**Задание Б – «СПИСОК»**

**Часть Б1. Смоделировать двумя способами** новый тип «Список»:

1) на основе одномерного динамического массива;

А

Б

В

Г

Д

Запас1 – при удалении не выгодно каждый раз сдвигать все элементы в начало массива, эффективнее накопить сначала M «свободных» элементов.

Конец  
(индекс)

Начало  
(индекс)

Запас2 – при добавлении не выгодно каждый раз выделять новую память и копировать в нее массив, эффективнее выделять память с запасом M элементов.

2) на основе типизированного/двоичного файла.

А

Б

В

Г

Д

Е

Ё

Запас – при удалении не выгодно каждый раз сдвигать все элементы в начало файла, эффективнее накопить сначала M «свободных» элементов.

Конец  
(позиция в файле)

Начало  
(позиция в файле)

Ж

Для этого **создать модуль** и описать **новый тип данных** и его структуру (на русском языке и на Delphi/C, без классов! Только записи/структуры, массивы и файловый тип) и **базовые операции**:

- проверка на пустоту списка;

- создание пустого списка или списка из одного элемента (как удобнее);

- добавление элемента в список (в начало/в конец/после текущего);

- удаление элемента из списка (из начала/из конца/после текущего);

- сделать текущим элементом в списке (первый/ последний/ следующий/ предыдущий);

- считать значение текущего элемента из списка;

- вывод (дописывание) содержимого списка в текстовый лог-файл для контроля за состоянием списка после каждой операции с начала и до конца работы программы.

#pragma once

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <cstdio>

#include <stdlib.h>

//#define MoreFilesMOREMOREMORE

using namespace std;

struct TInfo {

char FirstName[30];

char LastName[30];

int Year;

int Weight;

int Height;

};

struct List {

FILE\* bin;

int dop1 = 0;

int n = 0;

};

void CorrectTInfo(TInfo &a); //Корректировка TInfo

bool Uslov(TInfo a); //Проверка условия

void FPrintTInfo(TInfo a, FILE\* buff); //Печать в поток

void FScanTInfo(TInfo &a, FILE\* txt); //Считать из поток

void ScanTInfo(TInfo &a); //Считать с консоли

bool IsNotEmpty(List a); //Проверка на пустоту

void CreateList(List &a, TInfo info, int argc, char\* argv[]); //Создание списка

void FirstElem(List &a); //Переход к первому элементу

bool NextElem(List &a); //Переход к следующему элементу

bool PreviousElem(List &a); //Переход к предыдущему элементу

void LastElem(List &a); //Переход к последнему элементу

TInfo getTInfo(List a); //Вернуть информацию

void PushBack(List &a, TInfo info, int argc, char\* argv[]); //Добавить в начало списка

void PopBack(List &a, TInfo info); //Добавить в конец

void MedBack(List &a, TInfo info); //Добавить после текущего

void DeleteFirst(List &a); //Удалить первый

void DeleteLast(List &a); //Удалить последний

void DeleteMedium(List &a); //Удалить после текущего

void LogFile(List a, int argc, char\* argv[]); //Напечатать в лог-файл

#include "FILE.h"

#include <time.h>

const TInfo nulll{ "","",-1,-1,-1 };

void CorrectTInfo(TInfo &a) {

time\_t t;

struct tm \*nabor;

time(&t);

nabor = localtime(&t);

if ((nabor->tm\_year + 1900 - a.Year) < 5) { a.Year = 2014; }

else if ((nabor->tm\_year + 1900 - a.Year) > 14) { a.Year = 2005; }

if (a.Weight < 1) { a.Weight = 1; }

else if (a.Weight > 200) { a.Weight = 200; }

if (a.Height < 1) { a.Height = 1; }

else if (a.Height > 200) { a.Height = 200; }

return;

}

bool Uslov(TInfo a) {

time\_t t;

struct tm \*nabor;

time(&t);

nabor = localtime(&t);

if (((nabor->tm\_year + 1900 - a.Year) >= 10) && ((nabor->tm\_year + 1900 - a.Year) <= 12) && (a.Height >= 155) && (a.Weight <= 45)) {

return true;

}

return false;

}

void FPrintTInfo(TInfo a, FILE\* buff) {

fprintf(buff, "FN:%s ", a.FirstName);

fprintf(buff, "LN:%s ", a.LastName);

fprintf(buff, "Y:%d ", a.Year);

fprintf(buff, "W:%d ", a.Weight);

fprintf(buff, "H:%d", a.Height);

return;

}

void FScanTInfo(TInfo &a, FILE\* txt) {

fscanf(txt, "%s", a.FirstName);

fscanf(txt, "%s", a.LastName);

fscanf(txt, "%d", &a.Year);

fscanf(txt, "%d", &a.Weight);

fscanf(txt, "%d", &a.Height);

return;

}

void ScanTInfo(TInfo &a) {

scanf("%30s", a.FirstName);

scanf("%30s", a.LastName);

scanf("%d", &a.Year);

scanf("%d", &a.Weight);

scanf("%d", &a.Height);

return;

}

bool IsNotEmpty(List a) {

return a.n;//Переполнение? Не, не слышали...

}

void CreateList(List &a, TInfo info, int argc, char\* argv[]) {

CorrectTInfo(info);

#ifndef MoreFilesMOREMOREMORE

a.bin = fopen(argv[1], "w+b");

#else

char name[16] = {};

scanf("%s", name);

a.bin = freopen(name, "w+b", a.bin);

#endif // !MoreFilesMOREMOREMORE

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

{

fwrite(&nulll, sizeof(TInfo), 1, a.bin);

}

fwrite(&info, sizeof(TInfo), 1, a.bin);

a.dop1 = 10;

a.n = 1;

fseek(a.bin, sizeof(TInfo)\*a.dop1, SEEK\_SET);

return;

}

void FirstElem(List &a) {

fseek(a.bin, sizeof(TInfo)\*a.dop1, SEEK\_SET);

return;

}

bool NextElem(List &a) {

int j = ftell(a.bin);

TInfo info;

fseek(a.bin, sizeof(TInfo), SEEK\_CUR);

fread(&info, sizeof(TInfo), 1, a.bin);

if (info.Height < 0) {

printf("Ошибка, попытка выйти за пределы списка.");

fseek(a.bin, j, SEEK\_SET);

return false;

}

fseek(a.bin, sizeof(TInfo) + j, SEEK\_SET);

return true;

}

bool PreviousElem(List &a) {

int j = ftell(a.bin);

TInfo info;

fseek(a.bin, j - sizeof(TInfo), SEEK\_SET);

fread(&info, sizeof(TInfo), 1, a.bin);

if (info.Height < 0) {

printf\_s("Ошибка, попытка выйти за пределы списка.");

fseek(a.bin, j, SEEK\_SET);

return false;

}

fseek(a.bin, j - sizeof(TInfo), SEEK\_SET);

return true;

}

void LastElem(List &a) {

fseek(a.bin, sizeof(TInfo)\*(a.dop1 + a.n - 1), SEEK\_SET);

return;

}

TInfo getTInfo(List a) {

int j = ftell(a.bin);

TInfo buff;

fread(&buff, sizeof(TInfo), 1, a.bin);

fseek(a.bin, j, SEEK\_SET);

return buff;

}

void PushBack(List &a, TInfo info, int argc, char\* argv[]) {

int j = ftell(a.bin);

CorrectTInfo(info);

if (a.dop1) {

fseek(a.bin, (--a.dop1) \* sizeof(TInfo), SEEK\_SET);

fwrite(&info, sizeof(TInfo), 1, a.bin);

a.n++;

fseek(a.bin, j, SEEK\_SET);

return;

}

TInfo temp;

FILE \* buff;

buff = fopen("bbbbb.bin", "wb");

for (int i = 0; i < 10; i++) { fwrite(&nulll, sizeof(TInfo), 1, buff); }

fwrite(&info, sizeof(TInfo), 1, buff);

fseek(a.bin, 0, SEEK\_SET); //fseek(a.bin, sizeof(TInfo)\*a.dop1, SEEK\_SET);

for (int i=0; i<a.n; i++)

{

fread(&temp, sizeof(TInfo), 1, a.bin);

fwrite(&temp, sizeof(TInfo), 1, buff);

}

fclose(a.bin);

fclose(buff);

remove(argv[1]);

rename("bbbbb.bin", argv[1]);

a.bin = fopen(argv[1], "r+b"); //

a.n++;

a.dop1 = 10;

fseek(a.bin, sizeof(TInfo) + j, SEEK\_SET);

}

void PopBack(List &a, TInfo info) {

int j = ftell(a.bin);

CorrectTInfo(info);

fseek(a.bin, (a.dop1 + a.n) \* sizeof(TInfo), SEEK\_SET);

fwrite(&info, sizeof(TInfo), 1, a.bin);

fseek(a.bin, j, SEEK\_SET);

a.n++;

}

void MedBack(List &a, TInfo info) {

int j = ftell(a.bin);

if ((a.dop1 + a.n - j / sizeof(TInfo)) < 2) { PopBack(a, info); return; }

CorrectTInfo(info);

TInfo buff1, buff2;

fseek(a.bin, sizeof(TInfo), SEEK\_CUR);

buff1 = getTInfo(a);

fwrite(&info, sizeof(TInfo), 1, a.bin);

buff2 = getTInfo(a);

while (buff1.Height > 0)

{

fwrite(&buff1, sizeof(TInfo), 1, a.bin);

fseek(a.bin, 0, SEEK\_CUR);

buff1 = buff2;

buff2 = getTInfo(a);

}

a.n++;

fseek(a.bin, j, SEEK\_SET);

return;

}

void DeleteFirst(List &a) {

int j = ftell(a.bin);

if (!IsNotEmpty(a)) {

//printf("Ошибка");

return;

}

fseek(a.bin, a.dop1 \* sizeof(TInfo), SEEK\_SET);

fwrite(&nulll, sizeof(TInfo), 1, a.bin);

a.n--;

if ((a.dop1++) \* sizeof(TInfo) == j) {

fseek(a.bin, 0, SEEK\_CUR);

return;

}

fseek(a.bin, j, SEEK\_SET);

return;

}

void DeleteLast(List &a) {

int j = ftell(a.bin);

if (!IsNotEmpty(a)) {

//printf("Ошибка");

return;

}

fseek(a.bin, (a.dop1 + a.n) \* sizeof(TInfo), SEEK\_SET);

fwrite(&nulll, sizeof(TInfo), 1, a.bin);

if ((a.dop1 + (a.n--)) \* sizeof(TInfo) == j) {

fseek(a.bin, j - sizeof(TInfo), SEEK\_SET);

return;

}

fseek(a.bin, j, SEEK\_SET);

return;

}

void DeleteMedium(List &a) {

int j = ftell(a.bin), p = j / sizeof(TInfo) - a.dop1 + 1;

TInfo buff1, buff2;

if (!NextElem(a)) {

//printf("Ошибка");

return;

}

fseek(a.bin, a.dop1 \* sizeof(TInfo), SEEK\_SET);

fread(&buff1, sizeof(TInfo), 1, a.bin);

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

{

buff2 = getTInfo(a);

fwrite(&buff1, sizeof(TInfo), 1, a.bin);

buff1 = buff2;

}

a.dop1++;

a.n--;

fseek(a.bin, j + sizeof(TInfo), SEEK\_SET);

}

void LogFile(List a, int argc, char\* argv[]) {

int j = ftell(a.bin);

FILE\* log;

TInfo buff;

log = fopen(argv[3], "r+t");

fseek(log, 0, SEEK\_END);

fseek(a.bin, (a.dop1 - 1) \* sizeof(TInfo), SEEK\_SET);

for (int i = a.dop1; i < a.dop1 + a.n; i++)

{

NextElem(a);

buff = getTInfo(a);

FPrintTInfo(buff, log);

fprintf(log, "\n");

}

fseek(a.bin, j, SEEK\_SET);

fprintf(log, "\n");

fclose(log);

system(argv[3]);

}

2) на основе типизированного/двоичного файла.

А

Б

В

Г

Д

Е

Ё

Запас – при удалении не выгодно каждый раз сдвигать все элементы в начало файла, эффективнее накопить сначала M «свободных» элементов.

Конец  
(позиция в файле)

Начало  
(позиция в файле)

Ж

Для этого **создать модуль** и описать **новый тип данных** и его структуру (на русском языке и на Delphi/C, без классов! Только записи/структуры, массивы и файловый тип) и **базовые операции**:

- проверка на пустоту списка;

- создание пустого списка или списка из одного элемента (как удобнее);

- добавление элемента в список (в начало/в конец/после текущего);

- удаление элемента из списка (из начала/из конца/после текущего);

- сделать текущим элементом в списке (первый/ последний/ следующий/ предыдущий);

- считать значение текущего элемента из списка;

- вывод (дописывание) содержимого списка в текстовый лог-файл для контроля за состоянием списка после каждой операции с начала и до конца работы программы.

#pragma once

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

//#define tinfo

using namespace std;

struct TInfo {

char FirstName[30];

char LastName[30];

int Year;

int Weight;

int Height;

};

struct List {

struct TInfo \*Mass;

int dop1=0;

int n=0;

int dop2=0;

};

bool IsNotEmpty(List a);

void CorrectTInfo(TInfo &a);

bool Uslov(TInfo a);

void FirstElem(List &a);

bool NextElem(List &a);

bool PreviousElem(List &a);

void LastElem(List &a);

void LogFile(List a, int argc, char\* argv[]);

void CreateList(List &a, TInfo info, int argc, char\* argv[]);

void PushBack(List &a, TInfo info, int argc, char\* argv[]);

void PopBack(List &a, TInfo info);

void MedBack(List &a, TInfo info);

void DeleteFirst(List &a);

void DeleteLast(List &a);

void DeleteMedium(List &a);

void FPrintTInfo(TInfo a, FILE\* buff);

void FScanTInfo(TInfo &a, FILE\* txt);

void ScanTInfo(TInfo &a);

TInfo getTInfo(List a);

#include "Mass.h"

#include <time.h>

const TInfo nulll{"","",-1,-1,-1};

bool IsNotEmpty(List a) {

return a.n;//Переполнение? Не, не слышали...

}

void CorrectTInfo(TInfo &a) {

time\_t t;

struct tm \*nabor;

time(&t);

nabor = localtime(&t);

if ((nabor->tm\_year + 1900 - a.Year) < 5) { a.Year = 2014; }

else if ((nabor->tm\_year + 1900 - a.Year) > 14) { a.Year = 2005; }

if (a.Weight < 1) { a.Weight = 1; }

else if (a.Weight > 200) { a.Weight = 200; }

if (a.Height < 1) { a.Height= 1; }

else if (a.Height > 200) { a.Height = 200; }

return;

}

bool Uslov(TInfo a) {

time\_t t;

struct tm \*nabor;

time(&t);

nabor = localtime(&t);

if (((nabor->tm\_year + 1900 - a.Year) >= 10) && ((nabor->tm\_year + 1900 - a.Year) <= 12) && (a.Height >= 155) && (a.Weight <= 45)) {

return true;

}

return false;

}

void FirstElem(List &a) {

int i = 0;

while (a.Mass[i].Height < 0) {

i++;

}

a.n = a.n + a.dop1 - i;

a.dop1 = i;

return;

}

bool NextElem(List &a) {

if (a.Mass[a.dop1+1].Height < 0) {

//printf("Ошибка, попытка выйти за пределы списка.");

return false;

}

a.dop1++; a.n--;

return true;

}

bool PreviousElem(List &a) {

if (a.Mass[a.dop1 - 1].Height < 0) {

//printf("Ошибка, попытка выйти за пределы списка.");

return false;

}

a.dop1--;

a.n++;

return true;

}

void LastElem(List &a) {

while (a.Mass[a.dop1+1].Height > 0) {

a.dop1++;

}

a.n = 1;

}

void LogFile(List a, int argc, char\* argv[]) {

int CurN = a.n, Curdop1 = a.dop1;

FILE\* log = fopen(argv[3], "r+t");

fseek(log, 0, SEEK\_END);

FirstElem(a);

for (int i = a.dop1; i < a.dop1 + a.n; i++)

{

FPrintTInfo(a.Mass[i], log);

fprintf(log, "\n");

}

a.n = CurN; a.dop1 = Curdop1;

fprintf(log, "\n");

fclose(log);

system(argv[3]);

}

void CreateList(List &a, TInfo info, int argc, char\* argv[]) {

CorrectTInfo(info);

if (IsNotEmpty(a)) { delete[] a.Mass; }//если не пуст, то удалить

a.Mass = new TInfo[21];

for (int i = 0; i < 10; i++) { a.Mass[i] = nulll; }

a.Mass[10] = info;

for (int i = 0; i < 10; i++) { a.Mass[i] = nulll; }

a.dop1 = 10;

a.n = 1;

a.dop2 = 10;

return;

}

void PushBack(List &a, TInfo info, int argc, char\* argv[]) {

CorrectTInfo(info);

int CurN = a.n, Curdop1 = a.dop1;

FirstElem(a);

if (a.dop1) { a.Mass[--a.dop1] = info; a.n++; return; } //Надеюсь тута без копии надо...

TInfo\* buff = new TInfo[a.n + 11 + a.dop2];//А тута не куда деваться...

for (int i = 0; i < 10; i++) { buff[i] = nulll; }

buff[10] = info;

for (int i = 0; i < a.n + a.dop2; i++) { buff[i + 11] = a.Mass[i]; }

delete[] a.Mass;

a.Mass = buff;

a.n = ++CurN; a.dop1 = 10;

return;

}

void PopBack(List &a, TInfo info) {

CorrectTInfo(info);

if (a.dop2--) { a.Mass[a.dop1 + a.n++] = info; return; } //Надеюсь тута без копии надо...

TInfo\* buff = new TInfo[a.dop1 + a.n + 11];//А тута не куда деваться...

FirstElem(a);

for (int i = 0; i < a.dop1 + a.n; i++) { buff[i] = a.Mass[i]; }

buff[a.dop1 + a.n] = info;

for (int i = a.dop1 + a.n + 1; i < a.dop1 + a.n + 11; i++) { buff[i] = nulll; }

delete[] a.Mass;

a.Mass = buff;

a.n++;

a.dop2 = 10;

return;

}

void MedBack(List &a, TInfo info) {

CorrectTInfo(info);

if (a.n<2) {

//printf("Ошибка");

PopBack(a, info);

return;

}

if (a.dop2--) {

TInfo buff1, buff2;

buff1 = a.Mass[a.dop1 + 1];

a.Mass[a.dop1 + 1] = info;

for (int i = a.dop1 + 2; i < a.dop1 + a.n; i++) {

buff2 = a.Mass[i];

a.Mass[i] = buff1;

buff1 = buff2;

}

a.Mass[a.dop1 + (a.n++)] = buff1;

return;

}

TInfo\* temp = new TInfo[a.dop1 + a.n + 11];

for (int i = 0; i < a.dop1 + 1; i++) { temp[i] = a.Mass[i]; }

temp[a.dop1 + 1] = info;

for (int i = a.dop1 + 1; i < a.dop1 + a.n; i++) { temp[i + 1] = a.Mass[i]; }

for (int i = a.dop1 + a.n + 1; i < a.dop1 + a.n + 11; i++) { temp[i] = nulll; }

delete [] a.Mass;

a.Mass = temp;

a.n++;

a.dop2 = 10;

}

void DeleteFirst(List &a) {

int CurN = a.n, Curdop1 = a.dop1;

FirstElem(a);

if (!IsNotEmpty(a)) {

//printf("Ошибка");

return;

}

a.Mass[a.dop1++] = nulll;

if ((a.n--) == CurN) {

return;

}

a.n = CurN; a.dop1 = Curdop1;

return;

}

void DeleteLast(List &a) {

if (!IsNotEmpty(a)) {

//printf("Ошибка");

return;

}

a.Mass[a.dop1 + a.n - 1] = nulll;

a.dop2++;

if (a.n == 1) { a.dop1--; return; }

a.n--;

}

void DeleteMedium(List &a) {

if (a.n < 2) {

//printf("Ошибка");

return;

}

for (int i = a.dop1 + 2; i < a.dop1 + a.n; i++)

{

a.Mass[i - 1] = a.Mass[i];

}

a.Mass[a.dop1 + a.n - 1] = nulll;

a.n--;

a.dop2++;

return;

}

void FPrintTInfo(TInfo a, FILE\* buff) {

fprintf(buff, "FN:%s ", a.FirstName);

fprintf(buff, "LN:%s ", a.LastName);

fprintf(buff, "Y:%d ", a.Year);

fprintf(buff, "W:%d ", a.Weight);

fprintf(buff, "H:%d ", a.Height);

}

void FScanTInfo(TInfo &a, FILE\* txt) {

fscanf(txt, "%s", a.FirstName);

fscanf(txt, "%s", a.LastName);

fscanf(txt, "%d", &a.Year);

fscanf(txt, "%d", &a.Weight);

fscanf(txt, "%d", &a.Height);

}

void ScanTInfo(TInfo &a) {

scanf("%s", a.FirstName);

scanf("%s", a.LastName);

scanf("%d", &a.Year);

scanf("%d", &a.Weight);

scanf("%d", &a.Height);

}

TInfo getTInfo(List a) { return a.Mass[a.dop1]; }

**Часть Б2. Создать приложение** (консольное, на C или Delphi), позволяющее выбрать и выполнить следующие действия над списком (только с помощью выше перечисленных базовых операций! Без прямого обращения к элементам массива/файла):

- очистить список;

- добавить элементы из текстового файла (в начало/ в конец/ после текущего);

- добавить один элемент, на основе данных введенных в интерактивном режиме (с клавиатуры или с формы) (в начало/ в конец/ после текущего);

- удалить N элементов списка (из начала/ из конца/ после текущего);

- сделать текущим элементом первый/последний/предыдущий/следующий;

- фильтр: *удалить* из списка все элементы, не удовлетворяющие *условию поиска*, указанному в списке задач по вариантам (стр.3), и *продублировать* удовлетворяющие условию.

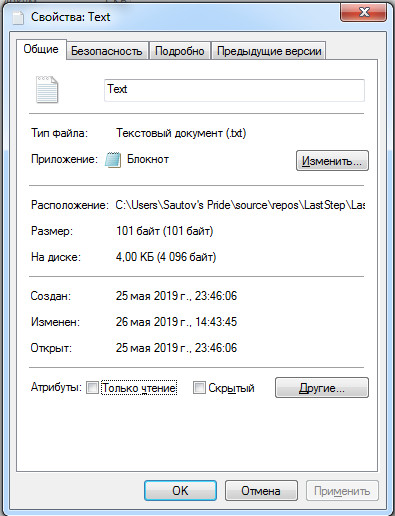
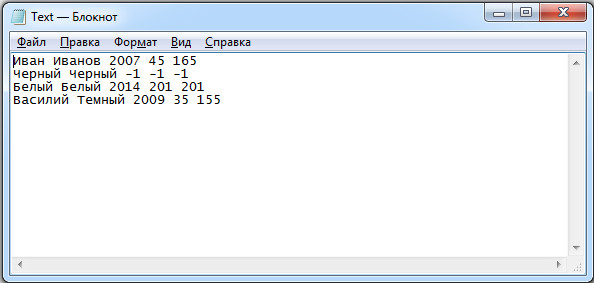
После каждой операции выводить содержимое списка в лог-файл.

При *добавлении* элемента предусмотреть его *коррекцию* в соответствии с условием *коррекции* из списка задач по вариантам (стр.3).

**УСЛОВИЕ:** 19. *Исходные данные*: О детях, поступающих в спортивную секцию, собраны следующие данные: фамилия, имя, год рождения, вес, рост. *Условие поиска*: Дети от 10 до 12 лет, рост которых не менее 155см, а вес не более 45 кг. *Коррекция*: Все дети не младше 5 лет и не старше 14 по году рождения, рост-вес положительные и не более 200.

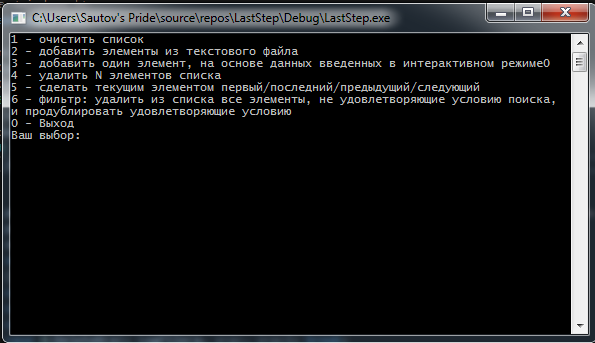
­­­­Формат входного файла (.txt)

Пример:



**Интерфейс**

Для интерактивного взаимодействия с пользователем используется меню:



**Программный код – одинаковый для разных способов моделирования. Разница лишь в имени подключаемого модуля с одноименными типами и операциями со списком:**

#pragma once

#define file

#ifdef file //argv[1] - файл для списка, argv[2] - текстовый файл для считывания, argv[3] - текстовый лог-файл.

#include "FILE.h"

#else

#include "Mass.h"

#endif // file

void DeleteAll(List &a);

void AddTXT(List &a, int argc, char\* argv[]);

void AddConsole(List &a, int argc, char\* argv[]);

void DeleteN(List &a);

void FirstElement(List &a);

bool NextElement(List &a);

bool PreviousElement(List &a);

void LastElement(List &a);

void DeleteThis(List &a);

#include "MainHeader.h"

void DeleteAll(List &a) {

while (IsNotEmpty(a)) {

DeleteFirst(a);

}

}

const TInfo nulll{ "","",-1,-1,-1 };

void AddTXT(List &a, int argc, char\* argv[]) {

FILE\* txt;

TInfo info;

int ch, i=0;

txt = fopen(argv[2], "rt");

if (!IsNotEmpty(a)) {

FScanTInfo(info, txt);

CreateList(a, info, argc, argv);

while (!feof(txt)) {

FScanTInfo(info, txt);

PopBack(a, info);

}

return;

}

printf("1 - В начало\n2 - После текущего\n3 - В конец");

printf("\nВаш выбор: ");

scanf("%d", &ch);

switch (ch)

{

case 1:

FScanTInfo(info, txt);

PushBack(a, info, argc, argv);

FirstElem(a);

while (!feof(txt)) {

FScanTInfo(info, txt);

MedBack(a, info);

NextElem(a);

i++;

}

for (int j = 0; j < i; j++) { PreviousElem(a); }

break;

case 2:

while (!feof(txt)) {

FScanTInfo(info, txt);

MedBack(a, info);

NextElem(a);

i++;

}

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

PreviousElem(a);

}

break;

case 3:

while (!feof(txt)) {

FScanTInfo(info, txt);

PopBack(a, info);

i++;

}break;

default:printf("Неправильная команда!");

break;

}

}

void AddConsole(List &a, int argc, char\* argv[]) {

int ch;

TInfo info;

int i = 1;

if (!IsNotEmpty(a)) {

ScanTInfo(info);

CreateList(a, info, argc, argv);

return;

}

printf("1 - В начало\n2 - После текущего\n3 - В конец");

printf("\nВаш выбор: ");

scanf("%d", &ch);

switch (ch)

{

case 1:

ScanTInfo(info);

PushBack(a, info, argc, argv);

break;

case 2:

ScanTInfo(info);

MedBack(a, info);

break;

case 3:

ScanTInfo(info);

PopBack(a, info);

break;

case 4:

for (int i = 0; i < 11; i++) { PushBack(a, getTInfo(a), argc, argv); }

break;

default:printf("Неправильная команда!");

break;

}

}

void DeleteN(List &a) {

int ch, i=0, n;

printf("Введите кол-во элементов для удаления: ");

scanf("%d", &n);

printf("1 - Из начала\n2 - После текущего\n3 - Из конца");

printf("\nВаш выбор: ");

scanf("%d", &ch);

switch (ch)

{

case 1:while (IsNotEmpty(a) && (i < n)) { DeleteFirst(a); i++; } break;

case 2:while (IsNotEmpty(a) && (i < n)) { DeleteMedium(a); i++; } break;

case 3:while (IsNotEmpty(a) && (i < n)) { DeleteLast(a); i++; } break;

default:printf("Неправильная команда!"); break;

}

}

void FirstElement(List &a) { FirstElem(a); }

bool NextElement(List &a) { return NextElem(a); }

bool PreviousElement(List &a) { return PreviousElem(a); }

void LastElement(List &a) { LastElem(a); }

void DeleteThis(List &a) {

FirstElement(a);

while (!Uslov(getTInfo(a)) && IsNotEmpty(a)) {

DeleteFirst(a);

}

if (Uslov(getTInfo(a)) && IsNotEmpty(a)) {

MedBack(a, getTInfo(a));

NextElement(a);

}

while (NextElement(a)){

if (Uslov(getTInfo(a))) {

MedBack(a, getTInfo(a));

NextElement(a);

}

else if (getTInfo(a).Height > 0){

PreviousElement(a);

DeleteMedium(a);

}

}

}

#include "MainHeader.h"

#include <windows.h>

using namespace std;

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

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

FILE\* log;

log = fopen(argv[3], "wt");

fclose(log);

List a;

int ch = 0;

do

{

printf("1 - очистить список\n2 - добавить элементы из текстового файла\n3 - добавить один элемент, на основе данных введенных в интерактивном режиме0\n4 - удалить N элементов списка\n5 - сделать текущим элементом первый/последний/предыдущий/следующий\n6 - фильтр: удалить из списка все элементы, не удовлетворяющие условию поиска,\nи продублировать удовлетворяющие условию\n0 - Выход\n");

printf("Ваш выбор: ");

scanf("%d", &ch);

switch (ch)

{

case 1:DeleteAll(a); LogFile(a, argc, argv); break;

case 2:AddTXT(a, argc, argv); LogFile(a, argc, argv); break;

case 3:AddConsole(a, argc, argv); LogFile(a, argc, argv); break;

case 4:DeleteN(a); LogFile(a, argc, argv); break;

case 5:

int i;

printf("1 - первый\n2 - следующий\n3 - предыдущий\n4 - последний");

printf("\nВаш выбор: ");

scanf("%d", &i);

switch (i) {

case 1:FirstElement(a); break;

case 2:NextElement(a); break;

case 3:PreviousElement(a); break;

case 4:LastElement(a); break;

}

break;

case 6:DeleteThis(a); LogFile(a, argc, argv); break;

case 0: break;

default:printf("Неправильная команда!"); break;

}

} while (ch != 0);

return 0;

}

­