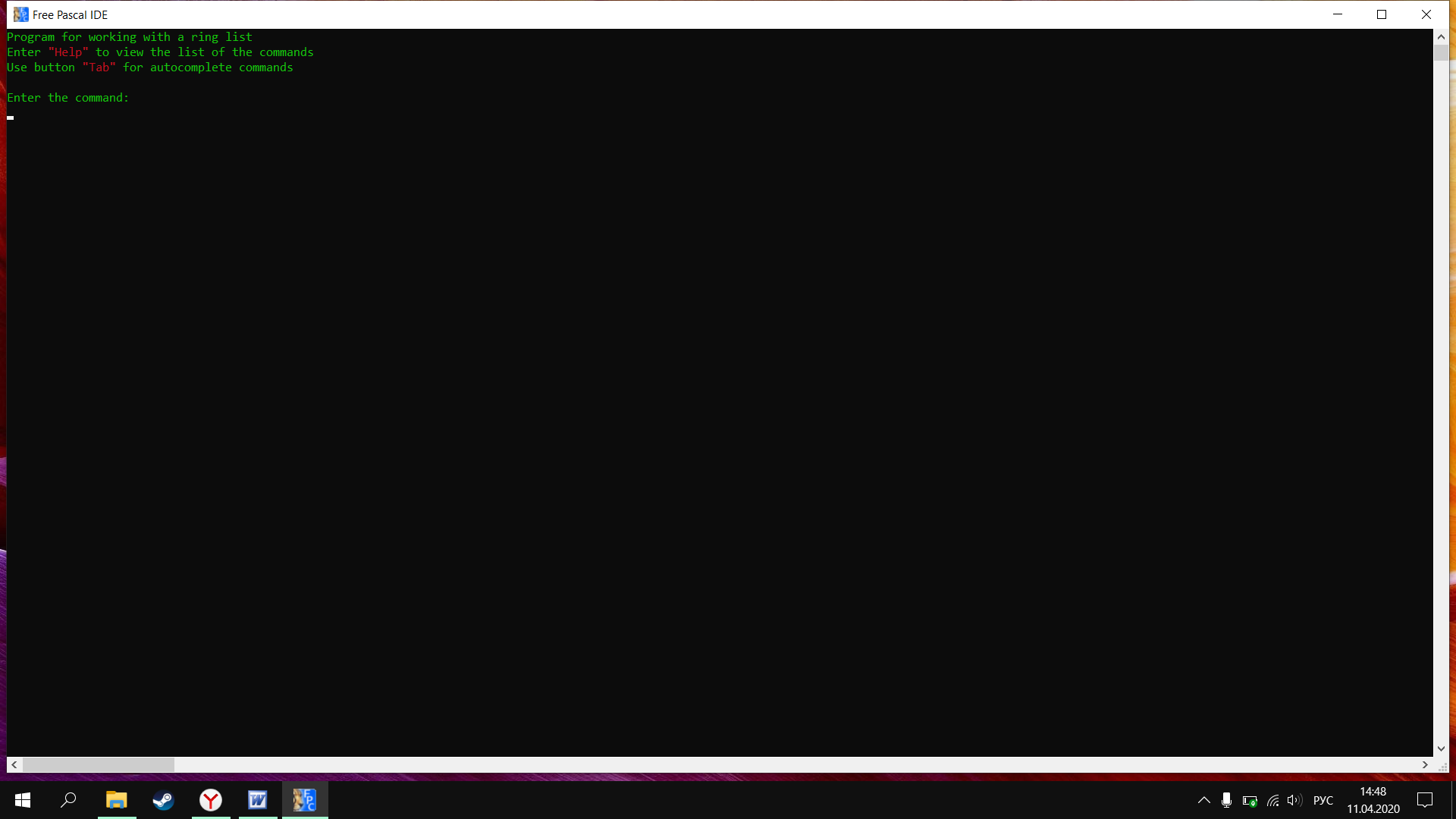
**Схема**



Нет



**Скриншоты программы**



**Вывод**

Язык программирования **Pascal** – это базовая высокоуровневая программа. Она используется как основа для программирования в большинстве современных школ. Такой востребованностью среди учебных заведений данный язык достиг благодаря своей простоте. Любая программа состоит из двух последовательных частей: описание последовательных действий, которое необходимо выполнить, и описание данных, которыми оперируют эти действия.

В ходе выполнения данной лабораторной работы была освоена работа с библиотеками. Освоили азы создания библиотек. Для этого необходимо создать отдельный текстовый документ и описать свою собственную библиотеку, которая включала бы все необходимые вам процедуры и функции. Для использования библиотеки в основной программе необходимо объявить её в разделе **Uses**.

Данная лабораторная работа потребовала использования библиотеки **Crt**. Она была необходима для реализации командной строки. В модуле Crt находятся мощные подпрограммы, которые дают возможность полного управления текстовым режимом. Подпрограммы модуля **Crt** обеспечивают контроль над текстовыми режимами экрана, расширенными кодами клавиатуры, цветами, окнами и звуком.

Лабораторная работы №6 потребовала использования динамической памяти. **Динамические переменные** - размещаются в памяти непосредственно в процессе работы программы. При динамическом размещении заранее неизвестны ни тип, ни количество размещаемых данных, к ним нельзя обращаться по именам, как к статическим переменным. **FreePascal** представляет средство управления динамической памятью: указатели.

В данной программе нужно было реализовать структуру данных **«список».** Список - структура данных, представляющая собой конечное множество упорядоченных элементов (узлов), связанных друг с другом посредством указателей, называется связным списком. Каждый элемент связного списка содержит поле с данными, а также указатель (ссылку) на следующий и/или предыдущий элемент. Эта структура позволяет эффективно выполнять операции добавления и удаления элементов для любой позиции в последовательности.