Министерство образования и науки Российской Федерации Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Пермский национальный исследовательский политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**О Т Ч Ё Т**

**по лабораторной работе №11.2**

Дисциплина: основы алгоритмизации и программирования

Тема: Информационные и динамические структуры. Очереди

Вариант: 25

Выполнил работу

студент группы РИС-20-1б

Рябов Никита Андреевич

Проверила

Доцент кафедры ИТАС Полякова О.А.

Пермь, 2021

**Цель**

Получить практические навыки работы с очередями.

**Постановка задачи**

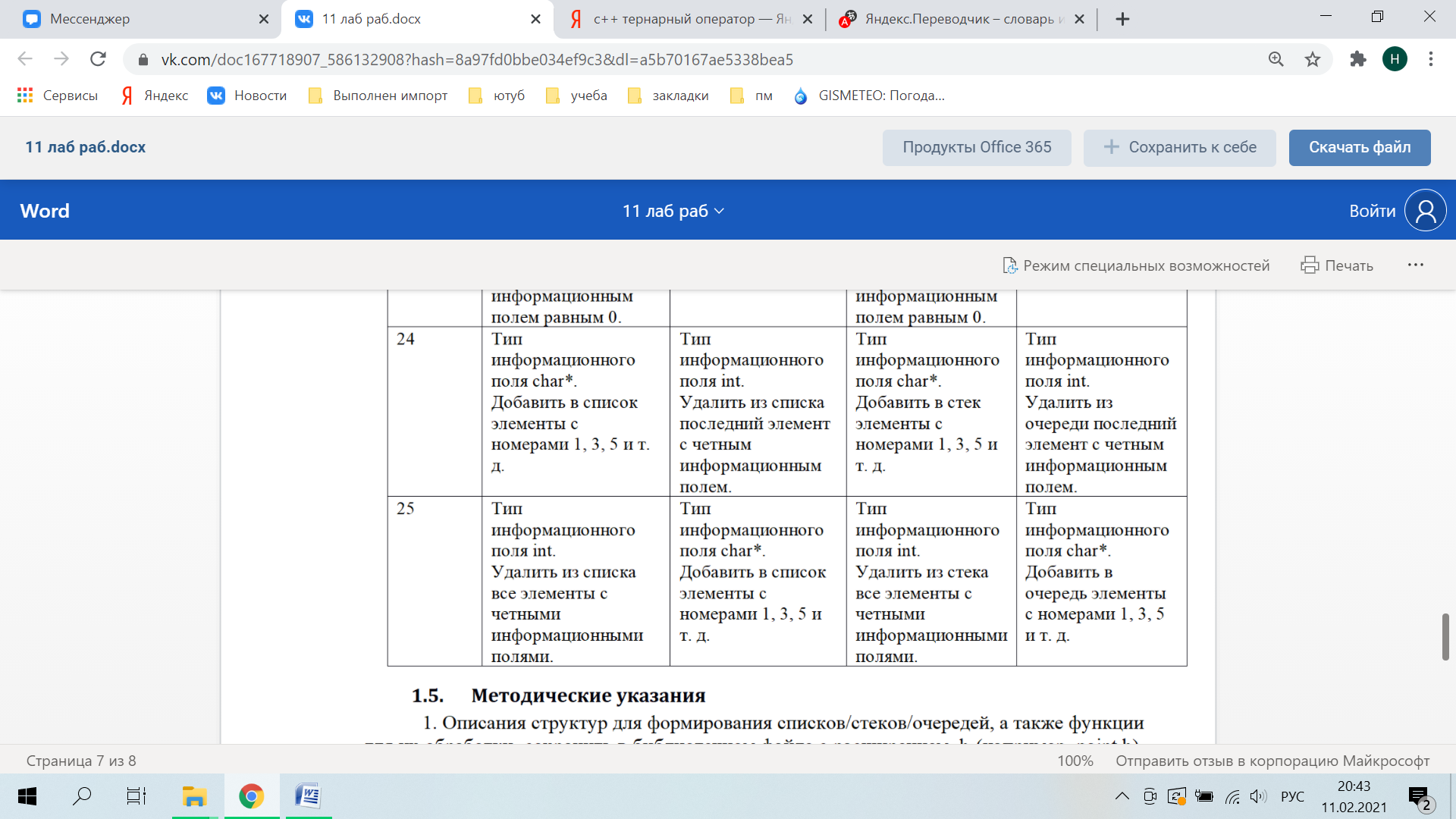
1. Сформировать однонаправленный и двунаправленный списки или стек и очередь. Тип информационного поля указан в варианте.

2. Распечатать полученную структуру.

3. Выполнить обработку структуры в соответствии с заданием.

4. Распечатать полученный результат.

5. Удалить соответствующую структуру из памяти.

****

**Анализ задачи**

*Какие предстоит выполнить действия.*

Функция Proc\_queue\_lib запрашивает элементы очереди, запрашивает новый элементы, перекладывает очередь по кругу с добавлением новых элементов. Затем очередь перемещается в стек и печатается.

for (int i=0;i<len;i++)

{

cout << "Введите слово\n";

cin >> s;

QUEUE.push(s);

}

for (int i=0;i<len;i++)

{

cout << "Введите новый элемент\n";

cin >> arr[i];

}

for (int i=0;i<len;i++)

{

QUEUE.push(QUEUE.front());

QUEUE.push(arr[len-i-1]);

QUEUE.pop();

}

for (int i=0;i<len\*2;i++)

{

STACK.push(QUEUE.front());

QUEUE.pop();

}

cout << "Результат\n";

for (int i=0;i<len\*2;i++)

{

cout << STACK.top() << endl;

STACK.pop();

}

Функция Proc\_queue\_fun выполняет аналогичные действия, но имена функций другие.

Функция POP\_queue удаляет первый элемент очереди

Queue\* POP\_queue(Queue\* QUEUE)

{

Queue\* El=QUEUE->last;

delete QUEUE;

return El;

}

Функция PUSH\_queue проходит до последнего элемента и добавляет к нему новый

void PUSH\_queue(Queue\* QUEUE, string s)

{

Queue\* El=QUEUE;

Queue\* NEW=new Queue;

while (El->last!=NULL) El=El->last;

NEW->data=s;

NEW->last=NULL;

El->last=NEW;

}

*С каким типом данных действуем*

Int, string и указатели на структуры

*В каком виде представлены данные*

Работа осуществлялась с очередями. Каждый из них состоял из структур.

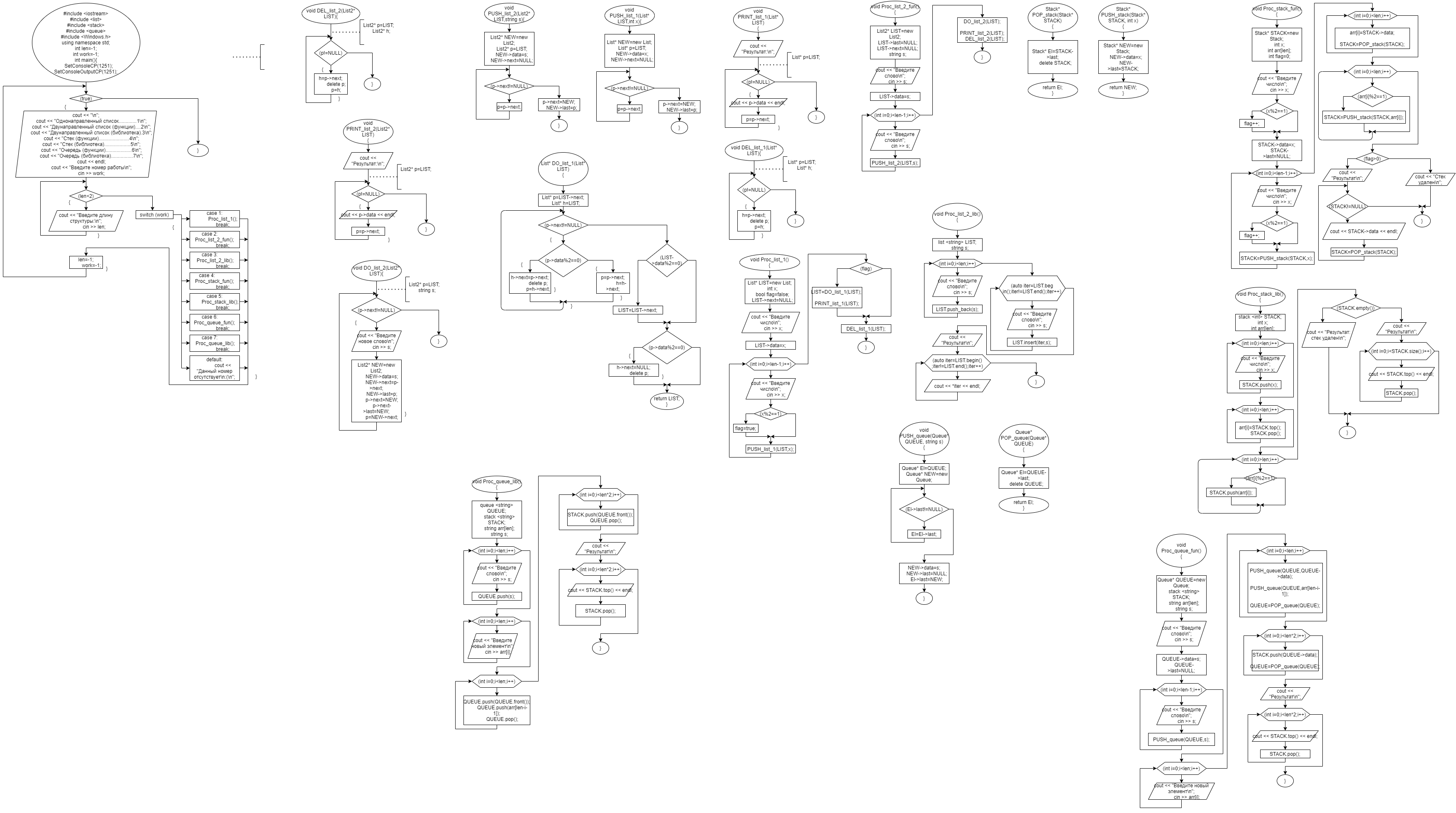
*Какие поля*

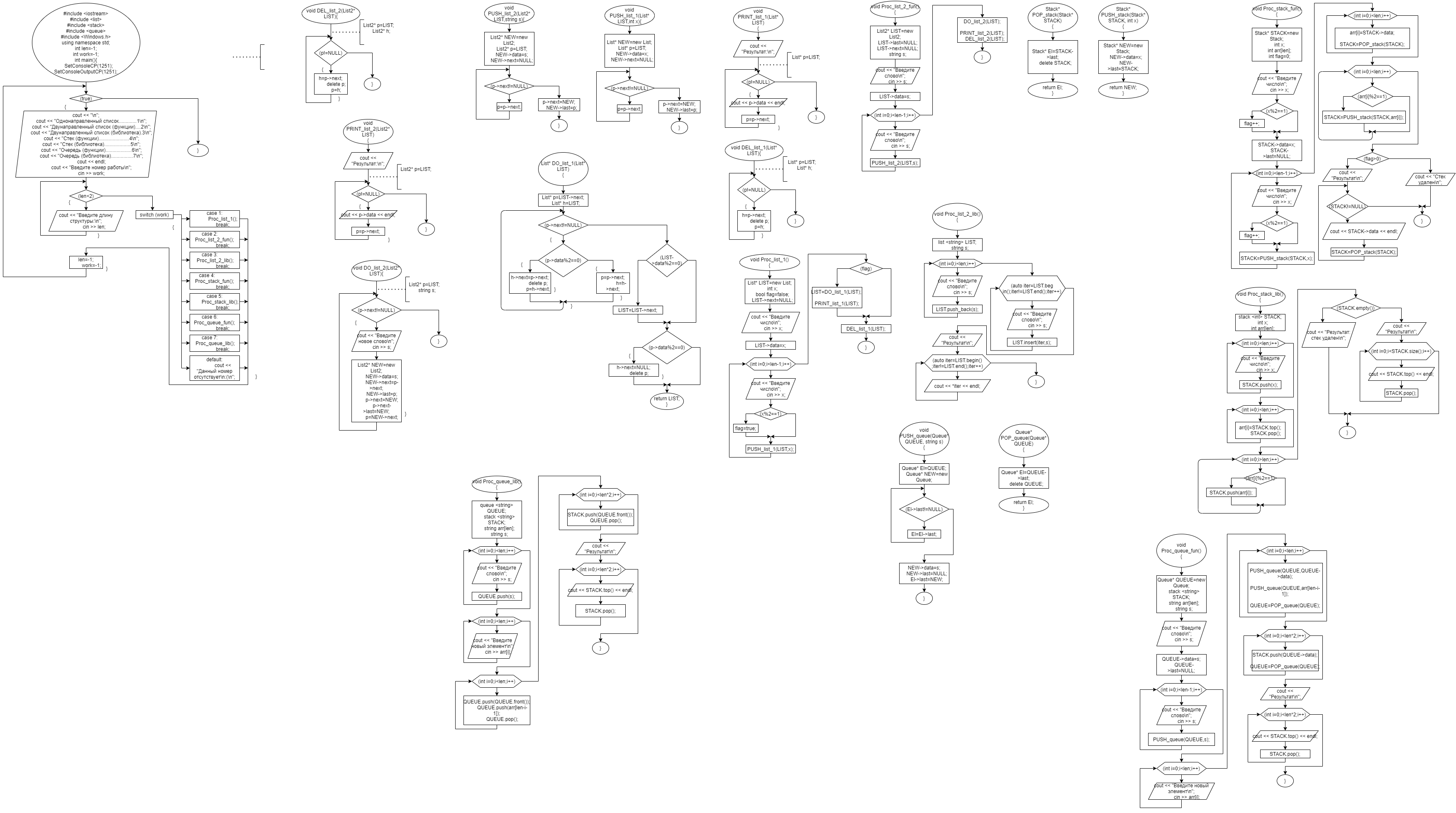
Структура содержит информационное поле типа int или string. Также существует указатель на такую же структуру (последующую или предыдущую)

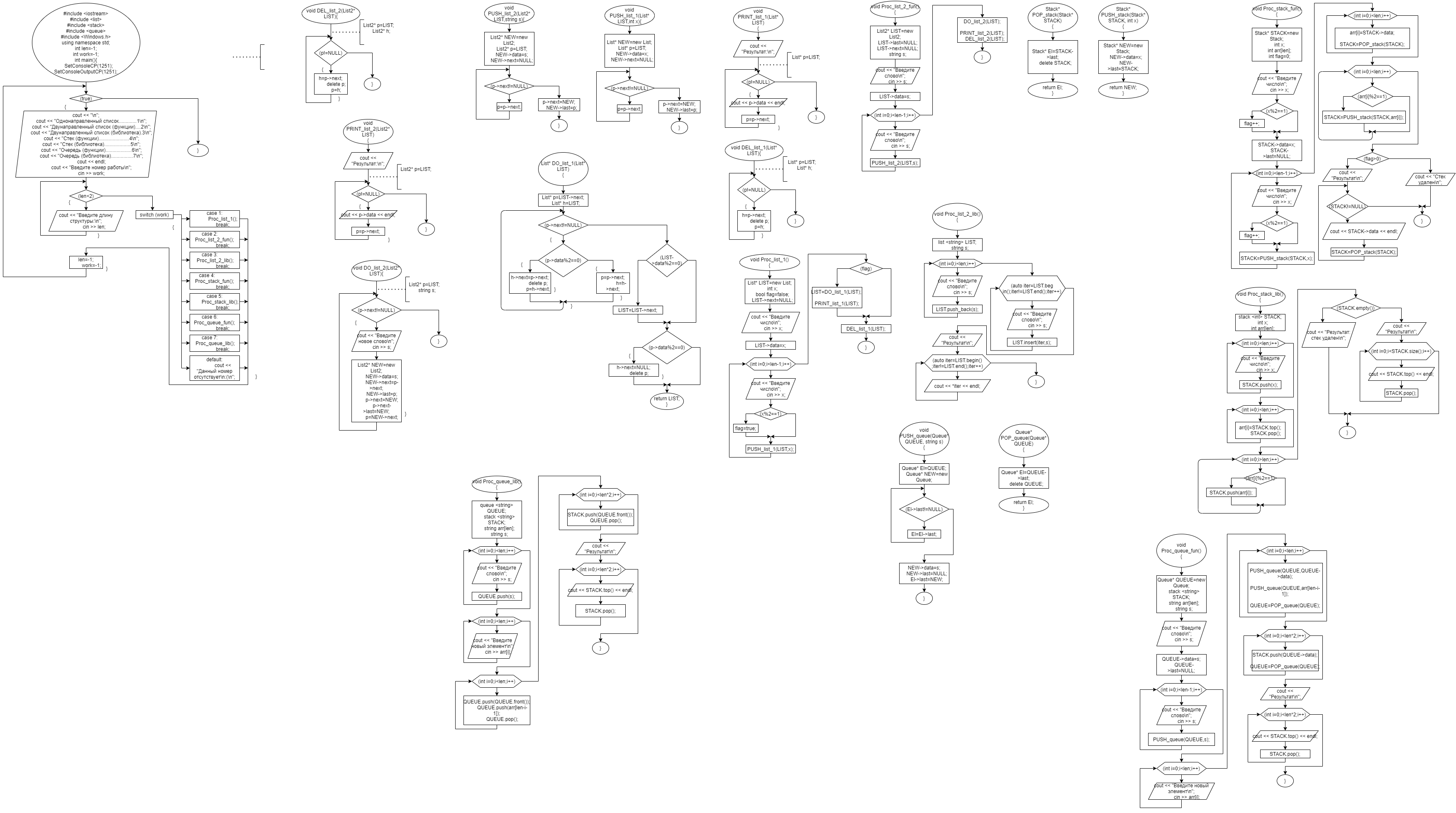
*Операторы ввода/вывода*

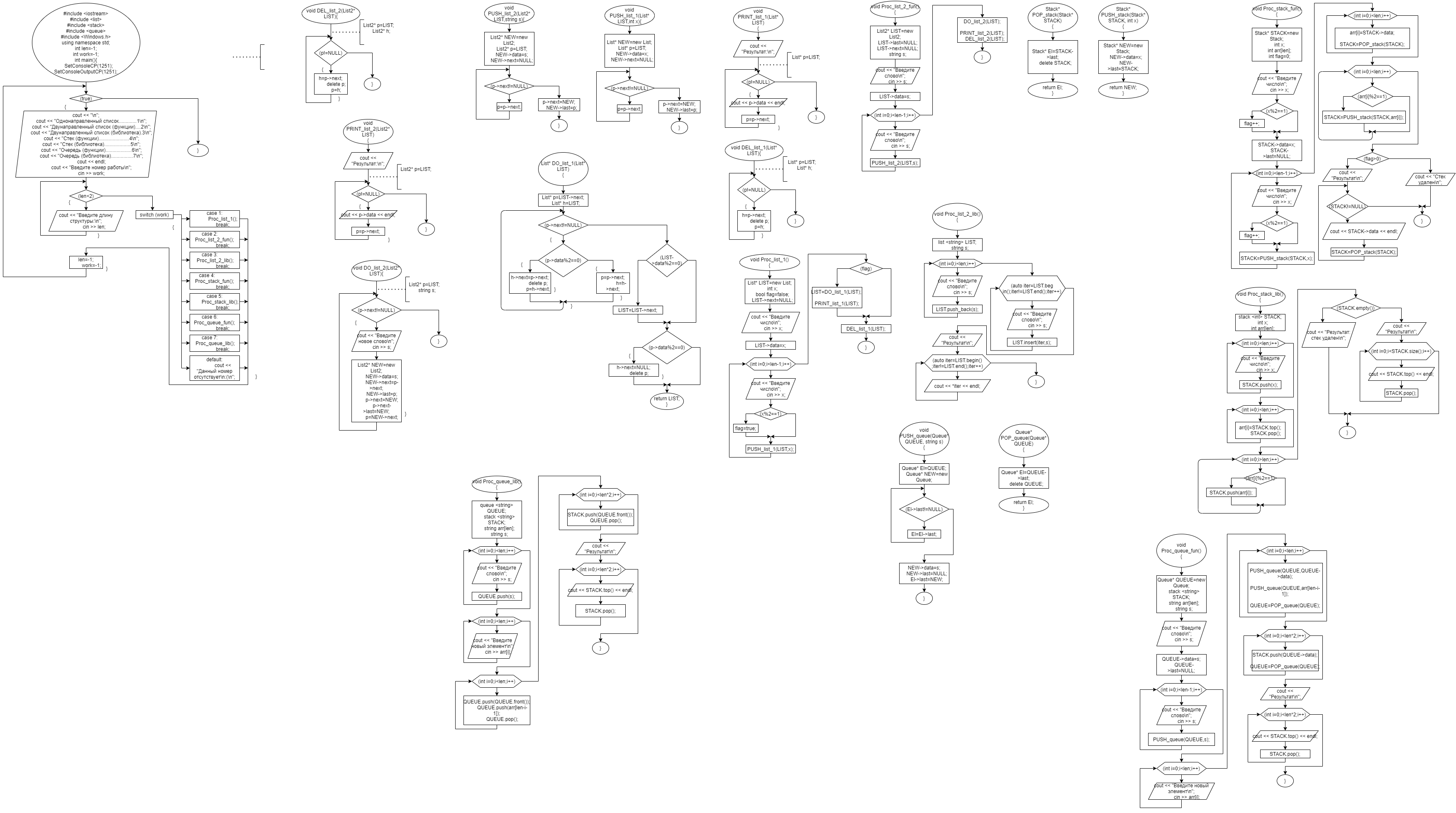
Cin/cout

**Блок-схема**

****

****

****

****

**Код**

#include <iostream>

#include <list>

#include <stack>

#include <queue>

#include <Windows.h>

using namespace std;

int len=-1;

int work=-1;

struct Queue

{

string data;

Queue\* last;

};

struct List2

{

string data;

List2\* next;

List2\* last;

};

struct Stack

{

int data;

Stack\* last;

};

struct List

{

int data;

List\* next;

};

void DEL\_list\_2(List2\* LIST)

{

List2\* p=LIST;

List2\* h;

while (p!=NULL)

{

h=p->next;

delete p;

p=h;

}

}

void PRINT\_list\_2(List2\* LIST)

{

cout << "Результат:\n";

List2\* p=LIST;

while (p!=NULL)

{

cout << p->data << endl;

p=p->next;

}

}

void DO\_list\_2(List2\* LIST)

{

List2\* p=LIST;

string s;

while (p->next!=NULL)

{

cout << "Введите новое слово\n";

cin >> s;

List2\* NEW=new List2;

NEW->data=s;

NEW->next=p->next;

NEW->last=p;

p->next=NEW;

p->next->last=NEW;

p=NEW->next;

}

}

void PUSH\_list\_2(List2\* LIST,string s)

{

List2\* NEW=new List2;

List2\* p=LIST;

NEW->data=s;

NEW->next=NULL;

while (p->next!=NULL) p=p->next;

p->next=NEW;

NEW->last=p;

}

void PUSH\_list\_1(List\* LIST,int x)

{

List\* NEW=new List;

List\* p=LIST;

NEW->data=x;

NEW->next=NULL;

while (p->next!=NULL) p=p->next;

p->next=NEW;

}

List\* DO\_list\_1(List\* LIST)

{

List\* p=LIST->next;

List\* h=LIST;

while (p->next!=NULL)

{

if (p->data%2==0)

{

h->next=p->next;

delete p;

p=h->next;

}

else

{

p=p->next;

h=h->next;

}

}

if (LIST->data%2==0) LIST=LIST->next;

if (p->data%2==0)

{

h->next=NULL;

delete p;

}

return LIST;

}

void PRINT\_list\_1(List\* LIST)

{

cout << "Результат:\n";

List\* p=LIST;

while (p!=NULL)

{

cout << p->data << endl;

p=p->next;

}

}

void DEL\_list\_1(List\* LIST)

{

List\* p=LIST;

List\* h;

while (p!=NULL)

{

h=p->next;

delete p;

p=h;

}

}

void Proc\_list\_1()

{

List\* LIST=new List;

int x;

bool flag=false;

LIST->next=NULL;

cout << "Введите число\n";

cin >> x;

LIST->data=x;

for (int i=0;i<len-1;i++)

{

cout << "Введите число\n";

cin >> x;

if (x%2==1) flag=true;

PUSH\_list\_1(LIST,x);

}

if (flag)

{

LIST=DO\_list\_1(LIST);

PRINT\_list\_1(LIST);

}

else cout << "Результат: список удален\n";

DEL\_list\_1(LIST);

}

void Proc\_list\_2\_fun()

{

List2\* LIST=new List2;

LIST->last=NULL;

LIST->next=NULL;

string s;

cout << "Введите слово\n";

cin >> s;

LIST->data=s;

for (int i=0;i<len-1;i++)

{

cout << "Введите слово\n";

cin >> s;

PUSH\_list\_2(LIST,s);

}

DO\_list\_2(LIST);

PRINT\_list\_2(LIST);

DEL\_list\_2(LIST);

}

void Proc\_list\_2\_lib()

{

list <string> LIST;

string s;

for (int i=0;i<len;i++)

{

cout << "Введите слово\n";

cin >> s;

LIST.push\_back(s);

}

for (auto iter=LIST.begin();iter!=LIST.end();iter++)

{

cout << "Введите новый элемент\n";

cin >> s;

LIST.insert(iter,s);

}

cout << "Результат\n";

for (auto iter=LIST.begin();iter!=LIST.end();iter++) cout << \*iter << endl;

}

Stack\* PUSH\_stack(Stack\* STACK, int x)

{

Stack\* NEW=new Stack;

NEW->data=x;

NEW->last=STACK;

return NEW;

}

Stack\* POP\_stack(Stack\* STACK)

{

Stack\* El=STACK->last;

delete STACK;

return El;

}

void Proc\_stack\_fun()

{

Stack\* STACK=new Stack;

int x;

int arr[len];

int flag=0;

cout << "Введите число\n";

cin >> x;

if (x%2==1) flag++;

STACK->data=x;

STACK->last=NULL;

for (int i=0;i<len-1;i++)

{

cout << "Введите число\n";

cin >> x;

if (x%2==1) flag++;

STACK=PUSH\_stack(STACK,x);

}

for (int i=0;i<len;i++)

{

arr[i]=STACK->data;

STACK=POP\_stack(STACK);

}

for (int i=0;i<len;i++) if (arr[i]%2==1) STACK=PUSH\_stack(STACK,arr[i]);

if (flag>0)

{

cout << "Результат\n";

while (STACK!=NULL)

{

cout << STACK->data << endl;

STACK=POP\_stack(STACK);

}

}

else cout << "Стек удален\n";

}

void Proc\_stack\_lib()

{

stack <int> STACK;

int x;

int arr[len];

for (int i=0;i<len;i++)

{

cout << "Введите число\n";

cin >> x;

STACK.push(x);

}

for (int i=0;i<len;i++)

{

arr[i]=STACK.top();

STACK.pop();

}

for (int i=0;i>len;i++) if (arr[i]%2==1) STACK.push(arr[i]);

if (STACK.empty()) cout << "Результат: стек удален\n";

else

{

cout << "Результат\n";

for (int i=0;i<STACK.size();i++)

{

cout << STACK.top() << endl;

STACK.pop();

}

}

}

void PUSH\_queue(Queue\* QUEUE, string s)

{

Queue\* El=QUEUE;

Queue\* NEW=new Queue;

while (El->last!=NULL) El=El->last;

NEW->data=s;

NEW->last=NULL;

El->last=NEW;

}

Queue\* POP\_queue(Queue\* QUEUE)

{

Queue\* El=QUEUE->last;

delete QUEUE;

return El;

}

void Proc\_queue\_fun()

{

Queue\* QUEUE=new Queue;

stack <string> STACK;

string arr[len];

string s;

cout << "Введите слово\n";

cin >> s;

QUEUE->data=s;

QUEUE->last=NULL;

for (int i=0;i<len-1;i++)

{

cout << "Введите слово\n";

cin >> s;

PUSH\_queue(QUEUE,s);

}

for (int i=0;i<len;i++)

{

cout << "Введите новый элемент\n";

cin >> arr[i];

}

for (int i=0;i<len;i++)

{

PUSH\_queue(QUEUE,QUEUE->data);

PUSH\_queue(QUEUE,arr[len-i-1]);

QUEUE=POP\_queue(QUEUE);

}

for (int i=0;i<len\*2;i++)

{

STACK.push(QUEUE->data);

QUEUE=POP\_queue(QUEUE);

}

cout << "Результат\n";

for (int i=0;i<len\*2;i++)

{

cout << STACK.top() << endl;

STACK.pop();

}

}

void Proc\_queue\_lib()

{

queue <string> QUEUE;

stack <string> STACK;

string arr[len];

string s;

for (int i=0;i<len;i++)

{

cout << "Введите слово\n";

cin >> s;

QUEUE.push(s);

}

for (int i=0;i<len;i++)

{

cout << "Введите новый элемент\n";

cin >> arr[i];

}

for (int i=0;i<len;i++)

{

QUEUE.push(QUEUE.front());

QUEUE.push(arr[len-i-1]);

QUEUE.pop();

}

for (int i=0;i<len\*2;i++)

{

STACK.push(QUEUE.front());

QUEUE.pop();

}

cout << "Результат\n";

for (int i=0;i<len\*2;i++)

{

cout << STACK.top() << endl;

STACK.pop();

}

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

while (true)

{

cout << "\n";

cout << "Однонаправленный список.............1\n";

cout << "Двунаправленный список (функции)....2\n";

cout << "Двунаправленный список (библиотека).3\n";

cout << "Стек (функции)......................4\n";

cout << "Стек (библиотека)...................5\n";

cout << "Очередь (функции)...................6\n";

cout << "Очередь (библиотека)................7\n";

cout << endl;

cout << "Введите номер работы\n";

cin >> work;

while (len<2)

{

cout << "Введите длину структуры:\n";

cin >> len;

}

switch (work)

{

case 1:

Proc\_list\_1();

break;

case 2:

Proc\_list\_2\_fun();

break;

case 3:

Proc\_list\_2\_lib();

break;

case 4:

Proc\_stack\_fun();

break;

case 5:

Proc\_stack\_lib();

break;

case 6:

Proc\_queue\_fun();

break;

case 7:

Proc\_queue\_lib();

break;

default:

cout << "Данный номер отсутствует\n:(\n";

}

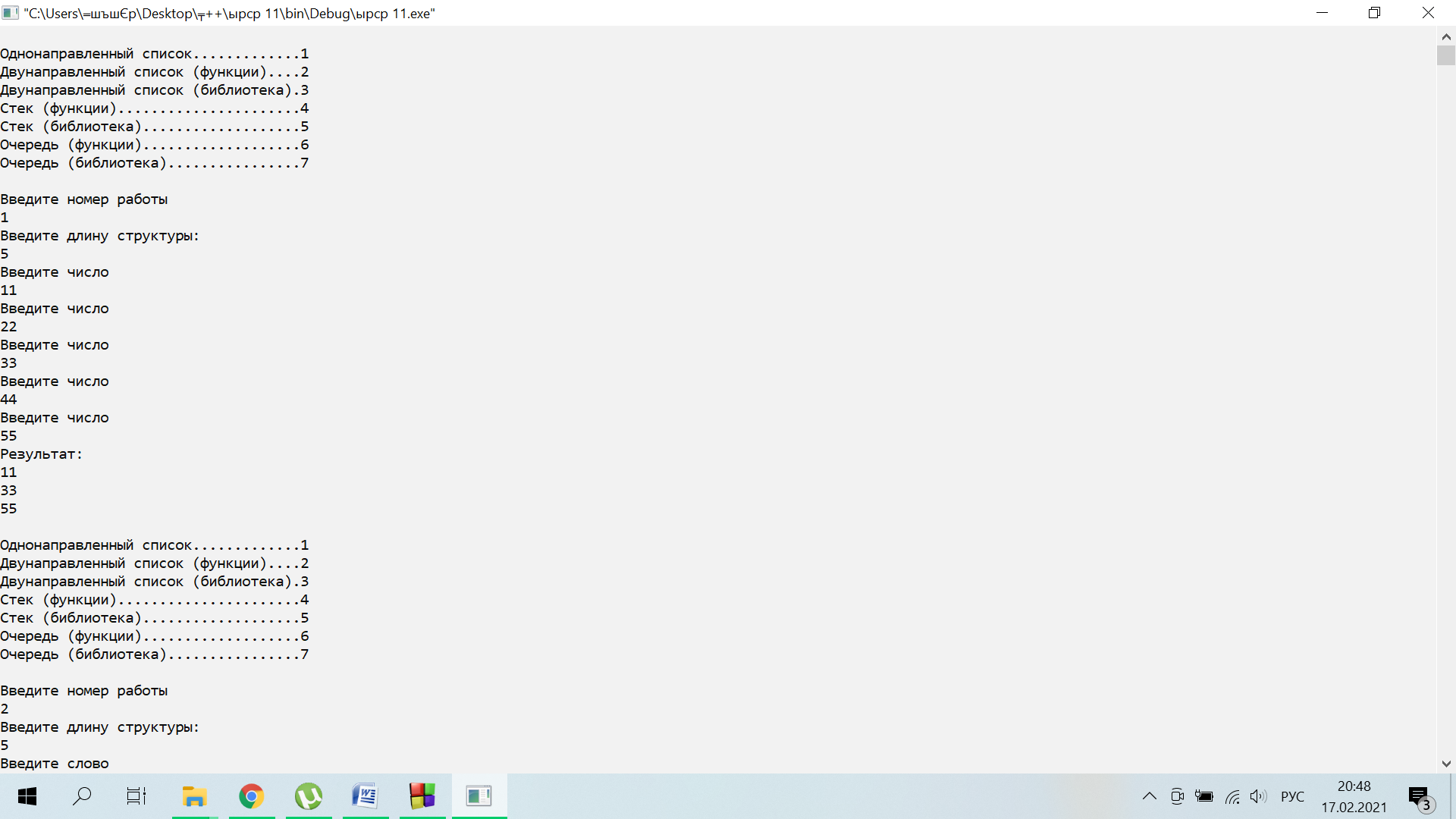
len=-1;

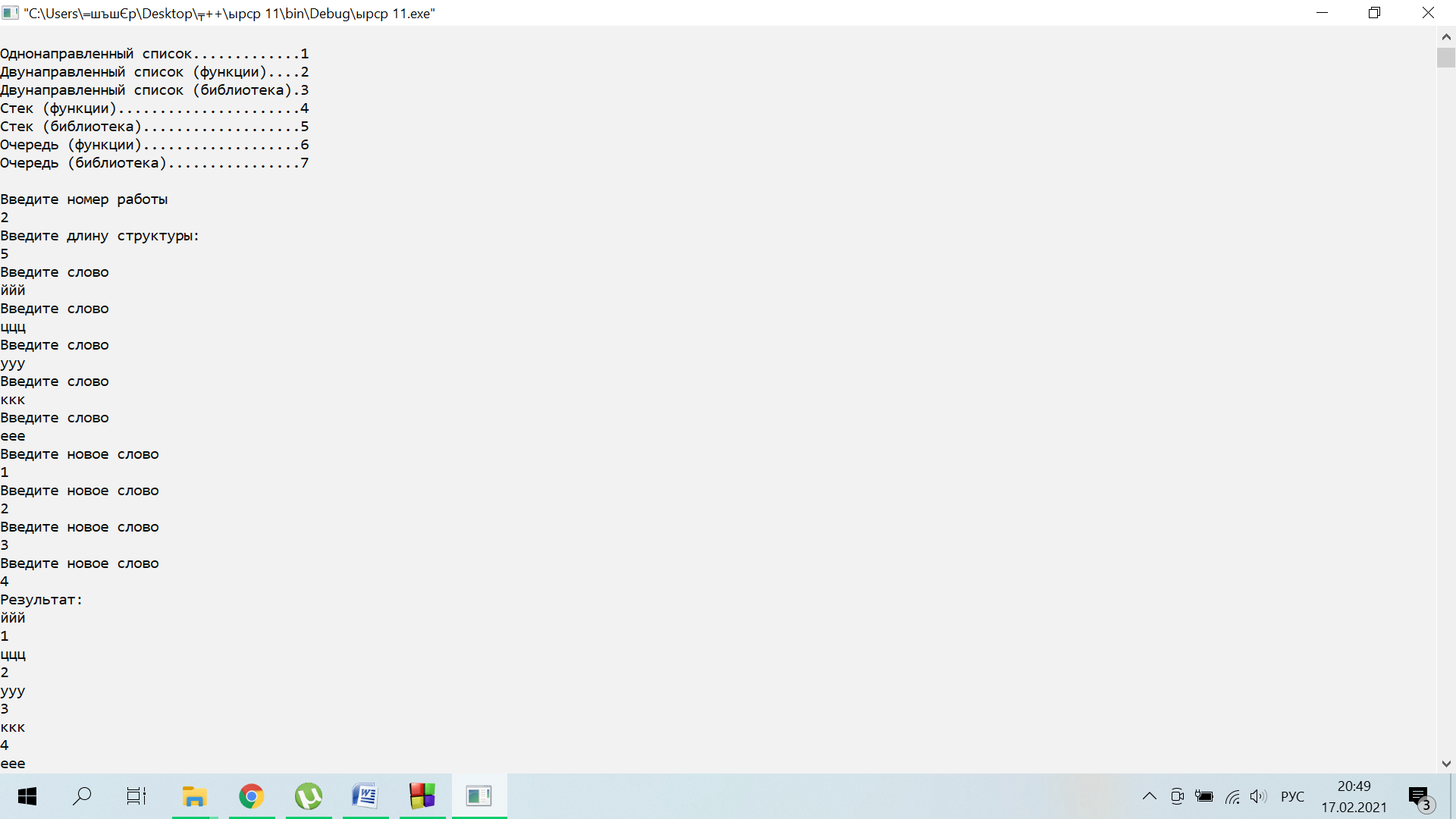
work=-1;

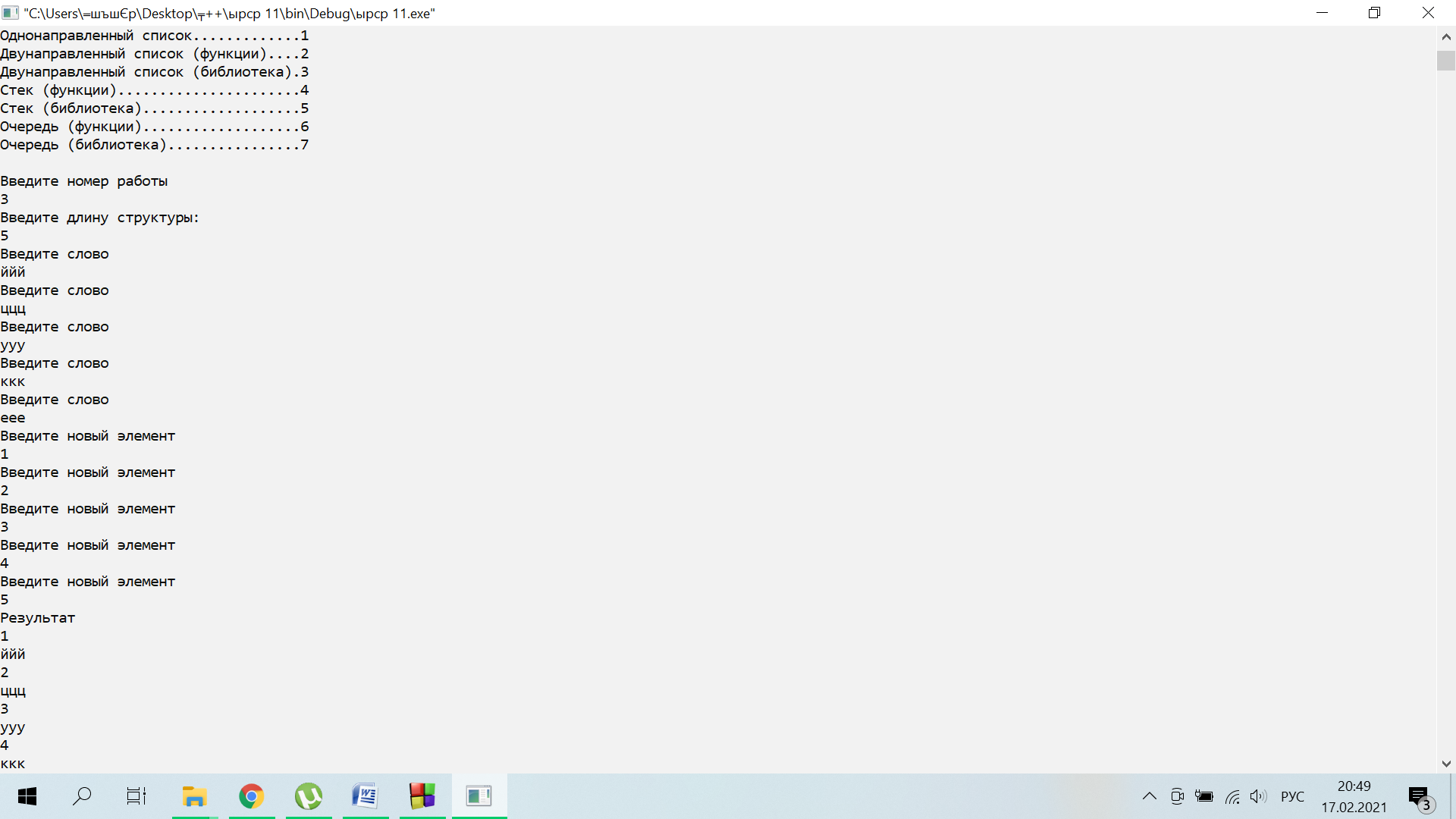
}

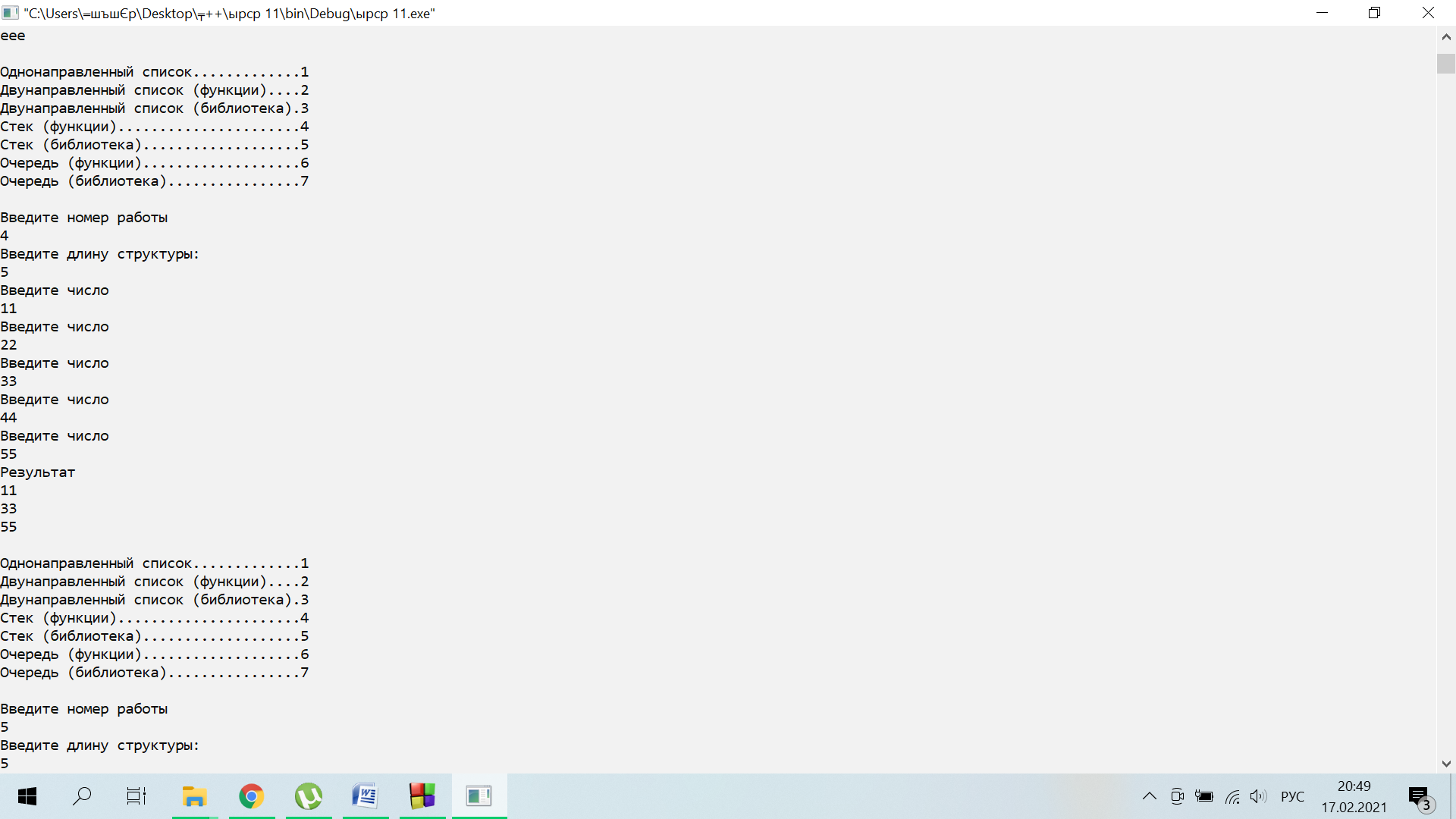
}

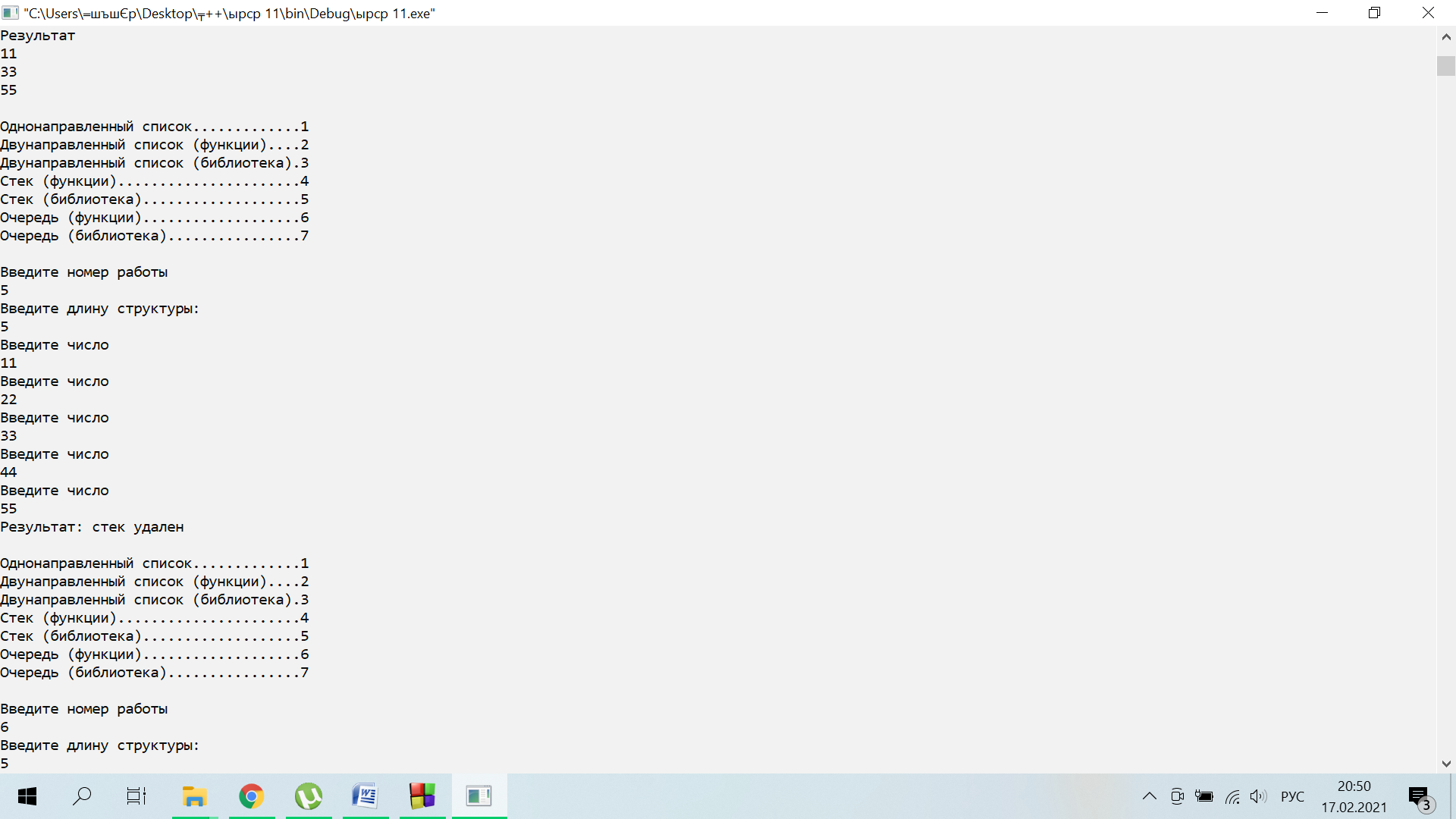
**Скриншоты**

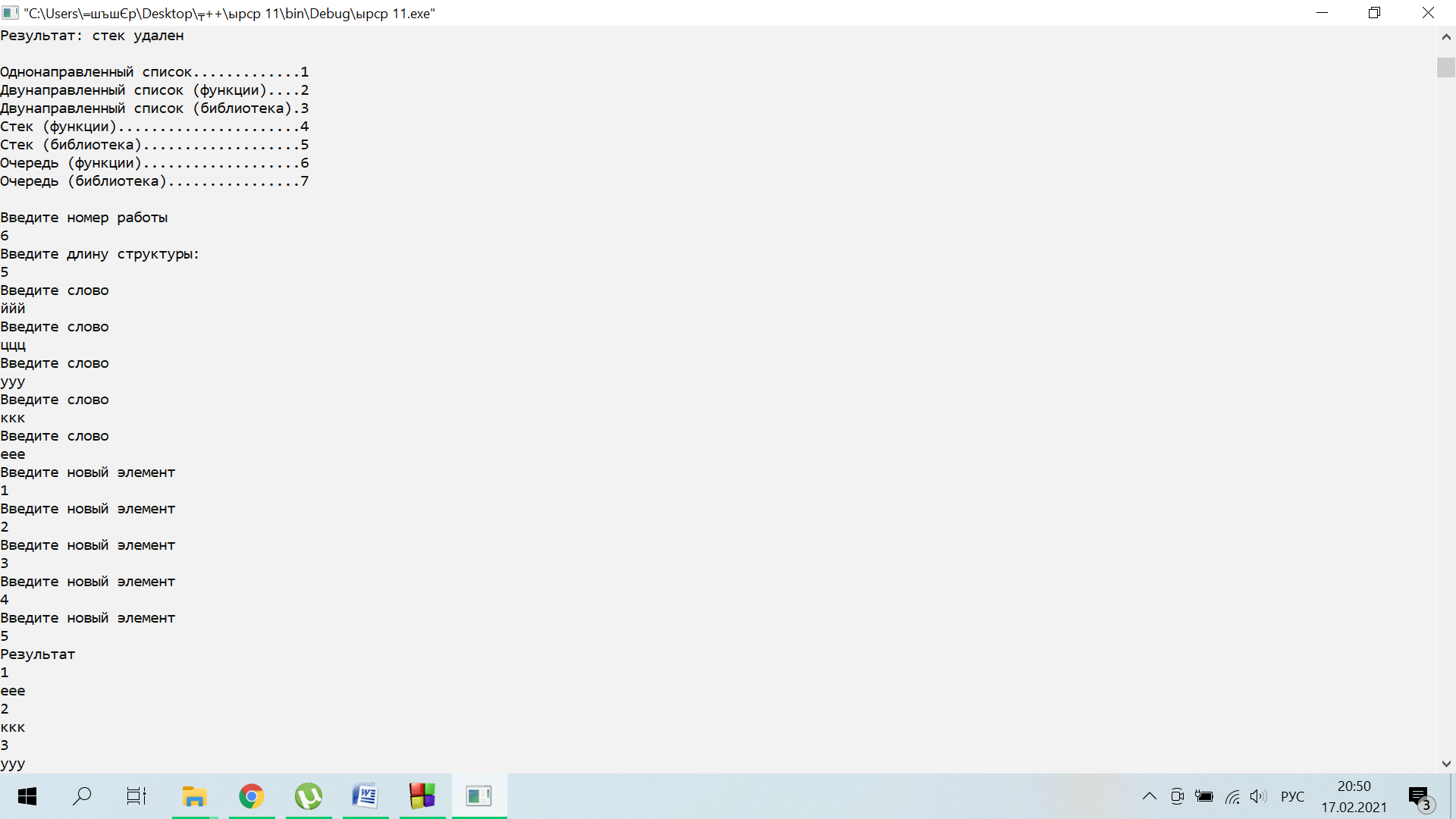
****

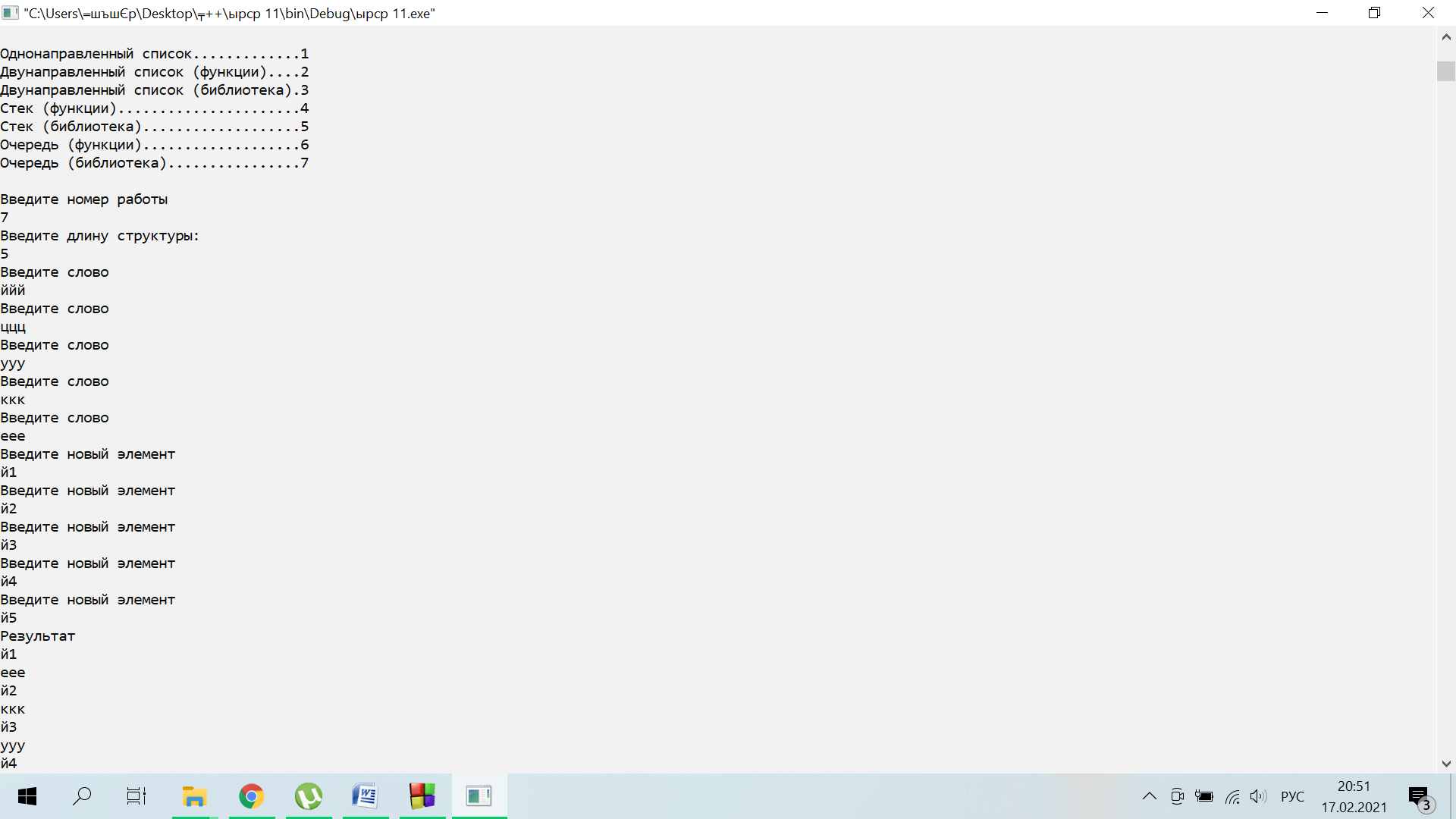
****

****

****

****

****

****