**“ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ**

**УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ” (ТУСУР)**

Кафедра автоматизированных систем управления (АСУ)

Отчет по лабораторной работе №3

«Связные списки»

По дисциплине СиАОДвЭВМ

Выполнил студент гр. 438-3:

\_\_\_\_\_\_\_\_\_\_\_\_\_ Канаев О.А.

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г.

Проверил

Доктор технических наук,

доцент каф. АСУ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Горитов А.Н

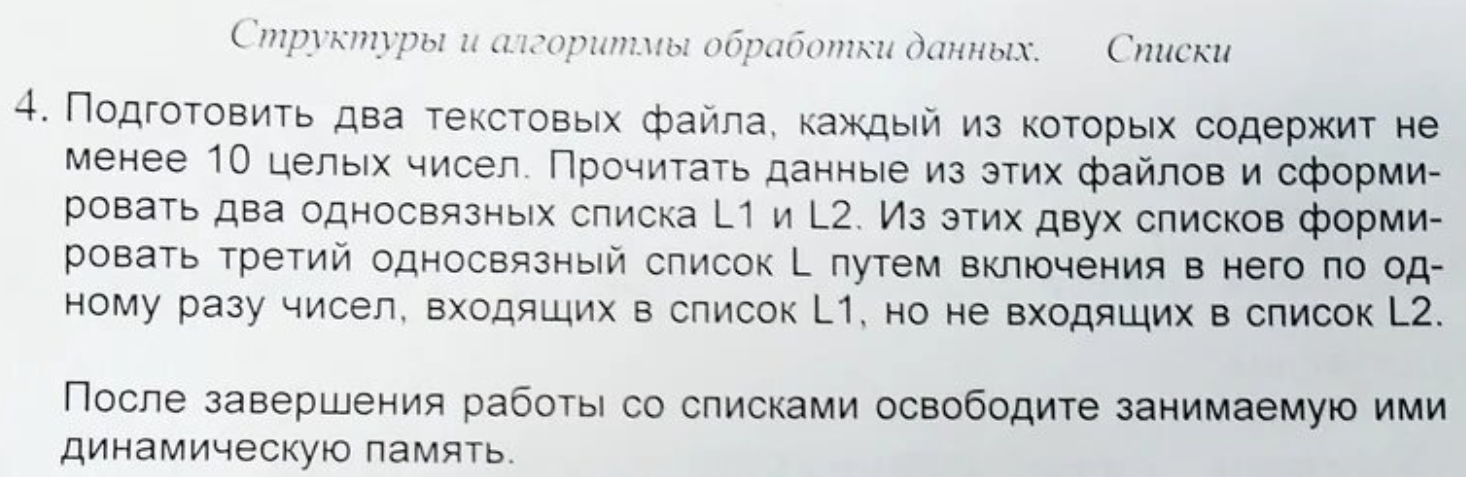
«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г.

Томск 2019

**Цель работы:**

Получить первичные навыки методов обработки абстрактных данных: связные списки.

**Задание:**



**Алгоритм:**

1. Считываем строки L1 и L2 из файлов.
2. Вставляем последовательно элементы в файл L из файла L1 и параллельно сравниваем элемент L1 с каждым элементом L2. Если L1[i]!=**∀**L2[j] вписываем L1[i] в новый файл L
3. Выводим список.
4. Удаляем все элементы списков и сами списки.

**Решение:**#include <iostream>

#include <fstream>

using namespace std;

template<typename T>

class List {

public:

template<typename T>

class node {

public:

node \*pNext;

T data;

node(T data= T(), node \*pN=nullptr)

{

this->data = data;

this->pNext = pN;

}

};

int size;

node<T> \*head;

List();

~List();

void P\_back(T data);

void insertElem(T value, int index);

void popFront();

void removeElem(int index);

int getSize() { return size; };

T& operator[](const int index);

};

template<typename T>

List<T>::List()

{

size = 0;

head = nullptr;

}

template<typename T>

List<T>::~List()

{

while (size > 0)

{

popFront();

}

}

template<typename T>

void List<T>::P\_back(T data)

{

if (head == nullptr)

head = new node<T>(data);

else {

node<T> \*cur = this->head;

while (cur->pNext!= nullptr) {

cur = cur->pNext;

}

cur->pNext = new node<T>(data);

}

size++;

}

template<typename T>

void List<T>::insertElem(T value, int index)

{

if (index == 0) { //insert 1st elem

head = new node<T>(value, head);

size++;

}

else //insert >1st elem

{

node<T> \*prev = this->head;

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

prev = prev->pNext;

node<T> \*newNode = new node<T>(value, prev->pNext);

prev->pNext = newNode;

size++;

}

}

template<typename T>

void List<T>::popFront()

{

node<T> \*temp = head;

head = head->pNext;

delete temp;

size--;

}

template<typename T>

void List<T>::removeElem(int index)

{

if (index == 0) //remove 1st elem

popFront();

else //remove >1st elem

{

node<T> \*prev = this->head;

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

prev = prev->pNext;

node<T> \*Delete = prev->pNext;

prev->pNext = Delete->pNext;

delete Delete;

size--;

}

}

template<typename T>

T & List<T>::operator[](const int index)

{

int counter=0;

node<T> \*cur = this->head;

while (cur != nullptr) {

if (counter == index) {

return cur->data;

}

cur = cur->pNext;

counter++;

}

}

template <typename T>

void foo(List<T> L1, List<T>L2); //Функция с карточки

int main() {

system("chcp 1251");

List<int> list;

for(int i=0; i<10; i++)

list.P\_back(i);

for (int i = 0; i < list.getSize(); i++)

cout << list[i] << "\t";

cout<<endl;

cout <<"\n4-ый элемент : "<< list[3] << endl;

cout << "последний элемент : " << list[list.getSize() - 1] << endl;

cout << "\nвставим пятый элемент 228" << endl;

list.insertElem(228, 4);

for (int i = 0; i < list.getSize(); i++)

cout << list[i] << "\t";

cout << "\n\nудалим 8й, 1ый и 10ый элементы" << endl;

list.removeElem(7);

list.removeElem(0);

list.removeElem(8);

for (int i = 0; i < list.getSize(); i++)

cout << list[i] << "\t";

cout << endl << endl << endl;//По карточке

List<int> L1, L2;

foo(L1, L2);

system("pause");

return 0;

}

template <typename T>

void foo(List<T> L1, List<T>L2) {

int num, i=0, j=0;

fstream file1, file2, newfile;

file1.open("list1.txt", ios::in);

file2.open("list2.txt", ios::in);

newfile.open("newlist.txt", ios::out);

while (!file1.eof())

{

file1 >> num;

L1.P\_back(num);

}

while (!file2.eof())

{

file2 >> num;

L2.P\_back(num);

}

List<int> L;

List<int> temp, temp2;

temp.head = L1.head;

temp2.head = L2.head;

while (temp.head->pNext != NULL)

{

while (temp2.head->pNext != NULL)

{

if (j == L2.getSize()) break;

if (L1[i] != L2[j] && (j == L2.getSize()-1)) L.P\_back(L1[i]);

else if (L1[i] == L2[j]) break;

temp2.head = temp.head->pNext;

j++;

}

if (i == L1.getSize()) break;

j = 0;

i++;

temp.head = temp.head->pNext;

}

/\*for (int i=0; i<L1.getSize(); i++)

for (int j = 0; j < L2.getSize(); j++)

{

if ((j == L2.getSize()-1) && (L1[i]!=L2[j])) L.P\_back(L1[i]);

else if (L1[i] == L2[j]) break;

}

for (int i = 0; i < L.getSize(); i++)

{

newfile << L[i];

newfile << " ";

cout << L[i] << "\t";

}

\*/

i = 0;

temp.head = L.head;

while (temp.head->pNext != NULL)

{

if (i == L.getSize()) break;

newfile << L[i];

newfile << " ";

cout << L[i] << "\t";

i++;

temp.head = temp.head->pNext;

}

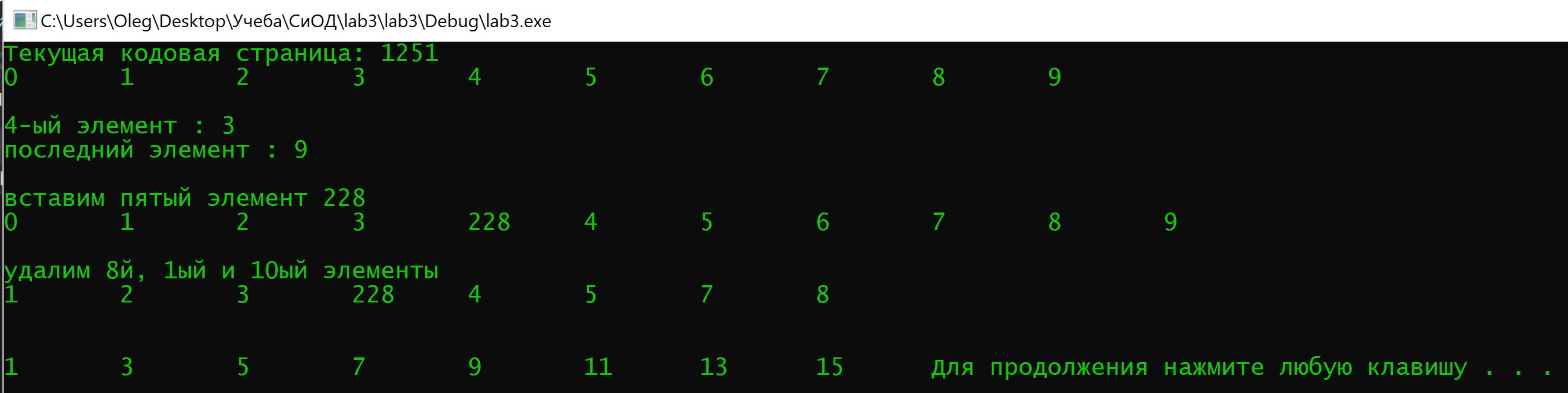
file1.close();

file2.close();

newfile.close();

}

**Результат работы:**



**Вывод:**Научился использовать абстрактный тип данных, связного списка, с учетом плюсов и минусов. Плюсы — эффективное добавление и удаление элементов, размер ограничен только объёмом памяти компьютера. Минусы — сложность прямого доступа к элементу, а именно определения физического адреса по его индексу в списке, на поля-указатели (указатели на следующий и предыдущий элемент) расходуется дополнительная память (в массивах, например, указатели не нужны).