**Государственный комитет Российской Федерации по телекоммуникациям**

**Сибирский государственный университет телекоммуникаций и информатики**

**КУРСОВАЯ РАБОТА**

**По дисциплине «Программирование (часть 2)»**

**Двусвязные списки**

**Вариант № 4**

Работу выполнил:

                                                                   студент 1 курса

                                                                   Группы:

ПБТ-01

                                                                   Чульдум Саян Андреевич

                                                                   Работу проверил:

### Перцев Игорь Владимирович

### Работа защищена:

                                                                   «\_\_\_» \_\_\_\_\_\_\_\_\_\_2020г.

                                                                                                                                                                                      С оценкой «\_\_\_\_\_\_\_»

**г. Новосибирск**

**2021**

**CОДЕРЖАНИЕ**

1. Задание
2. Введение
3. Постановка комплекса задач
4. Проектный раздел
5. Результаты тестирования и выполнения задания**ЗАДАНИЕ**

Разработать программу для создания и работы с двусвязным списком, состоящим из структур. Для работы со списком создать меню со следующими пунктами:

1. Создание списка.

2. Просмотр списка.

3. Добавление в список новой записи.

4. Поиск и корректировка записи в списке.

5. Удаление записи из списка.

6. Сохранение списка в файле.

7. Загрузка списка из файла.

8. Выход.

Структура содержит название книги, Ф.И.О. автора, год издания. Удалять книги заданного автора. Изменять год издания по введённому названию книги. Добавлять новые записи перед последней.

**ВВЕДЕНИЕ**

Работа посвящена разработке простой программы на языке программирования высокого уровня C.

Целью работы является закрепление основ и углубление изученный знаний приемов программирования на языке C, а также получение практических навыков в создании программного продукта.

Задачей курсовой работы является создание программы, которая создает и обрабатывает двусвязный список структур. Работа с программой будет осуществляться с помощью меню.

1. **ПОСТАНОВКА КОМПЛЕКСА ЗАДАЧ**

Структура – это набор элементов разных типов. Элементами структуры могут быть базовые типы, массивы, указатели, структуры и т.д. Элементы структуры вместе с их описанием называются полями. Над полями можно выполнять действия, допустимые для данных этого типа.

Список – это последовательность структур, каждая из которых содержит ссылку, связывающую её с другой структурой. Для организации списков используются структуры, состоящие из двух смысловых частей – информационной и дополнительной. Информационная часть содержит подлежащую обработке информацию, в дополнительной находятся указатели на последующую или предыдущую структуру списка. В двусвязном списке каждая структура содержит две ссылки: на предыдущую и последующую структуры. Таким образом, по списку можно перемещаться от начала к концу и от конца к началу. Для доступа к началу и концу списка должны быть известны их адреса, которые могут сохраняться в переменных типа указатель.

Задача может быть разбита на несколько подзадач:

* создание списка;
* вывод содержимого списка;
* добавление в список новых элементов;
* корректировка списка;
* удаление записи из списка;
* сохранение списка в файле;
* загрузка списка из файла;
* организация интерфейса.

Организацию интерфейса логично расположить в основной программе, а подзадачи работы со списком оформить в виде функций, вызываемых из основной программы в зависимости от действий пользователя.

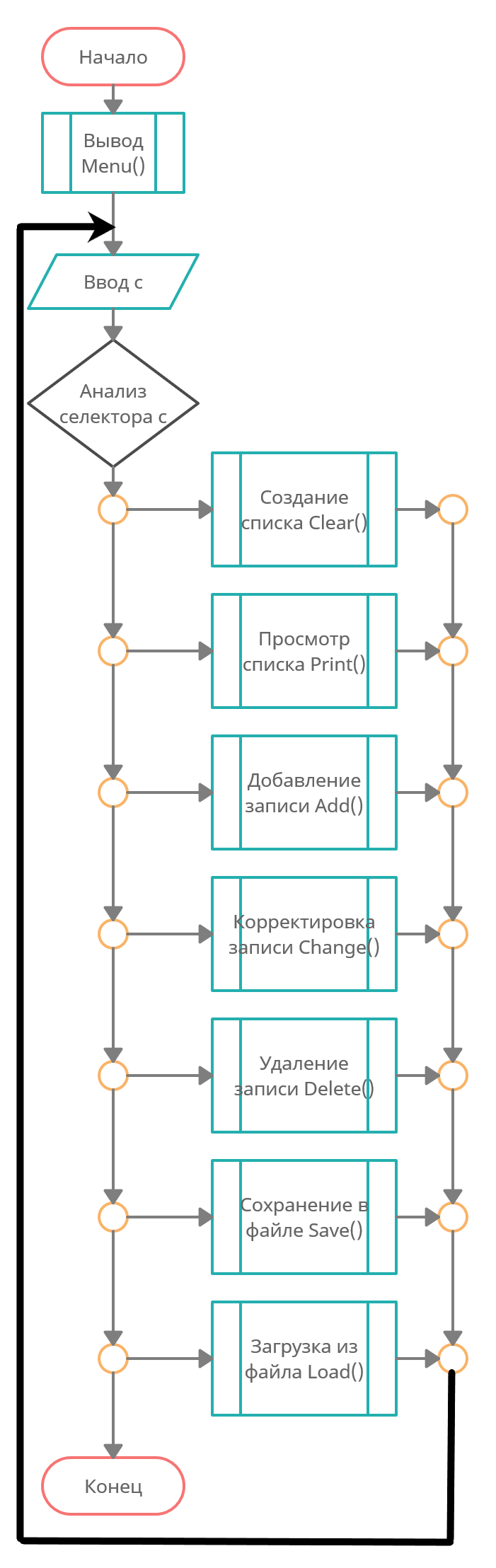
****

Рис. 1. Блок-схема функционирования системы

1. **ПРОЕКТИРОВАНИЕ**

При запуске программы, на экран терминала выводится список доступных опций. Пользователь должен выбрать одну опцию, которая выполняет ту или иную функцию:

1. Создание списка.
2. Просмотр списка.
3. Добавление в список новой записи.
4. Поиск и корректировка записи в списке.
5. Удаление записи из списка.
6. Сохранение списка в файле.
7. Загрузка списка из файла.
8. Выход.

После обработки выбора пользователя управление возвращается в основную программу.

Каждый элемент двусвязного списка содержит следующие поля:

* title – название книги;
* name – фамилия автора;
* year – год издания;
* prev – указатель на предыдущий элемент;
* next – указатель на следующий элемент.

Для работы со списком имеются следующие указатели:

* head – указатель на первый элемент списка;
* tail – указатель на последний элемент списка.

**Описание отдельных функций**

Функция Menu() выводит меню на экран.

Функция Clear() очищает список, т.е. удаляет все элементы списка из динамической памяти и обнуляет указатели на начало и конец списка.

Функция Print() выводит список на экран. Каждый элемент списка выводится в отдельной строке.

Функция Add() добавляет элемент перед последним элементом списка. Выделяет для элемента списка динамическую память, запрашивает ввод с клавиатуры всех полей элемента списка и настраивает связи между элементами списка.

Функция Change() изменяет год издания по названию книги. Запрашивает ввод названия книги и, в случае наличия такого названия в списке, запрашивает ввод года издания.

Функция Delete() удаляет из списка. Запрашивает ввод фамилии автора и, в случае наличия в списке изданий с заданным автором, удаляет запись. При этом при удалении элемента происходит перенастройка связей между элементами списка, соседними с удаляемым.

Функция Save() сохраняет список в файле inf.dat. Записи списка сохраняются в файле построчно.

Функция Load() загружает список из файла inf.dat. Перед загрузкой список уничтожается из динамической памяти.

**Модуль основной программы**

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

#include <string.h>

const char FileName[] = "inf.dat"; // Имя файла

// Структура элемента двусвязного списка

struct Inf

{

char title[30]; // Название книги

char name[20]; // Фамилия автора

int year; // Год издания

Inf\* prev; // Указатель на предыдущий элемент

Inf\* next; // Указатель на следующий элемент

};

Inf\* head = NULL; // Указатель на первый элемент списка

Inf\* tail = NULL; // Указатель на последний элемент списка

void Menu();

void Clear();

void Print();

void Add();

void Change();

void Delete();

void Save();

void Load();

void main()

{

Menu();

char c;

while (1) {

c = \_getch();

switch (c) {

case '1': // Создание списка

Clear();

puts("The list was created");

break;

case '2': // Просмотр списка

puts("The list:");

Print();

break;

case '3': // Добавление в список новой записи

Add();

break;

case '4': // Поиск и корректировка записи в списке

Change();

break;

case '5': // Удаление записи из списка

Delete();

break;

case '6': // Сохранение списка в файле

Save();

break;

case '7': // Загрузка списка из файла

Load();

break;

case '8': // Выход

return;

}

}

}

// Выводит меню на экран

void Menu() {

puts("1 - Create the list");

puts("2 - Print the list");

puts("3 - Add a new note");

puts("4 - Change a note");

puts("5 - Delete a note");

puts("6 - Save the list");

puts("7 - Load the list");

puts("8 - Exit\n");

}

// Очищает список

void Clear() {

Inf\* p = head;

while (p != NULL) {

Inf\* q = p;

p = p->next;

free(q); // Удаляем элемент списка из динамической памяти

}

head = NULL;

tail = NULL;

}

// Выводит список на экран

void Print()

{

Inf\* p = head;

while (p != NULL) { // Цикл по всем элементам списка

printf("%30s ", p->title);

printf("%20s ", p->name);

printf("%10d\n", p->year);

p = p->next;

}

}

// Добавляет новую запись перед последней

void Add()

{

Inf\* p = (Inf\*) malloc(sizeof(Inf)); // Выделяем динамическую память

printf("Book`s name: "); // Вводим все поля элемента списка

scanf("%s", p->title);

printf("Author`s lastname: ");

scanf("%s", p->name);

printf("The year of publishing: ");

scanf("%d", &p->year);

p->prev = NULL;

p->next = NULL;

if (head == NULL) // Если список пустой

head = tail = p;

else if (head->next == NULL) { // Если одна запись в списке

p->next = head;

head->prev = p;

head = p;

}

else {

p->prev = tail->prev;

p->next = tail;

p->prev->next = p;

tail->prev = p;

}

}

// Изменяет год издания по названию книги

void Change()

{

char title[30];

printf("Name: "); // Вводим название книги

scanf("%s", title);

Inf\* p = head;

int found = 0;

while (p != NULL) { // Цикл по всем элементам списка

if (!strcmp(p->title, title)) {

printf("The year of publishing: ");

scanf("%d", &p->year);

found = 1;

break;

}

p = p->next;

}

if (found == 0)

puts("Cannot find the book");

else

puts("The publishing year has been changed");

}

// Удаляет из списка книги заданного автора

void Delete()

{

char name[20];

printf("Enter author`s lastname: ");

scanf("%s", name);

Inf\* p = head;

int found = 0;

while (p != NULL) {

if (!strcmp(p->name, name)) {

// Если удаляемый элемент является одновременно первым и последним

if (p->prev == NULL && p->next == NULL) {

free(p); // Удаляем элемент списка из динамической памяти

head = NULL;

tail = NULL;

break;

}

// Если удаляем первый элемент

else if (p->prev == NULL) {

Inf\* q = p;

p = p->next;

free(q); // Удаляем элемент списка из динамической памяти

p->prev = NULL;

head = p;

}

// Если удаляем последний элемент

else if (p->next == NULL) {

Inf\* q = p;

p = p->prev;

free(q); // Удаляем элемент списка из динамической памяти

p->next = NULL;

tail = p;

break;

}

// Если удаляем элемент из середины списка

else {

p->prev->next = p->next;

p->next->prev = p->prev;

Inf\* q = p;

p = p->next;

free(q); // Удаляем элемент списка из динамической памяти

}

found = 1;

}

else

p = p->next;

}

if (found == 0)

puts("Editions not found");

else

puts("Editions deleted");

}

// Сохраняет список в файле

void Save()

{

FILE\* f; // файл

f = fopen(FileName, "w"); // открываем файл для записи

Inf\* p = head;

while (p != NULL) { // Цикл по всем элементам списка

fprintf(f, "%s %s %d\n", p->title, p->name, p->year);

p = p->next;

}

fclose(f);

printf("The list saved in %s\n", FileName);

}

// Загружает список из файла

void Load()

{

Clear();

FILE\* f; // файл

f = fopen(FileName, "r"); // открываем файл для чтения

while (!feof(f)) {

Inf\* p = (Inf\*) malloc(sizeof(Inf)); // Выделяем динамическую память

fscanf(f, "%s", p->title); // Вводим все поля элемента списка

if (feof(f))

break;

fscanf(f, "%s", p->name);

fscanf(f, "%d", &p->year);

p->prev = NULL;

p->next = NULL;

if (head == NULL) // Если список пустой

head = tail = p;

else {

tail->next = p;

p->prev = tail;

tail = p;

}

}

fclose(f);

printf("The list loaded from %s\n", FileName);

}

1. **РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ ПРОГРАММЫ**

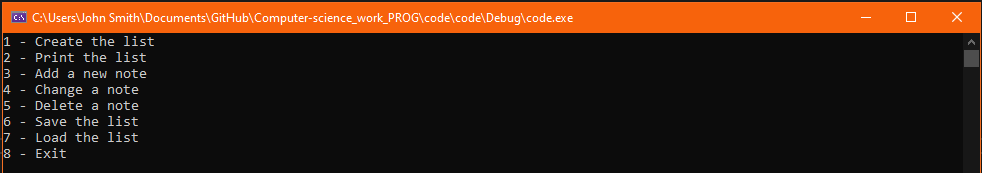
****

Рис. 2. Список меню

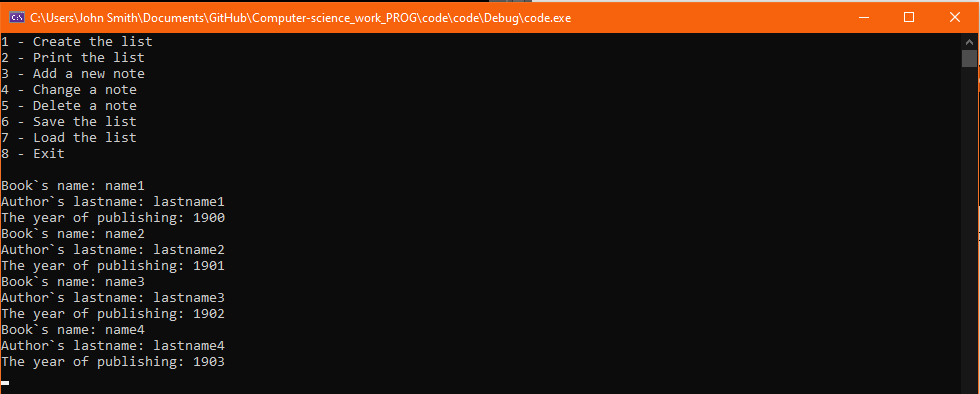


Рис. 3. Добавление 4-х элементов

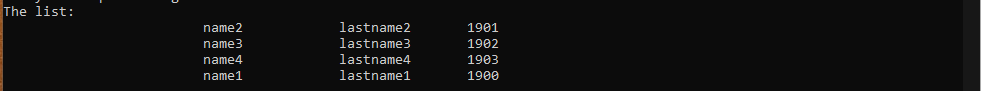


Рис. 4. Просмотр списка



Рис. 5. Изменение одной записи

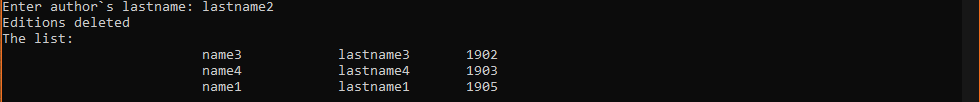


Рис. 6. Удаление одной записи