МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний технічний університет

«Харківський політехнічний інститут»

Факультет Комп’ютерних наук та програмної інженерії

Кафедра Інформатики та інтелектуальної власності

ЗВІТ

До лабораторной роботи №1 з дисципліни

«Об’єкто-орієнтоване програмування»

Студент Бородай Д.А

Викладач Івашко А.В.

Харків 2022

**Лабораторна робота №1**

**Тема: «Найпростіші класи та об’єкти»**

**Мета роботи:** навчитись створювати прості класи та використовувати їх на прикладі простих об’єктів.

**Завдання:** Розробити клас згідно до свого варіанта. Включити до класу методи **set** (…), **get** (…), **show** (…) та ін., використовуючи принцип інкапсуляції. Написати програму, яка створює список, масив чи колекцію, об’єктів на основі відповідного класу. Під час створення використати випадковий підхід для генерації числової та текстової інформації об’єктів списку. Передбачити можливість додання, редагування та видалення об’єктів зі списку за певними критеріями. В програмі реалізувати функціонал за своїм варіантом.

2. **Abiturient**: Прізвище, Ім'я, По-батькові, Адреса, Оцінки. Створити масив об'єктів. Вивести:

· список абітурієнтів, які мають незадовільні оцінки;

· список абітурієнтів, сума балів у яких не менша за задану;

· вибрати N абітурієнтів, що мають найвищу суму балів, та список абітурієнтів, які мають пів прохідний бал.

**1 Виконання роботи**

Програма побудована за допомогою двозв’язних списків на основі класу Abiturient. У інформаційному полі, згідно завданню, існують дві змінні типу string FullName і address, та змінна типу short int grade з інкапсуляцією private за умовчанням. Також, для доступу к змінним класу були об’явлені методи setterAll, getterFullname, getterAddress, getterGrade. Для виведення у консоль було використано метод show. А також, три дружні функції для сортування списку та роботи з фалом, читання та зипис, такі як: Sort, LoadFromFileList, SaveInFileList. Додатково, для роботи зі списками у полі public було створено покажчики на наступний та попередній елементи списку. Код реалізації класу та методів наведено нижче.

class Abiturient {

string FullName;

string address;

short int grade;

public:

Abiturient\* next, \* prev;

void setterAll(const string& FullName, const string& address, const short int& grade);

string getterFullname() const;

string getterAddress() const;

short int getterGrade() const;

void show();

~Abiturient(){};

friend void Sort(Abiturient\*\* head, short int set);

friend void LoadFromFileList(Abiturient\*\* head, Abiturient\*\* tail);

friend void SaveInFileList(Abiturient\* head);

};

Повний код програми наведено у додатку Б.

Метод setterAll дозволяє проініціалізувати змінні класу. Приклад коду наведено нижче та повна реалізація методу наведено нижче.

void Abiturient::setterAll(const string& FullName, const string& address, const short int& grade)

{

this->FullName = FullName;

this->address = address;

this->grade = grade;

}

Повний код програми наведено у додатку Б.

Що до методів getter, то вони отримують значення цих змінних шляхом return. Повна реалізація методів наведено нижче.

string Abiturient::getterFullname() const { return this->FullName; }

string Abiturient::getterAddress() const { return this->address; }

short int Abiturient::getterGrade() const { return this->grade; }

Повний код програми наведено у додатку Б.

Метод show дозволяє вивести у заданому порядку частину таблиці, що продемонстровано на рисунку 1.1.



Рисунок 1.1 - Друк списку

У робочому класі Main згідно варіанту завдання, наведеного вище, було створено методи unsatisfactory, byGrade, hightesGradeandHalf. Окрім цього, наведено методи по роботі з структурою, а також у інформаційному полі розмір структури. Реалізація класу наведено нижче.

class Main

{

private:

int size;

public:

Main() : size(0) {};

int GetSize() { return size; };

void CreateList(short int count, Abiturient\*\* head, Abiturient\* tail);

void PrintList(Abiturient\* head);

void Insert(Abiturient\*\* head, Abiturient\*\* tail, int pos);

void DeleteInList(Abiturient\*\* Dhead, Abiturient\*\* Dtail, int pos);

bool DcheckNULL(Abiturient\* head);

void FreeList(Abiturient\*\* head);

void table\_header();

// main tasks

void unsatisfactory(Abiturient\*\* head);

void byGrade(Abiturient\*\* head, short int setted);

void hightesGradeandHalf(Abiturient\*\* head);

// working method

void main();

~Main() {};

};

Повний текст програми наведено у додатку А. Для повного ознайомлення з методами по роботі з структурою наведено у додатку Б.

Алгоритм методу unsatisfactory полягає у тому, що існує масив об’єктів класу списка, у який записуються данні студентів, які мають незадовільні оцінки, нижче шести десяти балів, та якщо такі є, то ці данні виводяться на екран з підрахунком студентів. У цьому алгоритмі були використані методи класу Abiturient, наведені вище. Приклад роботи методу наведено на рисунках 1.2 та 1.3.

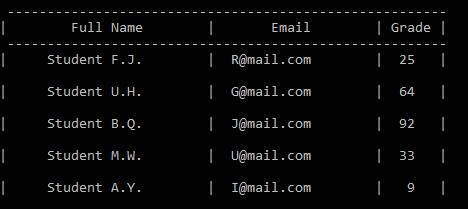


Рисунок 1.2 - Повний список із п'яти студентів

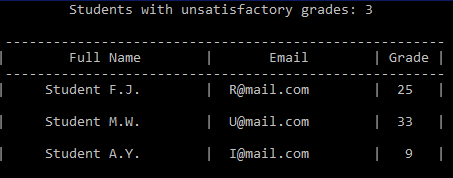


Рисунок 1.3 - Список студенів, оцінка яких є незадовільною

Приклад реалізації цього методу наведено нижче.

Приклад коду метода unsatisfactory:

void Main::unsatisfactory(Abiturient\*\* head)

{

int count = 0;

Abiturient\* current = \*head;

Abiturient\* unsatisfactoryGrade = new Abiturient[MAXSIZE];

if (DcheckNULL((\*head)) == 0) {return;}

else {

while (current) {

if (current->getterGrade() < 60) {

unsatisfactoryGrade[count].setterAll(current->getterFullname(), current->getterAddress(), current->getterGrade());

count++;

}

current = current->next;

}

}

if (count != 0) {

cout << "\t Students with unsatisfactory grades: " << count << endl;

table\_header();

for (size\_t i = 0; i < count; i++) {

unsatisfactoryGrade[i].show(); cout << endl;

}

}

else

cout << "\tStudents haven`t been found!\n";

delete[] unsatisfactoryGrade; unsatisfactoryGrade = NULL;

}

Повний текст програми наведено у додатку Б.

Алгоритм методу byGrade полягає у тому, що існує масив об’єктів класу списка, у який записуються данні студентів, які мають оцінки не нижче заданої оцінки користувачем, та якщо такі є, то ці данні виводяться на екран з підрахунком студентів. У цьому алгоритмі були використані методи класу Abiturient, наведені вище. Приклад роботи методу наведено на рисунках 1.4 та 1.5.

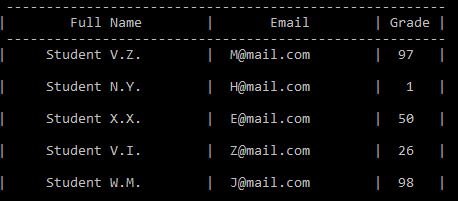


Рисунок 1.4 – Повний список із п’яти студентів

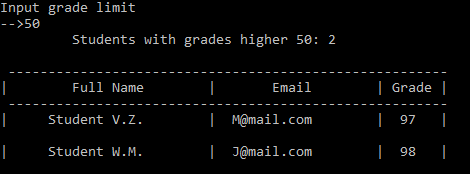


Рисунок 1.5 – Список студентів, сума білів яких більше за 50

Приклад реалізації цього методу наведено нижче.

Приклад коду метода byGrade:

void Main::byGrade(Abiturient\*\* head, short int setted )

{

int count = 0;

Abiturient\* current = \*head;

Abiturient\* byGrade = new Abiturient[MAXSIZE];

if (DcheckNULL((\*head)) == 0) { return; }

else {

while (current) {

if (current->getterGrade() > setted) {

byGrade[count].setterAll(current->getterFullname(), current->getterAddress(), current->getterGrade());

count++;

}

current = current->next;

}

}

if (count != 0) {

cout << "\t Students with grades higher " << setted << ": " << count << endl;

table\_header();

for (size\_t i = 0; i < count; i++) {

byGrade[i].show(); cout << endl;

}

}

else

cout << "\tStudents haven`t been found!\n";

delete[] byGrade; byGrade = NULL;

}

Повний текст програми наведено у додатку Б.

Алгоритм методу hightesGradeandHalf полягає у тому, що існує масив об’єктів класу списка, у який записуються данні студентів, які мають оцінки не нижче 90 і від 34 до 60, та якщо такі є, то ці данні виводяться на екран з підрахунком студентів. Також існує другий масив робота для відбору студентів користувачем та виведення інформації цих студентів на екран. У цьому алгоритмі були використані методи класу Abiturient, наведені вище. Приклад роботи методу наведено на рисунках 1.6 та 1.7.

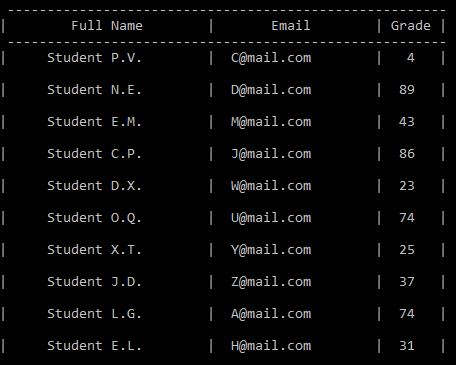


Рисунок 1.6 – Повний список десяти студентів

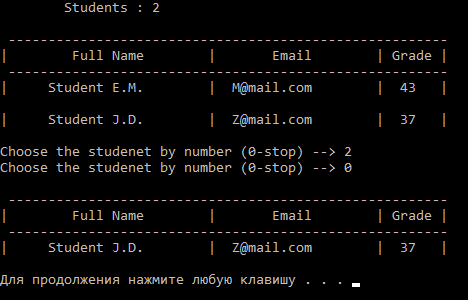


Рисунок 1.7 – Результат виконання методу hightesGradeandHalf

Приклад реалізації цього методу наведено нижче.

Приклад коду метода hightesGradeandHalf:

void Main::hightesGradeandHalf(Abiturient\*\* head)

{

int count = 0;

Abiturient\* current = \*head;

Abiturient\* byGrade = new Abiturient[MAXSIZE];

if (DcheckNULL((\*head)) == 0) { return; }

else {

while (current) {

if (current->getterGrade() > 89 or ( current->getterGrade() > 34 and current->getterGrade() < 60)) {

byGrade[count].setterAll(current->getterFullname(), current->getterAddress(), current->getterGrade());

count++;

}

current = current->next;

}

}

if (count != 0) {

cout << "\tStudents " << ": " << count << endl;

table\_header();

for (size\_t i = 0; i < count; i++) {

byGrade[i].show(); cout << endl;

}

if (count != 1) {

short int choose = -1;

Abiturient\* chosen = new Abiturient[count];

short int flag = 0;

short index = 0;

current = \*head;

while (choose != 0 and flag != count) {

cout << "Choose the studenet by number (0-stop) --> "; cin >> choose;

if (choose > count)

cout << "\tFalse input, try again!\n"; continue;

if (choose != 0) { flag++; };

for (size\_t i = 0; i < choose; i++) {

if (i == choose - 1) {

chosen[index] = byGrade[choose - 1];

index++;

}

current = current->next;

}

}

if (flag != 0) {

table\_header();

for (size\_t i = 0; i < flag; i++) {chosen[i].show(); cout << endl;}

}

else

cout << "\tStudents haven`t been chosen!\n";

delete[] chosen; chosen = NULL;

}

else

cout << "You shouldn`t to choose the one student!\n";

}

else

cout << "\tStudents haven`t been found!\n";

delete[] byGrade; byGrade = NULL;

}

Повний текст програми наведено у додатку Б.

**2 Довідка по роботі з програмою для користувача.**

Головне меню зустрічає користувача п’ятьма пунктами. Перший пункт – це робота вже з існуючим списком, якщо списку не існує, то користувачу програма пропонує його створити. Другий – збереження цього списку на локальному диску, новоствореного чи вже існуючого. Третій пункт – це завантаження до програми вже існуючого списку. Четвертий – це видалення з пам’яті програми списку, з яким користувач працював. Та останній – вихід з програми. Важливо те, що до цього меню можна завжди повернутися, це дає можливість після кожної операції самостійно зберігати список, не втрачаючи данні, або почати роботу вже з новим. Меню наглядно продемонстровано на рисунку 2.1.

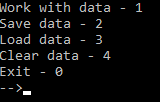


Рисунок 2.1 – Головне меню

У меню по роботі зі списком наведено сім пунктів, перші три з яких відносяться до завдання, що було поставлено раніше. Що продемонстровано на рисунку 2.2.

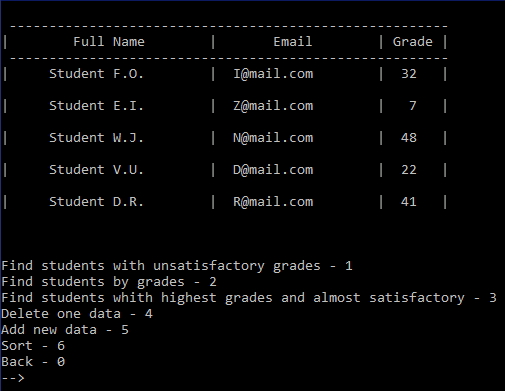


Рисунок 2.2 – Робота зі списком

Пункт четвертий пропонує користувачу видалити одну комірку даних студента у заданому місці, що наведено на рисунку 2.3 та 2.4.



Рисунок 2.3 – Задання номеру комірки для видалення



Рисунок 2.4 – Результат видалення третьої комірки

П’ятий пункт дозволяє навпаки додати після заданої комірки даних, результат роботи наведено на рисунку 2.5 та 2.6.



Рисунок 2.5 – Задання комірки, після якої буде додано нові данні



Рисунок 2.6 – Результат додавання нових даних після другої комірки

У шостому пункті користувач може відсортувати від найбільшого до найменшого оцінки або ФІО, задля зручного особистого пошуку або користування. Роботу пункту продемонстровано на рисунку 2.7.



Рисунок 2.7 – Результат сортування за оцінками

Повертаючись до трьох перших пунктів, у першому – користувач може за допомогою програми віднайти студентів з незадовільними оцінками, див. рис.1.2 та 1.3. У другому пункті – користувач може знайти студентів, які маю не нижчу оцінку, задану ним див. рис.1.4 та 1.5. Третій пункт дозволяє віднайти студентів з оцінкою вище 89 балів та студентів які мають можливість вийти на перездачу та/або повторний екзамен, див. рис.1.6 та 1.7. Останній пункт дозволяє повернутися до головного меню.

**Висновок**

Підчас цієї лабораторної роботи було засвоєно базові навички по робі з класами, їх створенню та роботами над об’єктами класу.

**Додаток А**

Header.h

#pragma once

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <string>

#include <iomanip>

#include <ctime>

#define MAXSIZE 300

using namespace std;

class Abiturient {

string FullName;

string address;

short int grade;

public:

Abiturient\* next, \* prev;

//Abiturient();

// setter

void setterAll(const string& FullName, const string& address, const short int& grade);

//getters

string getterFullname() const; string getterAddress() const; short int getterGrade() const;

void show();

~Abiturient(){};

friend void Sort(Abiturient\*\* head, short int set);

friend void LoadFromFileList(Abiturient\*\* head, Abiturient\*\* tail);

friend void SaveInFileList(Abiturient\* head);

};

class Main

{

private:

int size;

public:

Main() : size(0) {};

int GetSize() { return size; };

void CreateList(short int count, Abiturient\*\* head, Abiturient\* tail);

void PrintList(Abiturient\* head);

void Insert(Abiturient\*\* head, Abiturient\*\* tail, int pos);

void DeleteInList(Abiturient\*\* Dhead, Abiturient\*\* Dtail, int pos);

bool DcheckNULL(Abiturient\* head);

void FreeList(Abiturient\*\* head);

void table\_header();

// main tasks

void unsatisfactory(Abiturient\*\* head);

void byGrade(Abiturient\*\* head, short int setted);

void hightesGradeandHalf(Abiturient\*\* head);

// working method

void main();

~Main() {};

};

**Додаток Б**

Source.cpp

#include "Header.h"

// main class

void Main::main()

{

short int count, button, checker;

Abiturient\* head = NULL, \* tail = NULL;

while (true){

//cout << "Work with data - 1 \nSave data - 2 \nLoad data - 3 \nClear data - 4 \nExit - 0 \n-->"; cin >> button;

cout << "Work with data - 1 \nClear data - 4 \nExit - 0 \n-->"; cin >> button;

if (cin.fail()) {

system("cls");

cin.clear();

cin.ignore(32767, '\n');

cout << " \t\t\tIncorrect input! ERROR! " << endl;

continue;

}

switch (button)

{

case 1:

while (button != 0) {

system("cls");

if (head == NULL) {

cout << "How long you list could be? \n---> "; cin >> count;

if (cin.fail()) {

system("cls");

cin.clear();

cin.ignore(32767, '\n');

cout << " \t\t\tIncorrect input! ERROR! " << endl;

continue;

}

CreateList(count, &head, tail);

system("cls");

}

table\_header();

PrintList(head);

cout << "Find students with unsatisfactory grades - 1\n" <<

"Find students by grades - 2\n" <<

"Find students whith highest grades and almost satisfactory - 3\n" <<

"Delete one data - 4\n" <<

"Add new data - 5\n" <<

"Sort - 6\n" <<

"Back - 0\n" <<

"-->";

cin >> button;

if (cin.fail()) {

system("cls");

cin.clear();

cin.ignore(32767, '\n');

cout << " \t\t\tIncorrect input! ERROR! " << endl;

continue;

system("pause");

}

switch (button)

{

case 1:

system("cls");

unsatisfactory(&head);

system("pause");

break;

case 2:

system("cls");

cout << "Input grade limit \n-->"; cin >> checker;

if (cin.fail()) {

system("cls");

cin.clear();

cin.ignore(32767, '\n');

cout << " \t\t\tIncorrect input! ERROR! " << endl;

system("pause");

continue;

}

byGrade(&head, checker);

system("pause");

break;

case 3:

system("cls");

hightesGradeandHalf(&head);

system("pause");

break;

case 4:

system("cls");

cout << "Input number data \n-->"; cin >> checker;

if (cin.fail()) {

system("cls");

cin.clear();

cin.ignore(32767, '\n');

cout << " \t\t\tIncorrect input! ERROR! " << endl;

system("pause");

continue;

}

DeleteInList(&head, &tail, checker);

break;

case 5:

system("cls");

cout << "Input number data \n-->"; cin >> checker;

if (cin.fail()) {

system("cls");

cin.clear();

cin.ignore(32767, '\n');

cout << " \t\t\tIncorrect input! ERROR! " << endl;

system("pause");

continue;

}

Insert(&head, &tail, checker);

break;

case 6:

system("cls");

cout << "Sort by name - 1 \nSort by grades - 2 \nBack - 0 \n-->"; cin >> button;

if (cin.fail()) {

system("cls");

cin.clear();

cin.ignore(32767, '\n');

cout << " \t\t\tIncorrect input! ERROR! " << endl;

system("pause");

continue;

}

switch (button)

{

case 1:

Sort(&head, button);

table\_header();

PrintList(head);

system("pause");

break;

case 2:

Sort(&head, button);

table\_header();

PrintList(head);

system("pause");

break;

case 0:

system("cls");

break;

default:

break;

}

break;

case 0:

system("cls");

break;

default:

break;

}

}

break;

/\*case 2:

system("cls");

SaveInFileList(head);

break;

case 3:

system("cls");

LoadFromFileList(&head, &tail);

break;\*/

case 4:

system("cls");

if (DcheckNULL(head) != 0)

FreeList(&head);

break;

case 0:

system("cls");

FreeList(&head);

exit(0);

break;

default:

break;

}

}

}

//set methods

void Abiturient::setterAll(const string& FullName, const string& address, const short int& grade)

{

this->FullName = FullName;

this->address = address;

this->grade = grade;

}

// get methods

string Abiturient::getterFullname() const { return this->FullName; }

string Abiturient::getterAddress() const { return this->address; }

short int Abiturient::getterGrade() const { return this->grade; }

// show method

void Abiturient::show()

{

cout << "|" << setw(17) << this->FullName << setw(9) << "|" << setw(12) << this->address << setw(9) << "|" << setw(4) << this->grade << setw(4) << "|" << endl;

}

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

///////////////////// Main tasks ///////////////////////////////////

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

void Main::unsatisfactory(Abiturient\*\* head)

{

int count = 0;

Abiturient\* current = \*head;

Abiturient\* unsatisfactoryGrade = new Abiturient[MAXSIZE];

if (DcheckNULL((\*head)) == 0) {return;}

else {

while (current) {

if (current->getterGrade() < 60) {

unsatisfactoryGrade[count].setterAll(current->getterFullname(), current->getterAddress(), current->getterGrade());

count++;

}

current = current->next;

}

}

if (count != 0) {

cout << "\t Students with unsatisfactory grades: " << count << endl;

table\_header();

for (size\_t i = 0; i < count; i++) {

unsatisfactoryGrade[i].show(); cout << endl;

}

}

else

cout << "\tStudents haven`t been found!\n";

delete[] unsatisfactoryGrade; unsatisfactoryGrade = NULL;

}

void Main::byGrade(Abiturient\*\* head, short int setted )

{

int count = 0;

Abiturient\* current = \*head;

Abiturient\* byGrade = new Abiturient[MAXSIZE];

if (DcheckNULL((\*head)) == 0) { return; }

else {

while (current) {

if (current->getterGrade() > setted) {

byGrade[count].setterAll(current->getterFullname(), current->getterAddress(), current->getterGrade());

count++;

}

current = current->next;

}

}

if (count != 0) {

cout << "\t Students with grades higher " << setted << ": " << count << endl;

table\_header();

for (size\_t i = 0; i < count; i++) {

byGrade[i].show(); cout << endl;

}

}

else

cout << "\tStudents haven`t been found!\n";

delete[] byGrade; byGrade = NULL;

}

void Main::hightesGradeandHalf(Abiturient\*\* head)

{

int count = 0;

Abiturient\* current = \*head;

Abiturient\* byGrade = new Abiturient[MAXSIZE];

if (DcheckNULL((\*head)) == 0) { return; }

else {

while (current) {

if (current->getterGrade() > 89 or ( current->getterGrade() > 34 and current->getterGrade() < 60)) {

byGrade[count].setterAll(current->getterFullname(), current->getterAddress(), current->getterGrade());

count++;

}

current = current->next;

}

}

if (count != 0) {

cout << "\tStudents " << ": " << count << endl;

table\_header();

for (size\_t i = 0; i < count; i++) {

byGrade[i].show(); cout << endl;

}

if (count != 1) {

short int choose = -1;

Abiturient\* chosen = new Abiturient[count];

short int flag = 0;

short index = 0;

current = \*head;

while (choose != 0 and flag != count) {

cout << "Choose the studenet by number (0-stop) --> "; cin >> choose;

if (choose > count) {

cout << "\tFalse input, try again!\n";

continue;

}

if (choose != 0) { flag++; };

for (size\_t i = 0; i < choose; i++) {

if (i == choose - 1) {

chosen[index] = byGrade[choose - 1];

index++;

}

current = current->next;

}

}

if (flag != 0) {

table\_header();

for (size\_t i = 0; i < flag; i++) {

chosen[i].show(); cout << endl;

}

}

else

cout << "\tStudents haven`t been chosen!\n";

delete[] chosen; chosen = NULL;

}

else {

cout << "You shouldn`t to choose the one student!\n";

}

}

else

cout << "\tStudents haven`t been found!\n";

delete[] byGrade; byGrade = NULL;

}

void Main::table\_header()

{

cout << "\n -------------------------------------------------------\n";

cout << "|" << setw(17) << "Full Name" << setw(9) << "|" << setw(12) << "Email" << setw(9) << "|" << setw(6) << "Grade" << setw(2) << "|";

cout << "\n -------------------------------------------------------\n";

}

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

///////////////////// Doubly Node ///////////////////////////////////

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

void Main::CreateList(short int count, Abiturient\*\* head, Abiturient\* tail)

{

string FullName = "Student ", init, address; short int grade;

if (count > 0) {

(\*head) = new Abiturient;

//cout << "Input you value \n--->"; //cin >> FullName >> address >> grade;

init = rand() % 26 + 65; init += '.'; init += rand() % 26 + 65; init += '.';

FullName += init;

address += rand() %26 + 65; address += "@mail.com";

grade = rand() % 101;

(\*head)->setterAll(FullName, address, grade);

(\*head)->prev = tail;

(\*head)->next = NULL;

CreateList(count - 1, &((\*head)->next), (\*head));

}

else (\*head) = NULL;

size = count;

}

void Main::PrintList(Abiturient\* head) {

if (head != NULL) {

head->show();

cout << endl;

PrintList(head->next);

}

else cout << endl << endl;

}

void Main::Insert(Abiturient\*\* head, Abiturient\*\* tail, int pos) {

Abiturient\* temp = new Abiturient;

if (pos == 0) {

temp->next = \*head;

(\*head)->prev = temp;

\*head = temp;

temp->prev = NULL;

}

else {

Abiturient\* current = \*head;

for (size\_t i = 0; i < pos - 1; i++)

current = current->next;

if (current->next == NULL) {

current->next = temp;

temp->prev = \*tail;

temp->next = NULL;

\*tail = temp;

}

else {

temp->next = current->next;

current->next = temp;

temp->next->prev = temp;

temp->prev = current;

}

}

string FullName = "Student ", init, address; short int grade;

//cout << "Insert: ";

//cin >> FullName >> address >> grade;

init = rand() % 26 + 65; init += '.'; init += rand() % 26 + 65; init += '.';

FullName += init;

address += rand() % 26 + 65; address += "@mail.com";

grade = rand() % 101;

temp->setterAll(FullName, address, grade);

}

void Main::DeleteInList(Abiturient\*\* head, Abiturient\*\* tail, int pos) {

Abiturient\* current = \*head;

if (pos == 0) {

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

(\*head)->prev = NULL;

delete current;

}

else {

for (size\_t i = 0; i < pos - 1; i++)

current = current->next;

Abiturient\* temp = current->next;

if (temp->next == NULL) {

current->next = NULL;

\*tail = current;

}

else {

current->next = temp->next;

temp->next->prev = current;

}

delete temp;

}

}

bool Main::DcheckNULL(Abiturient\* head) { // cheking on empty

if (head == NULL) {

cout << "List is empty." << endl;

return 0;

}

return 1;

}

void Main::FreeList(Abiturient\*\* head) { // clean list

Abiturient\* current = \*head;

while (current) {

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

free(current);

current = \*head;

}

}

void Sort(Abiturient\*\* head, short int set) {

Abiturient\* left = \*head;

Abiturient\* right = (\*head)->next;

Abiturient\* temp = new Abiturient;

// preset switch

switch (set) {

case 1: // sort by name

while (left->next) {

while (right) {

if ((left->FullName.compare(right->FullName) > 0)) {

temp->FullName = left->FullName;

left->FullName = right->FullName;

right->FullName = temp->FullName;

}

right = right->next;

}

left = left->next;

right = left->next;

}

break;

case 2: // sort by grades

while (left->next) {

while (right) {

if (right->grade > left->grade) {

temp->grade = right->grade;

right->grade = right->grade;

left->grade = temp->grade;

}

right = right->next;

}

left = left->next;

right = left->next;

}

break;

default:

//system("cls");

break;

}

}

void SaveInFileList(Abiturient\* head) { // writing in data.txt

Abiturient\* current = head;

FILE\* data;

fopen\_s(&data, "data.txt", "w");

if (!data) {

cout << "ERROR. Missing data.txt" << endl;

}

while (current) {

fprintf(data, "\t%s\t%s\t%i\n", current->FullName, current->address, current->grade);

current = current->next;

}

fclose(data);

}

void LoadFromFileList(Abiturient\*\* head, Abiturient\*\* tail) { // reading from data.txt

Abiturient\* current, \* prev;

prev = current = (\*head) = new Abiturient;

(\*head)->prev = NULL;

FILE\* data;

fopen\_s(&data, "data.txt", "r");

if (!data) {

cout << "ERROR. Missing data.txt";

delete current;

(\*head) = NULL;

return;

}

if (fgetc(data) == EOF) {

cout << "File is empty.";

delete current;

(\*head) = nullptr;

return;

}

fscanf(data, "\t%s\t%s\t%i\n", current->FullName, current->address, current->grade);

while (!feof(data)) {

current = new Abiturient;

fscanf(data, "\t%s\t%s\t%i\n", current->FullName, current->address, current->grade);

prev->next = current;

current->prev = prev;

prev = current;

}

current->next = NULL;

\*tail = current;

cout << "File has been successfully saved." << endl;

fclose(data);

}