Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра автоматизированных систем управления (АСУ)

**“Стеки и очереди”**

**Отчет по лабораторной работе №3**

**По дисциплине**

**«Структуры и алгоритмы обработки данных в ЭВМ»**

Студент гр. 431-3

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д.П. Андреев

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

Проверил: профессор кафедры АСУ, д.т.н.

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

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

Томск 2023

# 1.Задание на лабораторную работу

Подготовить два текстовых файла, каждый из которых содержит не менее 12 целых чисел. Прочитать данные из этих файлов и сформировать два односвязных списка L1 и L2. Из этих двух списков формировать третий односвязный список L путем включения в него по одному разу чисел, входящих одновременно в оба списка. Вывести на экран исходные списки и сформированный список L. После завершения работы со списками освободите занимаемую ими динамическую память.

# 2.Алгоритм решения задачи

Первый шаг это написание структуры узла List с полями информация и массив указателей. Далее прописываем операции инициализации, добавления узла, удаление узла, удаление корня , поиск элемента и вывод листа на экран. В основной части создаем три листа, два для записи информации из файлов. Для этого открываем файлы IN1 и IN2 и через цикл While с помощью операции добавления добавляем в списки элементы из файлов. Выводим получившиеся списки в консоль. После, путём перебора, находим повторяющиеся элементы в первых двух списках и добавляем их в третий список. Сортировка происходит следующим образом. Создаём цикл while (пока наш первый список не закончится), а в нём создаём копию второго списка и переходим во второй цикл while(пока второй цикл не закончится), где производим сравнение информационных ячеек. При положительном результате, добавляем элемент в третий список. После завершения сортировки мы выводим третий список в консоль. Освобождаем память.

# 3.Листинг программы

#include <iostream>

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

#include <windows.h>

#include <fstream>

using namespace std;

struct list//Структура узла

{

int info;//информация

struct list\* next;//Указатель на следующей эл-т

};

struct list\* Init()

{

struct list\* lst;

lst = (struct list\*)malloc(sizeof(struct list));// выделение памяти под корень списка

if (lst)

{

lst->info = 0;

lst->next = NULL; // это последний узел списка

return(lst);

}

else

{

cout << "ERROR" << endl;

return(0);

}

}

struct list\* AddElem(list\* lst, int number)//добовление узла в список

{

struct list\* temp, \* p;

temp = (struct list\*)malloc(sizeof(list));

if (temp)

{

p = lst->next; // сохранение указателя на следующий узел

lst->next = temp; // предыдущий узел указывает на создаваемый

temp->info = number; // сохранение поля данных добавляемого узла

temp->next = p; // созданный узел указывает на следующий элемент

return(temp);

}

else

{

cout << "ERROR" << endl;

return(0);

}

}

struct list\* DeletElem(list\* lst, list\* root)//Удаление узла

{

int flag = 0;

if (lst != NULL)

{

struct list\* temp;

temp = root;

while (temp->next != lst|| temp != NULL) // просматриваем список начиная с корня

{ // пока не найдем узел, предшествующий lst

if (temp->next != lst)

{

flag = 1;

}

temp = temp->next;

}

if (flag = 1)

{

temp->next = lst->next; // переставляем указатель

free(lst); // освобождаем память удаляемого узла

return(temp);

}

}

else

{

cout << "ERROR" << endl;

return(0);

}

}

void SearchElem(list\* lst, list\* root)//Поиск узла

{

int flag = 0;

if (lst != NULL)

{

struct list\* temp=root;

while (temp != NULL) // просматриваем список начиная с корня

{

if (temp->next == lst)

{

cout << "Элемент найден" << endl;

flag = 1;

}

temp = temp->next;

}

if (flag == 0)

{

cout << "Элемент не найден" << endl;

}

}

}

void PrintList(list\* lst)

{

if (lst != NULL)

{

struct list\* p;

p = lst;

while (p != NULL)

{

cout << p->info << endl;// вывод значения элемента p

p = p->next; // переход к следующему узлу

}

}

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

//list\* L1 = new list();

list\* L1 = Init();

list\* L2 = Init();

list\* L = Init();

ifstream in1("IN1.txt");

ifstream in2("IN2.txt");

int n;

while (in1 >> n)

{

AddElem(L1, n);

}

while (in2 >> n)

{

AddElem(L2, n);

}

cout << "Исходный список 1: " << endl;

PrintList(L1);

cout << "Исходный список 2: " << endl;

PrintList(L2);

while (L1 != NULL)

{

list\* p = L2;

while (p != NULL)

{

if (L1->info == p->info && p->info!=0)

{

AddElem(L, L1->info);

}

p = p->next;

}

L1 = L1->next;

}

cout << "Сформированный список: " << endl;

PrintList(L);

free(L1);

free(L2);

free(L);

return 0;

}

# 4.Пример решения

Входные данные можно увидеть на рисунке 4.1,4.2 .

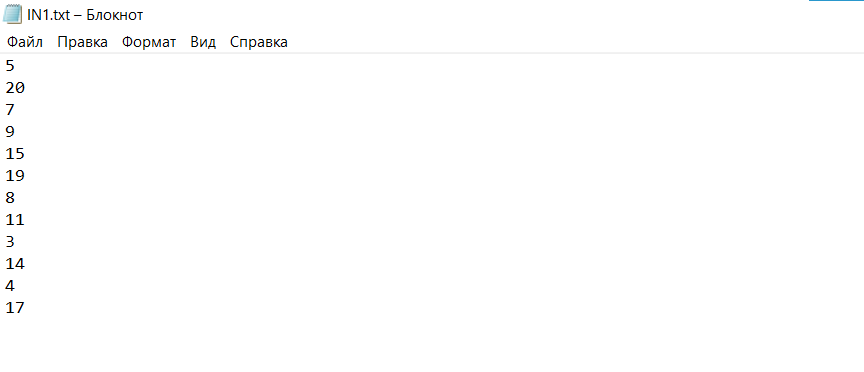
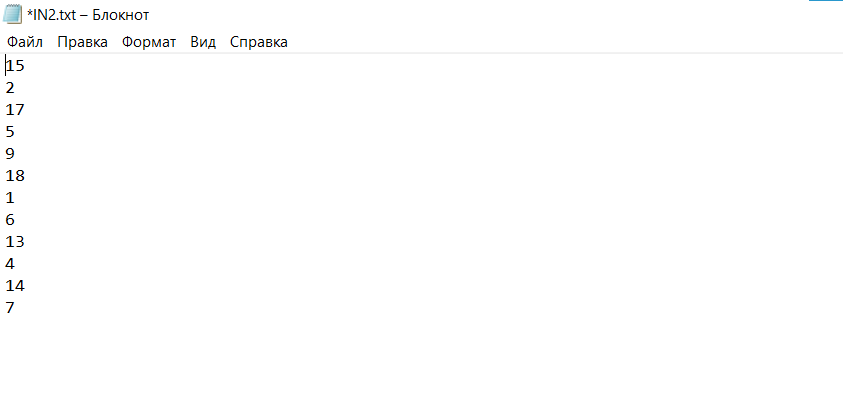


Рисунок 4.1— Входные данные первого файла

Рисунок 4.2— Входные данные второго файла

Результат работы программы можно увидеть на рисунке 4.3

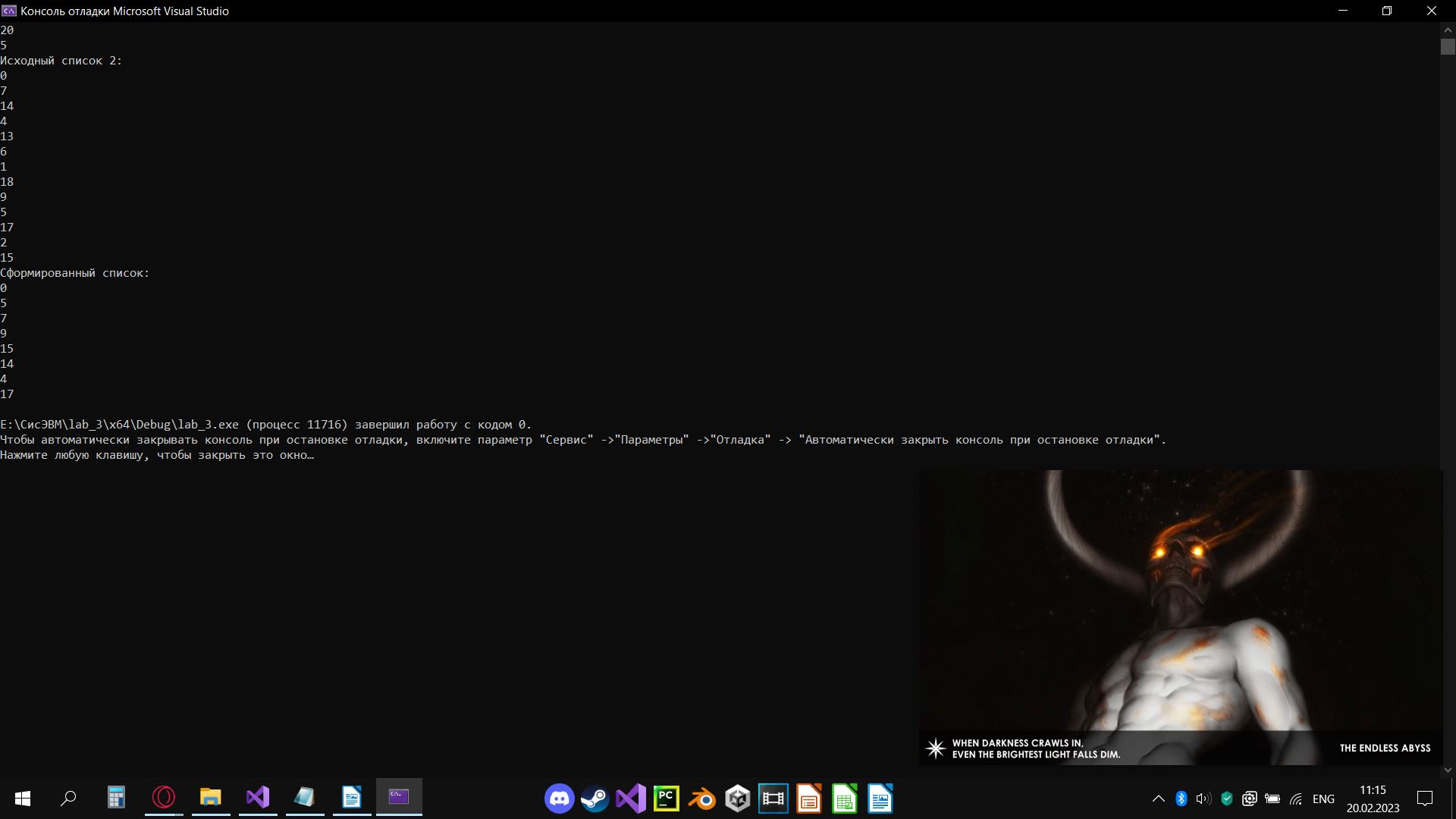


Рисунок 4.2 —Результат работы программы

# 5.Вывод

В результате лабораторной работы были принципы работы со структурой данных список.  При выполнении этой лабораторной работе были изучены и реализованы все функции для работы с этой структурой данных.