# ЛАБОРАТОРНАЯ РАБОТА №5

# «ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ МОДУЛЕЙ»

Цель работы

Приобретение навыков разработки и отладки программ, использующих модули.

Задания

1. Программу из лабораторной работы №3 или №4 (вариант 12) разбить на несколько частей и оформить в виде модулей. Объединить полученные библиотеки в единый проект и выполнить компиляцию программы;
2. Разработать тестовые примеры и выполнить отладку программы;

Текст программы

В качестве программы для разбиения на модули выбрана программа из лабораторной работы №3. Произведено разделение на 4 модуля. Полученные файлы объединены в проект LAB5.dev.

Содержание файла main.cpp:

#include "prog\_work.h"

**int** main(**int** argc, **char** \*argv[]) {

work();

**return** 0;

}

Содержание файла prog\_work.h:

#ifndef PROG\_WORK\_H

#define PROG\_WORK\_H

#include <stdio.h>

#include <stdlib.h>

**struct** date { *//структура для даты*

**short** day; *//число*

**short** month; *//месяц*

**short** year; *//год*

};

**struct** NOTE { *//структура для данных о человеке*

**int** n; *//номер*

**char** name[50]; //имя

**char** num[15]; *//номер телефона*

**date** birth; *//дата рождения*

};

**struct** list { *//структура для списка*

**struct** **NOTE** inf;

**struct** **list** \*next;

};

**void** work(); *//прототип главной функции*

#endif

Содержание файла prog\_work.cpp:

#include "list\_create.h" *//создание листа*

#include "prog\_oper.h" *//операции с листом*

#include "prog\_file.h" *//ввод-вывод в файл*

**void** work() {

**list** \*start=NULL; *//указатель на начало списка*

**while** (1) {

*//текстовые указания к меню*

puts("----Menu----");

puts("1 - Organize list");

puts("2 - Output list");

puts("3 - Delete element");

puts("4 - Load list in file");

puts("5 - Downoad list from file");

puts("6 - Remove first element");

puts("0 - End");

**int** c; printf("\nEnter command - ");

scanf("%d",&c);

*//вызов функций меню*

**switch** (c) {

**case** 1: start=create\_list(start); **break**; *//организация списка*

**case** 2: print\_list(start); **break**; *//вывод содержимого списка*

**case** 3: start=delete\_data(start); **break**; *//удаление элемента*

**case** 4: load\_list(start); **break**; *//выгрузка в файл*

**case** 5: start=download\_list(start); **break**; *//загрузка из файла*

**case** 6: start=remove\_first(start); **break**; *//перемещение элемента*

**case** 0: start=clean\_list(start); **return** 1; *//конец программы*

**default**: puts("No such command\n"); *//команды не существует*

}

}

}

Содержание файла list\_create.h:

#ifndef LIST\_CREATE\_H

#define LIST\_CREATE\_H

#include "prog\_work.h"

**NOTE** add\_element(); *//заполнение инф. поля*

**list**\* create\_list(**list** \*start); *//организация списка*

**void** print\_list(**list** \*start); *//вывод списка*

#endif

Содержание файла list\_create.cpp:

#include "list\_create.h"

*//------заполнение информационного поля-----------*

**NOTE** add\_element() {

**NOTE** name;

printf("\n Number (0-exit) - "); *//номер*

scanf("%d",&name.n);

**if** (name.n==0) { *//конец записи*

printf("\n");

**return** name;

}

printf(" Name - "); *//имя*

scanf("%s",&name.name);

printf(" Phone number - "); *//телефон*

scanf("%s",&name.num);

printf(" Day - "); *//день*

scanf("%d",&name.birth.day);

printf(" Month - "); //месяц

scanf("%d",&name.birth.month);

printf(" Year - "); *//год*

scanf("%d",&name.birth.year);

**return** name;

}

*//---------организация списка--------------*

**list**\* create\_list(**list** \*start) {

printf("\n----List organiser----\n");

*//ввод первого элемента при отсутствии списка*

**if** (start == NULL) {

start=(**list**\*)malloc(**sizeof**(**list**));

start->inf = add\_element();

start->next = NULL;

**if** (start->inf.n==0)

**return**(NULL);

}

*//заполнение элементов*

**while** (1) {

**list** \*t;

t=(**list**\*)malloc(**sizeof**(**list**));

t->inf = add\_element();

t->next = NULL;

**list** \*end=start; *//указатель на конец списка*

**while** (end->next!=NULL) *//поиск конца*

end=end->next;

end->next=t;7

**if** (t->inf.n==0) { *//конец заполнения*

end->next=NULL;

**return** start;

}

}

}

*//------------вывод списка-----------------*

**void** print\_list(**list** \*start) {

printf("\n----List output----\n");

*//проверка наличия списка*

**if** (start==NULL) {

printf("No list found\n\n");

**return**; }

**list** \*t=start;

**while** (t!=NULL) {

printf(" Number - %d\n",t->inf.n);

printf(" Name - %s\n",t->inf.name);

printf(" Phone number - %s\n",t->inf.num);

printf(" Birthday - ");

**if** (t->inf.birth.day>9)

printf("%d.",t->inf.birth.day);

**else** printf("0%d.",t->inf.birth.day);

**if** (t->inf.birth.month>9)

printf("%d.",t->inf.birth.month);

**else** printf("0%d.",t->inf.birth.month);

printf("%d\n\n",t->inf.birth.year);

t=t->next;

}

**return**;

}

Содержание файла prog\_oper.h:

#ifndef PROG\_OPER\_H

#define PROG\_OPER\_H

#include "prog\_work.h"

**void** load\_list(**list** \*start); *//загрузка из файла*

**list**\* read\_element(**FILE** \*f, **list** \*t); *//чтение элемента*

**list**\* clean\_list(**list** \*start); *//очищение списка*

#endif

Содержание файла prog\_oper:

#include "prog\_oper.h"

*//---------удаление элемента--------------*

**list**\* delete\_data(**list** \*start) {

**list** \*end=start;

*//проверка наличия списка*

**if** (start==NULL) {

printf("\nNo list found\n\n");

**return** start;

}

*//проверка на >1 элементов*

**if** (start->next==NULL) {

printf("Last element deleted\n\n");

**return** NULL;

}

**while** (end->next->next!=NULL)

end=end->next;

end->next=NULL;

printf("Last element deleted\n\n");

**return** start;

}

*//---------перемещение первого элемента--------------*

**list**\* remove\_first(**list** \*start) {

*//проверка наличия списка*

**if** (start==NULL) {

printf("No list found\n\n");

**return** start;

}

*//проверка наличия >1 элементов*

**if** (start->next==NULL) {

printf("Nothing changed\n\n");

**return** start;

}

**list** \*t=start;

start=start->next;

**list** \*end=start;

**while** (end->next!=NULL)

end=end->next;

t->next=NULL;

end->next=t;

printf("Element removed\n\n");

**return** start;

}

*//---------очищение памяти--------------*

**list**\* clean\_list(list \***start**) {

**list** \*t=start;

**while** (start!=NULL) {

t=start;

start=start->next;

free(t);

}

**return** start;

}

Содержание файла prog\_file.h:

#ifndef PROG\_FILE\_H

#define PROG\_FILE\_H

#include "prog\_work.h"

#include "prog\_oper.h"

**list**\* delete\_data(**list** \*start); *//удаление элемента*

**list**\* remove\_first(**list** \*start); *//перемещение первого*

**list**\* download\_list(**list** \*start); *//загрузка из файла*

#endif

Содержание файла prog\_file.cpp:

#include "prog\_file.h"

*//------------запись в файл-----------------*

**void** load\_list(**list** \*start) {

*//проверка наличия списка*

**if** (start==NULL) {

printf("No list found\n\n");

**return**;

}

*//указатель на файл*

**FILE** \*f;

**if** ((f = fopen("data.txt","wt"))==NULL) {

fprintf(stderr,"Can't open output file \n");

**return**;

}

**list** \*t=start;

*//запись данных элемента списка*

**while** (t!=NULL) {

fprintf(f,"%d\n",t->inf.n);

fprintf(f,"%s\n",t->inf.name);

fprintf(f,"%s\n",t->inf.num);

fprintf(f,"%d\n",t->inf.birth.day);

fprintf(f,"%d\n",t->inf.birth.month);

fprintf(f,"%d\n",t->inf.birth.year);

t=t->next;

}

fclose(f);

printf("File loaded\n\n");

**return**;

}

*//------чтение элемента списка из файла-----------*

**list**\* read\_element(**FILE** \*f, **list** \*t) {

**if** (fscanf(f,"%d",&t->inf.n)==EOF) **return** NULL;

**if** (fscanf(f,"%s",&t->inf.name)==EOF) **return** NULL;

**if** (fscanf(f,"%s",&t->inf.num)==EOF) **return** NULL;

**if** (fscanf(f,"%d",&t->inf.birth.day)==EOF) **return** NULL;

**if** (fscanf(f,"%d",&t->inf.birth.month)==EOF) **return** NULL;

**if** (fscanf(f,"%d",&t->inf.birth.year)==EOF) **return** NULL;

t->next = NULL;

**return** t;

}

*//------загрузка списка из файла-----------*

**list**\* download\_list(**list** \*start) {

start=clean\_list(start);

*//указатель на файл*

**FILE** \*f;

**if** ((f = fopen("data.txt","rt"))==NULL) {

printf("Can't open output file \n");

**return** start;

}

*//чтение первого элемента*

start=(**list**\*)malloc(**sizeof**(**list**));

start=read\_element(f, start);

**if** (!start) {

fclose(f);

**return** NULL;

}

**list** \*t;

**list** \*end=start;

**while** (end->next!=NULL)

end=end->next;

*//чтение списка*

**while** (1) {

t=(**list**\*)malloc(**sizeof**(**list**));

t=read\_element(f, t);

*//конец файла*

**if** (!t) {

fclose(f);

printf("List downloaded\n\n");

**return** start;

}

end->next=t;

end=end->next;

}

}

Тестовые примеры

Проверка функций ввода и вывода списка. С помощью соответствующих команд меню был введён список, состоящий из двух элементов. Затем список был выведен на экран. Данные совпали с введёнными.

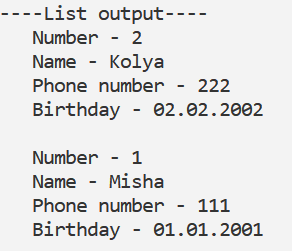


Рисунок 2 – Проверка вывода списка

Была вызвана функция перемещения первого элемента после последнего. В результате вывода на экран обновлённого списка элементы 1 и 2 поменялись позициями в списке.

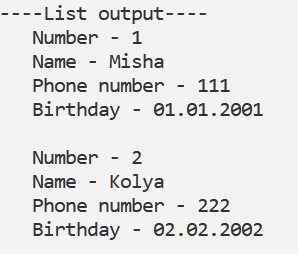


Рисунок 3 – Проверка перемещения первого элемента списка

Была вызвана функция выгрузки списка в файл. В результате в папке, где хранится программа, появился файл data.txt, содержащий список из программы.

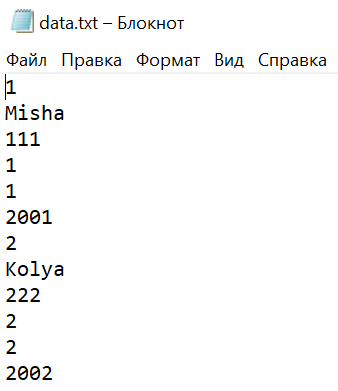


Рисунок 4 – Файл с выгруженным списком

Была вызвана функция удаления последнего элемента списка. В результате в списке остался лишь один элемент.

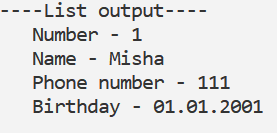


Рисунок 5 – Проверка удаления последнего элемента списка

Файл data.txt был вручную переписан. В него были введены значения двух новых элементов списка.

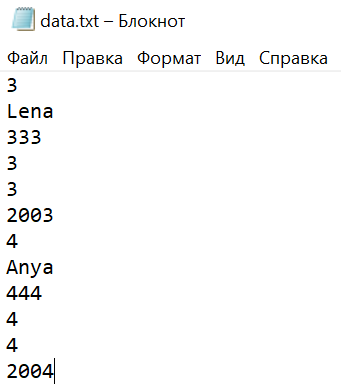


Рисунок 6 – Новые данные файла

Была вызвана функция загрузки списка из файла. В результате данные списка программы были заменены данными файла data.dat.

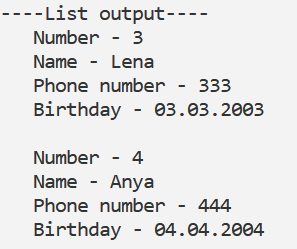


Рисунок 7 – Список, загруженный из файла

При вводе в меню команды 0 память, выделенная под список, была освобождена, программа была завершена.

Вывод

В ходе работы были освоены базовые навыки модульного программирования. В результате был написан программный проект на языке C, состоящий из ряда объединённых модулей и содержащий инструменты работы с модулями и библиотеками.