# Лабораторная работа № 7

«Программирование линейных списков на языке С/С++»

# Цель работы

Изучение списковых структур данных и приобретение навыков разработки и отладки программ, использующих динамическую память. Исследование особенностей организации списков средствами языка С/С++.

# Ход выполнения работы

# Задание

Представить одну из приведенных ниже таблиц в виде линейного списка L, элементами которого являются строки таблицы. Написать функции организации, добавления элемента в список, исключения элемента из списка, просмотра списка, а также одну из функций в соответствии с вариантом, приведенным ниже. Значения и количество записей в таблице студент выбирает самостоятельно. Исходные данные после организации списка должны сохраняться в файле и при повторном запуске программы считываться из файла. Количество строк таблицы не задается.

# Описание алгоритма решения задания

# Определение входных и выходных данных

В главном меню программы от пользователя требуется ввести число от 0 до 8. При вводе любого другого числа программа выводит ошибку.

При создании элемента очереди от пользователя требуется ввести ФИО (строка), Год рождения (число), Пол (символ), Семейное положение (строка), Количество детей (число), Оклад (число).

# Структурная схема алгоритма

Структурные схемы алгоритма представлены в Приложении Б на Рисунках Б.8 - Б.11.

# Исходный код программы (ЯП С, ОС GNU/Linux Ubuntu, компилятор gcc)

* + - 1. Файл main.c

///////////////////////////////////

// Блок препроцессорных директив //

///////////////////////////////////

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "tables.h"

#include "files.h"

#define MENU\_EXIT 0

#define MENU\_LOAD\_DATA 1

#define MENU\_SAVE\_DATA 2

#define MENU\_ORG 3

#define MENU\_ADD\_ONE 4

#define MENU\_DEL 5

#define MENU\_PRINT\_TBL 6

#define MENU\_SORT\_LIST 7

#define MENU\_SADD\_ONE 8

#define DEFAULT\_FILENAME ".7.database"

///////////////////////////////////

// Прототипы функций //

///////////////////////////////////

void menu\_Print ();

t\_row \*menu\_OrgTable();

void menu\_DelElem (t\_row \*\*head);

void menu\_AddOne (t\_row \*\*head);

void menu\_SAddOne (t\_row \*\*head);

void menu\_SaveTable (const t\_row \*head, const char \*name);

void menu\_LoadTable (t\_row \*\*head, const char \*name);

///////////////////////////////////

// Основная функция //

///////////////////////////////////

int main ()

{

// Начальный этап

t\_row \*head = NULL;

int u\_answ = -1;

// Основной алгоритм

while (u\_answ != MENU\_EXIT)

{

menu\_Print();

do {

printf("$: ");

fseek(stdin, 0, SEEK\_END);

u\_answ = getc(stdin);

} while (u\_answ == '\n');

switch (u\_answ -= (int) '0')

{

case MENU\_SAVE\_DATA:

menu\_SaveTable (head, DEFAULT\_FILENAME);

break;

case MENU\_LOAD\_DATA:

menu\_LoadTable(&head, DEFAULT\_FILENAME);

break;

case MENU\_ORG:

printf("Начато удаление старой таблицы...");

row\_del\_rec(head); head = 0;

printf("УДАЧА\n");

printf("Организация новой таблицы...\n\n");

head = menu\_OrgTable();

break;

case MENU\_PRINT\_TBL:

printf("Начата печать данных в табличном виде.\n\n");

t\_print(head);

break;

case MENU\_DEL:

printf("Запущена процедура удаления элемента таблицы...\n\n");

menu\_DelElem(&head);

break;

case MENU\_ADD\_ONE:

printf("Запуск функции добавления одного элемента в конец таблицы...\n\n");

menu\_AddOne(&head);

break;

case MENU\_SORT\_LIST:

printf("Попытка отсортировать таблицу...");

table\_sort(head);

printf("УДАЧА\n");

break;

case MENU\_SADD\_ONE:

printf("Запуск функции добавления одного элемента...\n\n");

menu\_SAddOne(&head);

break;

case MENU\_EXIT:

printf("Выход из программы...\n");

break;

default:

printf("\033[41;1;37mOШИБКА\033[0m: Действие \"%d\" неизвестно программе. Повторите ввод.\n", u\_answ);

break;

}

}

// Завершительный этап

row\_del\_rec(head);

head = NULL;

return 0;

}

///////////////////////////////////

// Определение функций //

///////////////////////////////////

void menu\_Print ()

{

printf("( %d ) Выйти из программы.\n" , MENU\_EXIT );

printf("( %d ) Загрузить данные из файла (по умолчанию %s).\n", MENU\_LOAD\_DATA, DEFAULT\_FILENAME);

printf("( %d ) Сохранить данные в файл (по умолчанию %s).\n", MENU\_SAVE\_DATA, DEFAULT\_FILENAME);

printf("( %d ) Удалить старую и Организовать новую таблицу.\n", MENU\_ORG );

printf("( %d ) Добавить один элемент в конец таблицы.\n" , MENU\_ADD\_ONE );

printf("( %d ) Удалить элемент.\n" , MENU\_DEL );

printf("( %d ) Вывести таблицу.\n" , MENU\_PRINT\_TBL);

printf("( %d ) Отсортировать данные в таблице (по ФИО).\n" , MENU\_SORT\_LIST);

printf("( %d ) \*my\* Вставить элемент.\n" , MENU\_SADD\_ONE );

printf("\n");

}

t\_row \*menu\_OrgTable ()

{

unsigned long int counter = 1;

unsigned int answer = 0;

t\_row \*head = 0, \*curr\_row = 0;

t\_data d;

printf("Создадим первый элемент!\n");

printf("Введите ФИО человека: "); fseek(stdin, 0, SEEK\_END); scanf("%s", d.FIO);

printf("Введите год рождения человека: "); scanf("%ld", &d.year);

printf("Введите пол человека: "); scanf("%s", d.gender);

printf("Введите семейное положение человека: "); scanf("%s", d.status);

printf("Введите количество детей в семье: "); scanf("%lu", &d.chcnt);

printf("Введите оклад: "); scanf("%lu", &d.salary);

head = curr\_row = row\_create(d);

printf("Продолжить ввод? [Y/n] "); getchar(); answer = getchar();

while (answer != 'n' && answer != 'N' && answer != 'н' && answer != 'Н')

{

printf("\n\n<===================>\n\n");

printf("Создадим %lu элемент таблицы!\n", ++counter);

printf("Введите ФИО человека: "); fseek(stdin, 0, SEEK\_END); scanf("%s", d.FIO);

printf("Введите год рождения человека: "); scanf("%ld", &d.year);

printf("Введите пол человека: "); scanf("%s", d.gender);

printf("Введите семейное положение человека: "); scanf("%s", d.status);

printf("Введите количество детей в семье: "); scanf("%lu", &d.chcnt);

printf("Введите оклад: "); scanf("%lu", &d.salary);

curr\_row->next = row\_create(d);

curr\_row->next->prev = curr\_row;

curr\_row = curr\_row->next;

printf("Продолжить ввод? [Y/n] "); getchar(); answer = getchar();

}

printf("Ввод завершён. Добавлено %lu элемент(-a,-ов). Возврат в меню.\n", counter);

return head;

}

/\*\*

\* @brief Запуск из меню функции удаления элемента.

\*

\* @param head Указатель на начало списка.

\*/

void menu\_DelElem (t\_row \*\*head)

{

char FIO[FIO\_LENGHT\_ \* MAX\_CHAR\_SIZE\_];

printf("Введите ФИО удаляемого человека:\n");

printf("$: "); scanf("%s", FIO);

t\_row \*row = searchRowByFIO(\*head, FIO);

if (row)

{

printf("Найдено поле:\n");

if (row->prev)

{

if (row->prev->prev)

printf("...\n");

print\_row(0, row->prev->data);

}

print\_row(0, row->data);

if (row->next)

{

print\_row(0, row->next->data);

if (row->next->next)

printf("...\n");

}

unsigned int answer;

printf("Вы действительно хотите удалить дайнный элемент? [y/N] "); getchar(); answer = getchar();

printf("Удаление элемента...");

if (answer == 'Y' || answer == 'y')

{

if (\*head == row)

{

\*head = row->next;

}

row\_del(row);

printf("УДАЧА\n");

}

else

printf("ОТКЛОНЕНО\n");

}

else

{

printf("Указанный элемент не найден.\n");

}

}

/\*\*

\* @brief Запуск из меню функции добавления одного элемента в список.

\*

\* @param head Указатель на начало списка.

\*/

void menu\_AddOne (t\_row \*\*head)

{

t\_data d;

printf("Введите ФИО человека: "); fseek(stdin, 0, SEEK\_END); scanf("%s", d.FIO);

printf("Введите год рождения человека: "); scanf("%ld", &d.year);

printf("Введите пол человека: "); scanf("%s", d.gender);

printf("Введите семейное положение человека: "); scanf("%s", d.status);

printf("Введите количество детей в семье: "); scanf("%lu", &d.chcnt);

printf("Введите оклад: "); scanf("%lu", &d.salary);

printf("Попытка добавления в конец файла...");

if (!\*head)

{

printf("\nТаблица пуста. Попытка добавить первую запись...");

\*head = row\_create(d);

}

else

{

t\_row \*tmp1 = \*head, \*tmp2 = row\_create(d);

while (tmp1->next)

tmp1 = tmp1->next;

tmp1->next = tmp2;

tmp2->prev = tmp1;

}

printf("УДАЧА\n");

}

void menu\_SAddOne (t\_row \*\*head)

{

t\_data d;

printf("Введите ФИО человека: "); fseek(stdin, 0, SEEK\_END); scanf("%s", d.FIO);

printf("Введите год рождения человека: "); scanf("%ld", &d.year);

printf("Введите пол человека: "); scanf("%s", d.gender);

printf("Введите семейное положение человека: "); scanf("%s", d.status);

printf("Введите количество детей в семье: "); scanf("%lu", &d.chcnt);

printf("Введите оклад: "); scanf("%lu", &d.salary);

printf("Попытка добавления в таблицу...");

SaddRowToTable(head, &d);

printf("УДАЧА\n");

}

void menu\_SaveTable(const t\_row \*head, const char \*name)

{

if (!head)

{

printf("Ошибка: таблицы не существует.\n");

return;

}

FILE \*f = fopen(DEFAULT\_FILENAME, "w");

if (!f)

{

printf("Ошибка: файла сохранений не существует.\n");

return;

}

while (head)

{

file\_saveData(f, &(head->data));

head = head->next;

}

fclose(f);

return;

}

void menu\_LoadTable (t\_row \*\*head, const char \*name)

{

row\_del\_rec(\*head);

\*head = NULL;

FILE \*f = fopen(name, "r");

if (!f)

{

printf("Ошибка: файла сохранений не существует.\n");

return;

}

t\_data tmp\_d;

t\_row \*tmp\_l;

while (file\_getData(f, &tmp\_d))

{

if (!\*head)

{

\*head = row\_create(tmp\_d);

tmp\_l = \*head;

}

else

{

tmp\_l->next = row\_create(tmp\_d);

tmp\_l->next->prev = tmp\_l;

tmp\_l = tmp\_l->next;

}

}

fclose(f);

return;

}

* + - 1. Файл tables.h

#ifndef TABLES\_H\_

#define TABLES\_H\_ 1

#ifndef MAX\_CHAR\_SIZE\_

#define MAX\_CHAR\_SIZE\_ 6

#endif

#define FIO\_LENGHT\_ 1024

#define STATUS\_LENGHT\_ 512

typedef struct T\_DATA {

char FIO[FIO\_LENGHT\_ \* MAX\_CHAR\_SIZE\_ + 1]; // ФИО

long int year; // Год рождения

char gender[MAX\_CHAR\_SIZE\_ + 1]; // Пол

char status[STATUS\_LENGHT\_ \* MAX\_CHAR\_SIZE\_ + 1]; // Семейное положение

unsigned long int chcnt; // Количество детей

unsigned long int salary; // Оклад

} t\_data;

typedef struct T\_ROW {

t\_data data;

struct T\_ROW \*next;

struct T\_ROW \*prev;

} t\_row;

///////////////////////

// Прототипы функций //

///////////////////////

t\_row \*row\_create (t\_data data);

void row\_del (t\_row \*row);

void print\_row (const unsigned long int c, const t\_data data);

void table\_del (t\_row \*row);

/////////////

// Функции //

/////////////

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @brief Инициализирует строку в таблице.

\*

\* @param data Данные, которые необходимо записать в новую строчку.

\*/

t\_row \*row\_create (t\_data data)

{

t\_row \*nrow = (t\_row \*) malloc(sizeof(t\_row));

nrow->next = nrow->prev = NULL;

nrow->data = data;

return nrow;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @brief Удаляет строку. Автоматически пересвязывает элементы.

\*

\* @param row Строчка таблицы для удаления.

\*/

void row\_del (t\_row \*row)

{

if (row->prev)

row->prev->next = row->next;

if (row->next)

row->next->prev = row->prev;

free(row);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @brief Печать строчки.

\*

\* @param c Порядковый номер.

\* @param data Данные для печати.

\*/

void print\_row (const unsigned long int c, const t\_data data)

{

printf("|%3lu| %20s | %4ld | %3s | %18s | %6lu | %8lu |\n",

c,

data.FIO,

data.year,

data.gender,

data.status,

data.chcnt,

data.salary

);

}

/\*\*

\* @brief Удаление текущего элемента и всех последующих.

\*

\* @param row Указатель на элемент, с которого необходимо удалять.

\*/

void row\_del\_rec (t\_row \*row)

{

if (!row) return;

row\_del\_rec(row->next);

free(row);

}

/\*\*

\* @brief Печать полноценной таблицы. Если таблица не содержит элементы, выводится ошибка.

\*

\* @param head Указатель на начало списка.

\*/

void t\_print (t\_row \*head)

{

printf("+---+----------------------+------+-----+--------------------+--------+----------+\n");

printf("| № | ФИО | ГРож | Пол | Семейное Положение | КолДет | Оклад |\n");

printf("+---+----------------------+------+-----+--------------------+--------+----------+\n");

if (!head)

{

printf("+ Данные отсутствуют +\n");

}

else

{

t\_row \*curr\_row = head;

unsigned long int counter = 0;

while (curr\_row)

{

print\_row(++counter, curr\_row->data);

curr\_row = curr\_row->next;

}

}

printf("+---+----------------------+------+-----+--------------------+--------+----------+\n");

printf("\n\n");

}

/\*\*

\* @brief Функция поиска элемента в списке по значению поля "FIO"

\*

\* @param head Указатель на начало списка.

\* @param FIO Значение FIO.

\* @return t\_row\* Указатель на найденный элемент. NULL, если элемент не был найден.

\*/

t\_row \*searchRowByFIO (t\_row \*head, char \*FIO)

{

t\_row \*curr\_row = head;

while (curr\_row)

{

if (!strcmp(curr\_row->data.FIO, FIO))

break;

curr\_row = curr\_row->next;

}

return curr\_row;

}

/\*\*

\* @brief Получить элемент по его id.

\*

\* @param head Указатель на голову списка.

\* @param id ID элемента.

\* @return \*t\_row Указатель на этот элемент. Если id недостижимЮ, возвращщает NULL.

\*/

t\_row \*getRowById (t\_row \*head, unsigned long int id)

{

for (unsigned long int curr\_id = 0; head && curr\_id < id; head = head->next, curr\_id++);

return head;

}

/\*\*

\* @brief Функция сортировки элементов в списке.

\*

\* @param head Указатель на начало списка.

\*/

void table\_sort (t\_row \*head)

{

t\_row \*tmp = 0; t\_data d;

unsigned long int elem\_count = 0;

for (tmp = head; tmp; elem\_count++, tmp = tmp->next);

for (int i = elem\_count - 1; i > 0; i--)

{

tmp = head;

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

{

if (strcmp(tmp->data.FIO, tmp->next->data.FIO) > 0)

{

d = tmp->data;

tmp->data = tmp->next->data;

tmp->next->data = d;

}

tmp = tmp->next;

}

}

}

/\*\*

\* @brief Вставка элемента row в список с головой head. Вставка происходит с учётом сортировки.

\*

\* @param head Указатель на начало списка.

\* @param row Указатель на элемент, который необходимо добавить.

\*/

void SaddRowToTable (t\_row \*\*head, t\_data \*d)

{

if (!(\*head))

{ // Если головы не существует

\*head = row\_create(\*d);

return;

}

if (strcmp((\*head)->data.FIO, d->FIO) >= 0)

{ // Если элемент нужно вставить перед головой

t\_row \*tmp = \*head;

\*head = row\_create(\*d);

(\*head)->next = tmp;

if ((\*head)->next) (\*head)->next->prev = \*head;

return;

}

t\_row \*tmp1;

for (tmp1 = \*head; tmp1->next && strcmp(tmp1->next->data.FIO, d->FIO) < 0; tmp1 = tmp1->next);

t\_row \*tmp2 = row\_create(\*d);

tmp2->next = tmp1->next;

tmp2->prev = tmp1;

tmp1->next = tmp2;

if (tmp2->next) tmp2->next->prev = tmp2;

return;

}

#endif

* + - 1. Файл files.h

#ifndef FILES\_H\_

#define FILES\_H\_ 1

size\_t file\_saveData(FILE \*f, t\_data \*d);

int file\_getData (FILE \*f, t\_data \*d);

size\_t file\_saveData (FILE \*f, t\_data \*d)

{

return fwrite(d, sizeof(t\_data), 1, f);

}

int file\_getData (FILE \*f, t\_data \*d)

{

return fread(d, sizeof(t\_data), 1, f);

}

#endif

# Описание тестовых примеров

Тестовые примеры продемонстрированы в Приложении В на Рисунках В.12 - В.17.

На Рисунке В.12 продемонстрировано главное меню программы.

На Рисунке В.13 продемонстрирована организация очереди. Был добавлен один элемент.

На Рисунке В.14 продемонстрирован процесс добавления одного элемента в конец очереди.

На Рисунке В.15 продемонстрирован процесс добавления одного элемента в очередь с сохранением сортировки.

На Рисунке В.16 продемонстрирован вывод очереди на экран.

На Рисунке В.17 продемонстрирована работоспособность функции сортировки очереди. Сначала очереди была отсортирована, а для проверки правильности работы программы очередь вновь была напечатана на экран.

# Вывод

При выполнении данной лабораторной работы были получены навыки разработки приложений, умеющих работать со ссылками и списками; написаны процедуры, осуществляющие главные операции над списками, а именно: Организация очереди, Добавление элементов в конец списка, Удаление первого и последнего элемента, Печать списка. Также были повторно закреплены навыки отладки программы, работы с файлами, а именно чтение данных из бинарного типизированного файла и запись в него. Полученные навыки в будущем помогут создавать новые таблицы данных, а также деревья, в том числе и бинарные.