**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра «Информационные Системы и Технологий (ИСиТ)»**

отчет

**по лабораторной работе №2**

**по дисциплине «Программирование»**

Тема: «Динамические массивы и двусвязные списки»

|  |  |  |
| --- | --- | --- |
| Студент гр. 0324 |  | Зимацкий С. Н. |
| Преподаватель |  | Глущенко А.Г. |

Санкт-Петербург

2021

**Цель работы.**

Изучение свойств и организация динамических массивов и двусвязных списков; получение практических навыков в работе с динамическими массивами и двусвязными списками; проведение сравнительной характеристики скорости вставки, получения и удаления элементов из них.

Необходимо реализовать программу, которая выполняет следующие действия.

1.   Формирование целочисленного одномерного массива размерности *N*, где:

a) пользователь вводит количество элементов в массиве, который будет автоматически заполняться случайными числами (0 до 99);

б) пользователь вводит в консоль элементы массива, *N* определяется автоматически по количеству введенных элементов;

в) \* массив считывается с файла, *N* определяется как количество элементов массива в файле.

2.   Определение скорости создания динамического массива п. 1.

3.   Вставка, удаление и получение элемента массива. Удаление и получение элемента необходимо реализовать по индексу и по значению.

4.   Определение скорости вставки, удаления и получения элемента массива п. 3.

5.   Формирование двусвязного списка размерности *N*, где:

a) пользователь вводит количество элементов в списке, который будет автоматически заполняться случайными числами (0 до 99);

б) пользователь вводит в консоль элементы списка, *N* определяется автоматически по количеству введенных элементов;

в) \* список считывается с файла, *N* определяется как количество элементов списка в файле.

6.   Определение скорости создания двусвязного списка п. 5.

7.   Вставка, удаление и получение элемента двусвязного списка. Удаление и получение элемента необходимо реализовать по индексу и по значению.

8.   Определение скорости вставки, удаление и получения элемента двусвязного списка п. 7.

Должна быть возможность запуска каждого пункта многократно, если есть возможность (если в списке/массиве нет элементов, то нельзя ничего удалить и об этом нужно сообщить пользователю). Необходимо сравнить результаты. Для этого пункты 1–4 и 5–8 должны принимать одинаковые значения.

**Основные теоретические положения.**

## Динамические объекты

В C++ можно использовать различные типы объектов, которые различаются по использованию памяти. Так, глобальные объекты создаются при запуске программы и освобождаются при ее завершении. Локальные автоматические объекты создаются в блоке кода и удаляются, когда этот блок кода завершает работу. Локальные статические объекты создаются перед их первым использованием и освобождаются при завершении программы.

Глобальные, а также статические локальные объекты помещаются в статической памяти, а локальные автоматические объекты размещаются в стеке. Объекты в статической памяти и стеке создаются и удаляются компилятором. Статическая память очищается при завершении программы, а объекты из стека существуют, пока выполняется блок, в котором они определены.

В дополнение к этим типам в C++ можно создавать динамические объекты. Продолжительность их жизни не зависит от того, где они созданы. Динамические объекты существуют, пока не будут удалены явным образом. Динамические объекты размещаются в динамической памяти.

Для управления динамическими объектами в С++ применяются операторы **new** и **delete**.

Оператор **new** выделяет место в динамической памяти для объекта и возвращает указатель на этот объект.

Оператор **delete** получает указатель на динамический объект и удаляет его из памяти.

## Выделение памяти

Создание динамического объекта:

int \*ptr = new int;

Оператор **new** создает новый объект типа int в динамической памяти и возвращает указатель на него. Значение такого объекта неопределенно.

## Освобождение памяти

Динамические объекты будут существовать пока не будут явным образом удалены. И после завершения использования динамических объектов следует освободить их память с помощью оператора **delete**:

int \*p1 = new int(12);

std::cout << "p1: " << \*p1 << "\n"; // 0

delete p1;

## Одномерные динамические массивы

Для того чтобы создать в динамической области некоторый объект, необходима одна обычная переменная-указатель (не динамическая переменная). Сколько таких объектов понадобится для одновременной обработки, столько необходимо иметь обычных переменных-указателей. Таким образом, проблема задач неопределенной размерности созданием одиночных динамических объектов решена быть не может.

Решить эту проблему поможет возможность создавать в динамической области памяти массивы объектов с таким количеством элементов, которое необходимо в данный момент работы программы, т. е. создание динамических массивов. Действительно, для представления массива требуется всего одна переменная-указатель, а в самом массиве, на который ссылается этот указатель, может быть столько элементов, сколько требуется в данный момент времени.

Для создания одномерного динамического массива используется следующий синтаксис инструкции new (стиль С++):

int \*Arr = new int [100];

Причем в этом случае оператор new также возвращает указатель на объект типа int - первый элемент в созданном массиве.

Освободить динамическую область от этого массива можно с помощью инструкции delete:

delete [] Arr;

После этого занятый участок памяти будет возвращен в список свободной памяти и может быть повторно использован для размещения других динамических объектов.

## Двусвязные линейные списки

Одним из недостатков односвязных списков является то, что узел (элемент списка) имеет указатель только на следующий элемент. Вернуться из текущего элемента к предыдущему явным способом невозможно.

Каждый узел двусвязного (двунаправленного) линейного списка содержит два поля указателей – на следующий и на предыдущий узлы. Указатель на предыдущий узел корня списка содержит нулевое значение. Указатель последнего узла также содержит нулевое значение.

Поскольку каждый элемент списка должен иметь три части, логичнее всего представить его в виде следующей структуры:

struct list

{

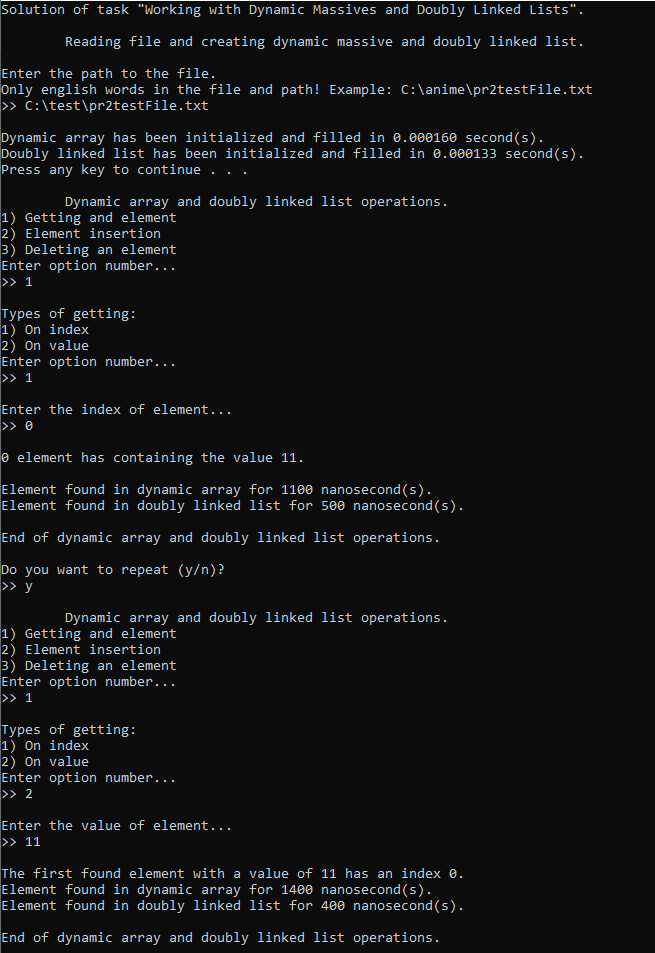
int data;

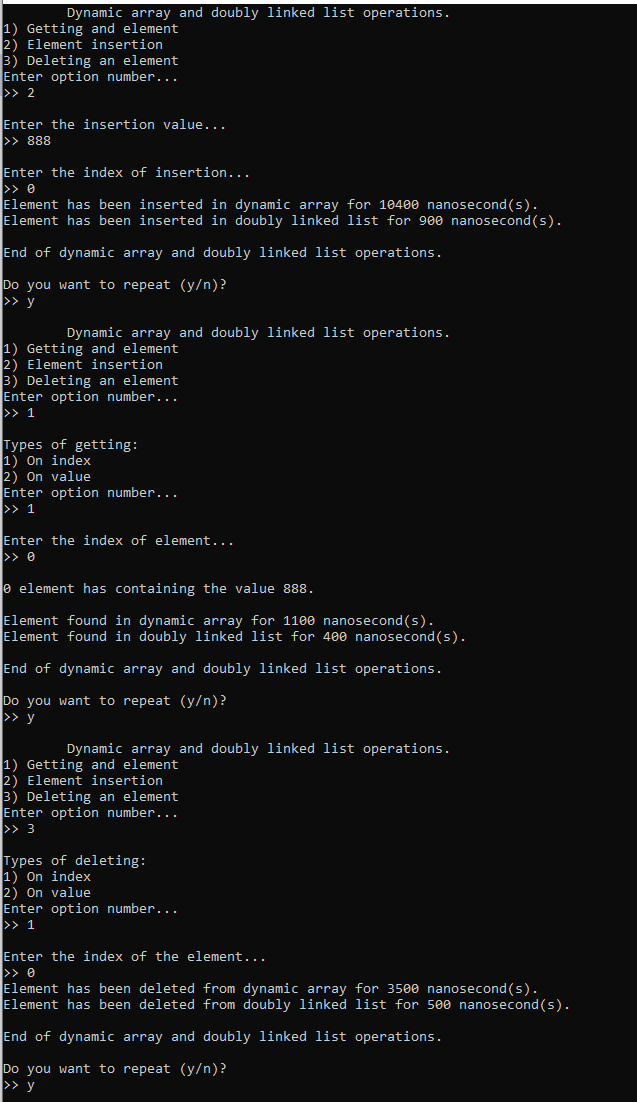
list \*head;

list \*tail;

};

**Экспериментальные результаты.**





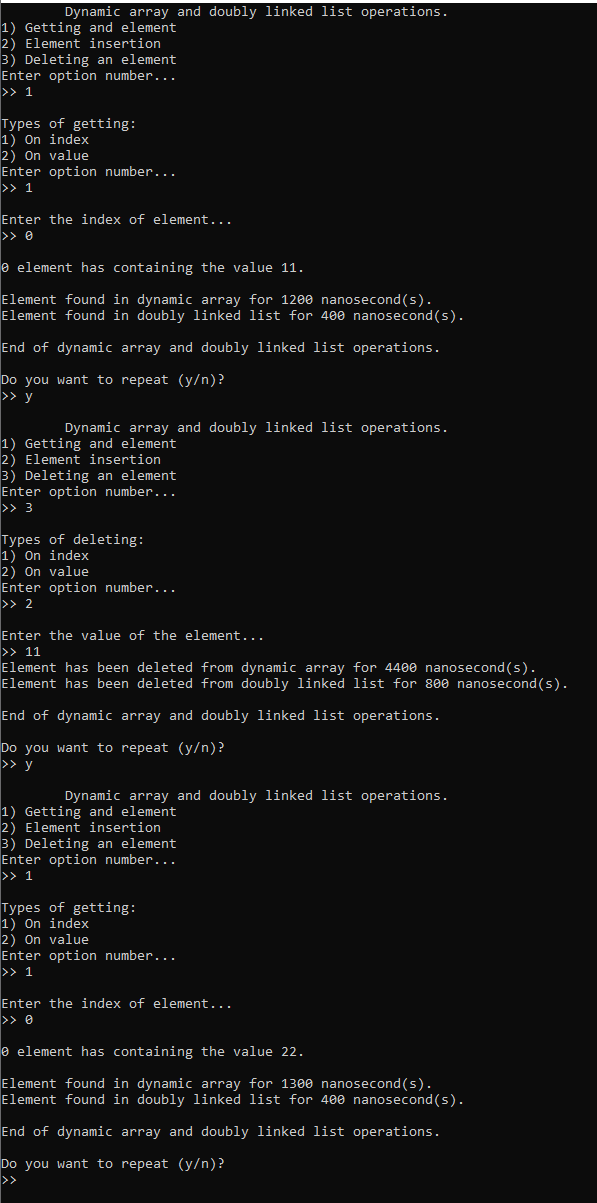


Рисунок 5 – Результат работы программы (полный код программы представлен в приложении А)

Приложение а  
полный код программы

#include <iostream>

#include <iomanip>

#include <math.h>

#include <string>

#include <fstream>

#include <regex>

using namespace std;

const int numOfStudentFields = 7;

struct timeHM {

string hour;

string minute;

string second;

};

struct date {

string dayOfWeek;

string month;

int dayOfMonth;

timeHM time;

int year;

};

struct student {

string fullName;

int group;

int id;

char sex;

string educationForm;

short grades[8];

date changedOn;

};

void practicalWork1();

void practicalWork2();

int countOfElements(ifstream \* file); // функции определены в другом файле

int countOfRecords(ifstream \*);

student readStudentFromFile(ifstream \*);

student readStudentFromKeyboard();

student \* addStruct(student \*, int \*);

void addStudentsToFile(ofstream\*, student\*, int);

void printStudent(student);

int searchStudent(student \*, string, int);

void printTop(student \*, int);

int countOfMale(student \*, int);

int countOfFemale(student \*, int);

int countOfScholarships(student \*, int);

int countOfGoodStudents(student \*, int);

int countOfExcellentStudents(student \*, int);

void printStudentsFromDate(student \*, date, int);

void printStudentsFromDateBeforeNoon(student \*, date, int);

void printStudentsFromDateAfterNoon(student \*, date, int);

date readFullTimeFromKeyboard();

date readDateFromKeyboard();

bool isFullName(string);

bool isLetter(char);

bool isDay(string);

bool isWeek(string);

void printDate(date);

void printGrades(short \*);

bool choiseNextAction();

int main() {

while (true) {

try {

system("CLS");

cout << "Practical works: \n" <<

"\t1) Structures \n" <<

"\t2) Dynamic Massives and Doubly Linked Lists \n" <<

"Enter the number of practical work or enter 0 for close the program... \n>> ";

int input;

cin >> input;

switch (input) {

case 1:

practicalWork1();

break;

case 2:

practicalWork2();

break;

default:

goto Exit;

}

}

catch (int exception) {

if (exception == -1)

goto Exit;

}

catch (...) {}

}

Exit:

cout << "\nClosing the program... \n";

system("pause");

return 0;

}

void practicalWork1() {

system("CLS");

cout << "Solution of task \"Working with structures\". \n\n"

<< "Task 0. Reading text file. \n";

ifstream ifile;

ofstream ofile;

string path;

int numberOfRecords;

student \* students = 0;

do {

cout << "Enter the path to the file. \n"

<< "Only english words in the file and path! Example: C:\\anime\\students.txt \n>> ";

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

getline(cin, path);

ifile.open(path);

if (!ifile.is\_open()) {

cout << "Error opening file! Please restart the program! \n";

throw -1;

}

ifstream \*pifile = &ifile;

numberOfRecords = countOfRecords(pifile);

if (!(numberOfRecords % numOfStudentFields == 0)) {

cout << "Error! Invalid number of lines in the file. Must be a multiple " << numOfStudentFields << ".\n";

continue;

}

numberOfRecords /= numOfStudentFields;

ifile.close();

ifile.open(path);

students = new student[numberOfRecords];

try {

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

students[i] = readStudentFromFile(pifile);

}

}

catch (const char\* msg) {

cout << msg;

continue;

}

} while (!ifile.is\_open());

ifile.close();

bool isInMenu = true;

do {

system("CLS");

cout << "Operations. \n"

<< "1) Add student \n"

<< "2) Change exist student \n"

<< "3) Display all students \n"

<< "4) Display all students from the group \n"

<< "5) Display rating \n"

<< "6) Display number of male and female students \n"

<< "7) Display the number of students who will receive the scholarship \n"

<< "8) Display the number of students who will not recieve the scholarship; have good and excellent grades; have only excellent grades \n"

<< "9) Display all students with the entered number (id) in groups \n"

<< "10) Display all students from the entered date \n\n"

<< "0) Exit to practical works menu \n>> ";

int input;

cin >> input;

cout << endl;

switch (input) {

case 1:

do {

system("CLS");

cout << "Task 1. Adding a student. \n";

student newStudent = readStudentFromKeyboard();

cout << "\nYou entered: \n";

printStudent(newStudent);

cout << "\nDo you want to add this student? Y/N \n>> ";

char input;

cin >> input;

if (input == 'Y' || input == 'y') {

ofstream \* pofile = &ofile;

students = addStruct(students, &numberOfRecords);

students[numberOfRecords - 1] = newStudent;

ofile.open(path);

addStudentsToFile(pofile, students, numberOfRecords);

ofile.close();

}

else if (input == 'N' || input == 'n') {

continue;

}

else

cout << "Error! Invalid input. \n";

} while (choiseNextAction());

continue;

case 2:

{

system("CLS");

cout << "Task 2. Change entry. \n";

bool isRecordsChanged = false;

try {

cout << "Enter fullname of student, which entry you want to change. \n"

<< "Format example: \"Ivanov Ivan Ivanovich\", only letters and only 3 words. \n>> ";

string input;

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

getline(cin, input);

if (isFullName(input)) {

int indexOfStudent = searchStudent(students, input, numberOfRecords);

if (indexOfStudent != -1) {

do {

cout << "What you need to change? \n"

<< "1) Fullname\n"

<< "2) Group\n"

<< "3) Id in the group\n"

<< "4) Sex\n"

<< "5) Form of education\n"

<< "6) Grades\n"

<< "7) Date of changed\n>> ";

int option;

cin >> option;

regex mask;

switch (option) {

case 1:

cout << "Enter fullName. Format example: \"Ivanov Ivan Ivanovich\", only letters and only 3 words. \n>> ";

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

getline(cin, input);

if (isFullName(input)) { // first option of check

students[indexOfStudent].fullName = input;

isRecordsChanged = true;

cout << "Entry updated! \n";

}

else

cout << "Error! Invalid students FullName. Format example: \"Ivanov Ivan Ivanovich\", only letters and only 3 words. \n";

break;

case 2:

cout << "Enter number of group. Example: \"2284\", only numbers, without spaces and any other characters. \n>> ";

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

getline(cin, input);

mask = "^[0-9]\*$"; // second option of check

if (regex\_search(input, mask)) {

students[indexOfStudent].group = stoi(input);

isRecordsChanged = true;

cout << "Entry updated! \n";

}

else

cout << "Error! Invalid number of group format. Example: \"2284\", only numbers, without spaces and any other characters. \n";

break;

case 3:

cout << "Enter student ID. Example: \"12\", only numbers, without spaces and any other characters. \n>> ";

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

getline(cin, input);

if (regex\_search(input, mask)) {

students[indexOfStudent].id = stoi(input);

isRecordsChanged = true;

cout << "Entry updated! \n";

}

else

cout << "Error! Invalid student id format. Example: \"12\", only numbers, without spaces and any other characters. \n";

break;

case 4:

cout << "Enter student sex. Example: only \"M\" or \"F\", without spaces and any other characters. \n>> ";

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

getline(cin, input);

mask = "^[MF|mf]{1}$";

if (regex\_search(input, mask)) {

students[indexOfStudent].sex = input[0];

isRecordsChanged = true;

cout << "Entry updated! \n";

}

else

cout << "Error! Invalid student sex format. Example: only \"M\" or \"F\", without spaces and any other characters. \n";

break;

case 5:

cout << "Enter student form of education. Example: \"distance\", without numbers, spaces and another characters. \n>> ";

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

getline(cin, input);

mask = "^[A-Za-z]\*$";

if (regex\_search(input, mask)) {

students[indexOfStudent].educationForm = input;

isRecordsChanged = true;

cout << "Entry updated! \n";

}

else

cout << "Error! Invalid form of education format. Example: \"distance\", without numbers, spaces and another characters. \n";

break;

case 6:

cout << "Enter student grades. Example: \"3 4 2 5 5 3 4 0 \", grade can be only 0-5, where \"0\" is empty grade. \n>> ";

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

getline(cin, input);

mask = "^[0-5]{1} {1}[0-5]{1} {1}[0-5]{1} {1}[0-5]{1} {1}[0-5]{1} {1}[0-5]{1} {1}[0-5]{1} {1}[0-5]{1}";

if (regex\_search(input, mask)) {

int j = 0;

for (int i = 0; i <= 14; i += 2) {

students[indexOfStudent].grades[j] = (int)input[i] - 48;

j++;

}

isRecordsChanged = true;

cout << "Entry updated! \n";

}

else

cout << "Error! Invalid grades format. Example: \"3 4 2 5 5 3 4 0 \", grade can be only 0-5, where \"0\" is empty grade. \n";

break;

case 7:

students[indexOfStudent].changedOn = readFullTimeFromKeyboard();

isRecordsChanged = true;

cout << "Entry updated! \n";

break;

default:

cout << "Error! Invalid input. \n";

break;

}

} while (choiseNextAction());

if (isRecordsChanged)

students[indexOfStudent].changedOn = readFullTimeFromKeyboard();

cout << "Entry updated! \n";

}

else

throw "Student was not found! \n";

}

else

throw "Error! Invalid students FullName. Format example: \"Ivanov Ivan Ivanovich\", only letters and only 3 words. \n";

}

catch (const char\* msg) {

cout << msg;

continue;

}

if (isRecordsChanged) {

ofstream \* pofile = &ofile;

ofile.open(path);

addStudentsToFile(pofile, students, numberOfRecords);

ofile.close();

}

}

break;

case 3:

system("CLS");

cout << "Task 3. Displaying all students. \n";

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

printStudent(students[i]);

cout << endl;

system("pause");

break;

case 4:

do {

system("CLS");

cout << "Task 4. Displaying all students from the group. \n";

cout << "Enter the group. Example: \"2284\", only numbers, without spaces and any other characters. \n>> ";

int group;

cin >> group;

bool isFound = false;

cout << "\nAll students from the group " << group << ": \n";

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

if (students[i].group == group) {

printStudent(students[i]);

isFound = true;

}

if (isFound)

cout << endl;

else

cout << "Students was not found! \n";

} while (choiseNextAction());

break;

case 5:

system("CLS");

cout << "Task 5. Displaying top of the students by by the highest average grade. \n";

printTop(students, numberOfRecords);

system("pause");

break;

case 6:

system("CLS");

cout << "Task 6. Displaying numbers of male and female students. \n";

cout << "Male students: " << countOfMale(students, numberOfRecords) << endl;

cout << "Female students: " << countOfFemale(students, numberOfRecords) << endl;

system("pause");

break;

case 7:

system("CLS");

cout << "Task 7. Displaying the number of students who will receive the scholarship. \n";

cout << "Will recieve the scholarship " << countOfScholarships(students, numberOfRecords) << " student(s). \n";

system("pause");

break;

case 8:

system("CLS");

cout << "Task 8. Displaying the number of students who will not recieve the scholarship; have good and excellent grades; have only excellent grades \n";

cout << "Will not recieve the scholarship " << numberOfRecords - countOfScholarships(students, numberOfRecords) << " student(s). \n";

cout << "Have good and excellent grades " << countOfGoodStudents(students, numberOfRecords) << " student(s). \n";

cout << "Have only excellent grades " << countOfExcellentStudents(students, numberOfRecords) << " student(s). \n";

system("pause");

break;

case 9:

do {

system("CLS");

cout << "Task 9. Displaying all students from the entered number (id) in groups. \n";

cout << "Enter student ID. Example: \"12\", only numbers, without spaces and any other characters. \n>> ";

int id;

cin >> id;

bool isFound = false;

cout << "\nAll students from the entered ID " << id << ": \n";

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

if (students[i].id == id) {

printStudent(students[i]);

isFound = true;

}

if (isFound)

cout << endl;

else

cout << "Students was not found! \n";

} while (choiseNextAction());

break;

case 10:

do {

system("CLS");

cout << "Task 10. Displaying all students from the entered date.";

cout << "Enter the date. Example: \"Mon Feb 22 12:00:01 2021\". \n>> ";

date inputDate = readDateFromKeyboard();

cout << "Options: \n"

<< "1) All day \n"

<< "2) Only before noon \n"

<< "3) Only after noon \n>> ";

int input2;

cin >> input2;

cout << endl;

switch (input2) {

case 1:

printStudentsFromDate(students, inputDate, numberOfRecords);

break;

case 2:

printStudentsFromDateBeforeNoon(students, inputDate, numberOfRecords);

break;

case 3:

printStudentsFromDateAfterNoon(students, inputDate, numberOfRecords);

break;

default:

cout << "Error! Invalid input. \n";

break;

}

} while (choiseNextAction());

break;

case 0:

isInMenu = false;

break;

default:

cout << "Error! Invalid input. \n";

break;

}

} while (isInMenu);

}

int countOfRecords(ifstream \* file) {

string t;

int count = 0;

while (getline(\*file, t)) {

count++;

}

return count;

}

student readStudentFromFile(ifstream \* file) {

struct student newStudent;

string input;

getline(\*file, input);

if (isFullName(input)) // first option of check

newStudent.fullName = input;

else

throw "Error! Invalid students FullName. Format example: \"Ivanov Ivan Ivanovich\", only letters and only 3 words. \n";

getline(\*file, input);

regex mask("^[0-9]\*$"); // second option of check

if (regex\_search(input, mask))

newStudent.group = stoi(input);

else

throw "Error! Invalid number of group format. Example: \"2284\", only numbers, without spaces and any other characters. \n";

getline(\*file, input);

if (regex\_search(input, mask))

newStudent.id = stoi(input);

else

throw "Error! Invalid student id format. Example: \"12\", only numbers, without spaces and any other characters. \n";

getline(\*file, input);

mask = "^[MF|mf]{1}$";

if (regex\_search(input, mask))

newStudent.sex = input[0];

else

throw "Error! Invalid student sex format. Example: only \"M\" or \"F\", without spaces and any other characters. \n";

getline(\*file, input);

mask = "^[A-Za-z]\*$";

if (regex\_search(input, mask))

newStudent.educationForm = input;

else

throw "Error! Invalid form of education format. Example: \"distance\", without numbers, spaces and another characters. \n";

getline(\*file, input);

mask = "^[0-5]{1} {1}[0-5]{1} {1}[0-5]{1} {1}[0-5]{1} {1}[0-5]{1} {1}[0-5]{1} {1}[0-5]{1} {1}[0-5]{1}";

if (regex\_search(input, mask)) {

int j = 0;

for (int i = 0; i <= 14; i += 2) {

newStudent.grades[j] = (int)input[i] - 48;

j++;

}

}

else

throw "Error! Invalid grades format. Example: \"3 4 2 5 5 3 4 0 \", grade can be only 0-5, where \"0\" is empty grade. \n";

getline(\*file, input);

mask = "^[A-Za-z]{3} {1}[A-Za-z]{3} {1}[0-9]{2} {1}[0-9]{2}:{1}[0-9]{2}:{1}[0-9]{2} {1}[0-9]{4}$";

if (!regex\_search(input, mask))

throw "Error! Invalid date format. Example: \"Mon Feb 22 12:00:01 2021\". \n";

string sub = input.substr(0, 3); // Here and below individual cases of incorrect date are checked

if (isDay(sub))

newStudent.changedOn.dayOfWeek = sub;

else

throw "Error! Invalid day format. Example: \"Mon\" \"Tue\", \"Wed\" etc. \n";

sub = input.substr(4, 3);

if (isWeek(sub))

newStudent.changedOn.month = sub;

else

throw "Error! Invalid month format. Example: \"Jan\" \"Feb\", \"Mar\" etc. \n";

sub = input.substr(8, 2);

int intForCheck = stoi(sub);

if (intForCheck >= 1 && intForCheck <= 31)

newStudent.changedOn.dayOfMonth = intForCheck;

else

throw "Error! Invalid day of month format. Example: \"18\", 1-31. \n";

sub = input.substr(11, 2);

intForCheck = stoi(sub);

if (intForCheck >= 0 && intForCheck <= 24)

newStudent.changedOn.time.hour = sub;

else

throw "Error! Invalid hour format. Example: \"13\", 00-23. \n";

sub = input.substr(14, 2);

intForCheck = stoi(sub);

if (intForCheck >= 0 && intForCheck <= 59)

newStudent.changedOn.time.minute = sub;

else

throw "Error! Invalid minute format. Example: \"29\", 00-59. \n";

sub = input.substr(17, 2);

intForCheck = stoi(sub);

if (intForCheck >= 0 && intForCheck <= 59)

newStudent.changedOn.time.second = sub;

else

throw "Error! Invalid second format. Example: \"29\", 00-59. \n";

newStudent.changedOn.year = stoi(input.substr(20, 4));

return newStudent;

}

student readStudentFromKeyboard() {

struct student newStudent;

string input;

bool inputIsCorrect = false;

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

do {

cout << "Enter full fame. Format example: \"Ivanov Ivan Ivanovich\", only letters and only 3 words. \n>> ";

getline(cin, input);

if (isFullName(input)) { // first option of check

newStudent.fullName = input;

inputIsCorrect = true;

}

else {

inputIsCorrect = false;

cout << "Error! Invalid students full fame. Format example: \"Ivanov Ivan Ivanovich\", only letters and only 3 words. \n";

}

} while (!inputIsCorrect);

do {

cout << "Enter number of group. Format example: \"2284\", only numbers. \n>> ";

getline(cin, input);

regex mask("^[0-9]\*$"); // second option of check

if (regex\_search(input, mask)) {

newStudent.group = stoi(input);

inputIsCorrect = true;

}

else {

inputIsCorrect = false;

cout << "Error! Invalid number of group format. Example: \"2284\", only numbers. \n";

}

} while (!inputIsCorrect);

do {

cout << "Enter student id. Format example: \"12\", only numbers. \n>> ";

getline(cin, input);

regex mask("^[0-9]\*$");

if (regex\_search(input, mask)) {

newStudent.id = stoi(input);

inputIsCorrect = true;

}

else {

inputIsCorrect = false;

cout << "Error! Invalid student id format. Example: \"12\", only numbers. \n";

}

} while (!inputIsCorrect);

do {

cout << "Enter student sex. Format example: only \"M\" or \"F\". \n>> ";

getline(cin, input);

regex mask("^[MF|mf]{1}$");

if (regex\_search(input, mask)) {

newStudent.sex = input[0];

inputIsCorrect = true;

}

else {

inputIsCorrect = false;

cout << "Error! Invalid student sex format. Example: only \"M\" or \"F\". \n";

}

} while (!inputIsCorrect);

do {

cout << "Enter form of education. Format example: \"distance\", without numbers and another characters. \n>> ";

getline(cin, input);

regex mask("^[A-Za-z]\*$");

if (regex\_search(input, mask)) {

newStudent.educationForm = input;

inputIsCorrect = true;

}

else {

inputIsCorrect = false;

cout << "Error! Invalid form of education format. Example: \"distance\", without numbers and another characters. \n";

}

} while (!inputIsCorrect);

do {

cout << "Enter grades. Format example: \"3 4 2 5 5 3 4 0 \", only 8 grades, grade can be only 0-5, where \"0\" is empty grade. \n>> ";

getline(cin, input);

regex mask("^[0-5]{1} {1}[0-5]{1} {1}[0-5]{1} {1}[0-5]{1} {1}[0-5]{1} {1}[0-5]{1} {1}[0-5]{1} {1}[0-5]{1}");

if (regex\_search(input, mask)) {

for (int i = 0, j = 0; i <= 14; i += 2) {

newStudent.grades[j] = (int)input[i] - 48;

j++;

}

inputIsCorrect = true;

}

else {

inputIsCorrect = false;

cout << "Error! Invalid grades format. Example: \"3 4 2 5 5 3 4 0 \", only 8 grades, grade can be only 0-5, where \"0\" is empty grade. \n";

}

} while (!inputIsCorrect);

do {

try {

cout << "Enter date. Format example: \"Mon Feb 22 12:00:01 2021\". Be careful, strict compliance with the format is required! \n>> ";

getline(cin, input);

regex mask("^[A-Za-z]{3} {1}[A-Za-z]{3} {1}[0-9]{2} {1}[0-9]{2}:{1}[0-9]{2}:{1}[0-9]{2} {1}[0-9]{4}$");

if (!regex\_search(input, mask))

throw "Error! Invalid date format. Example: \"Mon Feb 22 12:00:01 2021\". \n";

string sub = input.substr(0, 3); // Here and below individual cases of incorrect date are checked

if (isDay(sub))

newStudent.changedOn.dayOfWeek = sub;

else

throw "Error! Invalid day format. Example: \"Mon\" \"Tue\", \"Wed\" etc. \n";

sub = input.substr(4, 3);

if (isWeek(sub))

newStudent.changedOn.month = sub;

else

throw "Error! Invalid month format. Example: \"Jan\" \"Feb\", \"Mar\" etc. \n";

sub = input.substr(8, 2);

int intForCheck = stoi(sub);

if (intForCheck >= 1 && intForCheck <= 31)

newStudent.changedOn.dayOfMonth = intForCheck;

else

throw "Error! Invalid day of month format. Example: \"18\", 1-31. \n";

sub = input.substr(11, 2);

intForCheck = stoi(sub);

if (intForCheck >= 0 && intForCheck <= 24)

newStudent.changedOn.time.hour = sub;

else

throw "Error! Invalid hour format. Example: \"13\", 00-23. \n";

sub = input.substr(14, 2);

intForCheck = stoi(sub);

if (intForCheck >= 0 && intForCheck <= 59)

newStudent.changedOn.time.minute = sub;

else

throw "Error! Invalid minute format. Example: \"29\", 00-59. \n";

sub = input.substr(17, 2);

intForCheck = stoi(sub);

if (intForCheck >= 0 && intForCheck <= 59)

newStudent.changedOn.time.second = sub;

else

throw "Error! Invalid second format. Example: \"29\", 00-59. \n";

newStudent.changedOn.year = stoi(input.substr(20, 4));

inputIsCorrect = true;

}

catch (const char\* msg) {

cout << msg;

continue;

}

} while (!inputIsCorrect);

return newStudent;

}

student \* addStruct(student \* obj, int \* size) {

student \* tempObj = new student[\*size + 1];

for (int i = 0; i < \*size; i++) {

tempObj[i] = obj[i];

}

delete[] obj;

obj = tempObj;

\*size = \*size + 1;

return obj;

}

void addStudentsToFile(ofstream \* file, student \* students, int size) {

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

\*file << students[i].fullName << endl

<< students[i].group << endl

<< students[i].id << endl

<< students[i].sex << endl

<< students[i].educationForm << endl

<< students[i].grades[0] << ' ' << students[i].grades[1] << ' ' << students[i].grades[2] << ' ' << students[i].grades[3] << ' ' << students[i].grades[4] << ' ' << students[i].grades[5] << ' ' << students[i].grades[6] << ' ' << students[i].grades[7] << " \n"

<< students[i].changedOn.dayOfWeek << ' ' << students[i].changedOn.month << ' ' << students[i].changedOn.dayOfMonth << ' ' << students[i].changedOn.time.hour << ':' << students[i].changedOn.time.minute << ':' << students[i].changedOn.time.second << ' ' << students[i].changedOn.year << endl;

}

}

void printStudent(student input) {

cout << input.fullName << endl;

cout << input.group << endl;

cout << input.id << endl;

cout << input.sex << endl;

cout << input.educationForm << endl;

printGrades(input.grades);

printDate(input.changedOn);

}

int searchStudent(student \* students, string input, int size) {

int result = -1;

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

if (students[i].fullName == input) {

result = i;

break;

}

return result;

}

void printTop(student \* students, int size) {

float \* avgGrades = new float[size];

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

avgGrades[i] = (students[i].grades[0] + students[i].grades[1] + students[i].grades[2] + students[i].grades[3] + students[i].grades[4] + students[i].grades[5] + students[i].grades[6] + students[i].grades[7]) / (float)8;

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

float maxGrade = 0;

int indexOfMaxGrade = -1;

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

if (avgGrades[i] > 0 && avgGrades[i] > maxGrade) {

maxGrade = avgGrades[i];

indexOfMaxGrade = i;

}

cout << "Avg grade is " << avgGrades[indexOfMaxGrade] << ": \n";

printStudent(students[indexOfMaxGrade]);

avgGrades[indexOfMaxGrade] = 0;

cout << endl;

}

}

int countOfMale(student \* students, int size) {

int result = 0;

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

if (students[i].sex == 'M' || students[i].sex == 'm')

result++;

return result;

}

int countOfFemale(student \* students, int size) {

int result = 0;

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

if (students[i].sex == 'F' || students[i].sex == 'f')

result++;

return result;

}

int countOfScholarships(student \* students, int size) {

int result = 0;

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

if (students[i].educationForm == "day"

&& students[i].grades[0] > 3

&& students[i].grades[1] > 3

&& students[i].grades[2] > 3

&& students[i].grades[3] > 3

&& students[i].grades[4] > 3

&& students[i].grades[5] > 3

&& students[i].grades[6] > 3

&& students[i].grades[7] > 3)

result++;

return result;

}

int countOfGoodStudents(student \* students, int size) {

int result = 0;

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

if (students[i].grades[0] > 3

&& students[i].grades[1] > 3

&& students[i].grades[2] > 3

&& students[i].grades[3] > 3

&& students[i].grades[4] > 3

&& students[i].grades[5] > 3

&& students[i].grades[6] > 3

&& students[i].grades[7] > 3)

result++;

return result;

}

int countOfExcellentStudents(student \* students, int size) {

int result = 0;

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

if (students[i].grades[0] > 4

&& students[i].grades[1] > 4

&& students[i].grades[2] > 4

&& students[i].grades[3] > 4

&& students[i].grades[4] > 4

&& students[i].grades[5] > 4

&& students[i].grades[6] > 4

&& students[i].grades[7] > 4)

result++;

return result;

}

void printStudentsFromDate(student \* students, date input, int size) {

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

if (students[i].changedOn.month == input.month

&& students[i].changedOn.dayOfMonth == input.dayOfMonth

&& students[i].changedOn.year == input.year)

printStudent(students[i]);

}

void printStudentsFromDateBeforeNoon(student \* students, date input, int size) {

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

if (students[i].changedOn.month == input.month

&& students[i].changedOn.dayOfMonth == input.dayOfMonth

&& students[i].changedOn.year == input.year

&& stoi(students[i].changedOn.time.hour) < 12)

printStudent(students[i]);

}

void printStudentsFromDateAfterNoon(student \* students, date input, int size) {

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

if (students[i].changedOn.month == input.month

&& students[i].changedOn.dayOfMonth == input.dayOfMonth

&& students[i].changedOn.year == input.year

&& stoi(students[i].changedOn.time.hour) > 12)

printStudent(students[i]);

}

date readFullTimeFromKeyboard() {

bool isCorrected = false;

date tempDate;

do {

cout << "Enter date of change. Example: \"Mon Feb 22 12:00:01 2021\". \n>> ";

string input;

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

getline(cin, input);

regex mask("^[A-Za-z]{3} {1}[A-Za-z]{3} {1}[0-9]{2} {1}[0-9]{2}:{1}[0-9]{2}:{1}[0-9]{2} {1}[0-9]{4}$");

if (!regex\_search(input, mask)) {

cout << "Error! Invalid date format. Example: \"Mon Feb 22 12:00:01 2021\". \n";

continue;

}

string sub = input.substr(0, 3); // Here and below individual cases of incorrect date are checked

if (isDay(sub))

tempDate.dayOfWeek = sub;

else {

cout << "Error! Invalid day format. Example: \"Mon\" \"Tue\", \"Wed\" etc. \n";

continue;

}

sub = input.substr(4, 3);

if (isWeek(sub))

tempDate.month = sub;

else {

cout << "Error! Invalid month format. Example: \"Jan\" \"Feb\", \"Mar\" etc. \n";

continue;

}

sub = input.substr(8, 2);

int intForCheck = stoi(sub);

if (intForCheck >= 1 && intForCheck <= 31)

tempDate.dayOfMonth = intForCheck;

else {

cout << "Error! Invalid day of month format. Example: \"18\", 1-31. \n";

continue;

}

sub = input.substr(11, 2);

intForCheck = stoi(sub);

if (intForCheck >= 0 && intForCheck <= 24)

tempDate.time.hour = sub;

else {

cout << "Error! Invalid hour format. Example: \"13\", 00-23. \n";

continue;

}

sub = input.substr(14, 2);

intForCheck = stoi(sub);

if (intForCheck >= 0 && intForCheck <= 59)

tempDate.time.minute = sub;

else {

cout << "Error! Invalid minute format. Example: \"29\", 00-59. \n";

continue;

}

sub = input.substr(17, 2);

intForCheck = stoi(sub);

if (intForCheck >= 0 && intForCheck <= 59)

tempDate.time.second = sub;

else {

cout << "Error! Invalid second format. Example: \"29\", 00-59. \n";

continue;

}

tempDate.year = stoi(input.substr(20, 4));

isCorrected = true;

} while (!isCorrected);

return tempDate;

}

date readDateFromKeyboard() {

bool isCorrected = false;

date tempDate;

do {

cout << "Enter date of change. Example: \"Feb 22 2021\". \n>> ";

string input;

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

getline(cin, input);

regex mask("^[A-Za-z]{3} {1}[0-9]{2} {1}[0-9]{4}$");

if (!regex\_search(input, mask)) {

cout << "Error! Invalid date format. Example: \"Feb 22 2021\". \n";

continue;

}

string sub = input.substr(0, 3);

if (isWeek(sub))

tempDate.month = sub;

else {

cout << "Error! Invalid month format. Example: \"Jan\" \"Feb\", \"Mar\" etc. \n";

continue;

}

sub = input.substr(4, 2);

int intForCheck = stoi(sub);

if (intForCheck >= 1 && intForCheck <= 31)

tempDate.dayOfMonth = intForCheck;

else {

cout << "Error! Invalid day of month format. Example: \"18\", 1-31. \n";

continue;

}

tempDate.year = stoi(input.substr(7, 4));

isCorrected = true;

} while (!isCorrected);

return tempDate;

}

bool isFullName(string str) {

int countOfSpaces = 0;

int length = str.length();

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

if (isLetter(str[i]))

continue;

else if ((str[i] == ' ') && (countOfSpaces < 2))

countOfSpaces++;

else {

return false;

}

}

return true;

}

bool isLetter(char c) {

if (

((c >= 65) && (c <= 90))

|| ((c >= 97) && (c <= 122))

)

return true;

else

return false;

}

bool isDay(string str) {

if (str == "Mon" || str == "mon"

|| str == "Tue" || str == "tue"

|| str == "Wed" || str == "wed"

|| str == "Thu" || str == "thu"

|| str == "Fri" || str == "fri"

|| str == "Sat" || str == "sat"

|| str == "Sun" || str == "sun"

)

return true;

else

return false;

}

bool isWeek(string str) {

if (str == "Jan" || str == "jan"

|| str == "Feb" || str == "feb"

|| str == "Mar" || str == "mar"

|| str == "Apr" || str == "apr"

|| str == "May" || str == "may"

|| str == "Jun" || str == "jun"

|| str == "Jul" || str == "jul"

|| str == "Aug" || str == "aug"

|| str == "Sep" || str == "sep"

|| str == "Oct" || str == "oct"

|| str == "Nov" || str == "nov"

|| str == "Dec" || str == "dec"

)

return true;

else

return false;

}

void printDate(date input) {

cout << input.dayOfWeek << ' ' << input.month << ' ' << input.dayOfMonth << ' ' << input.time.hour << ':' << input.time.minute << ':' << input.time.second << ' ' << input.year << endl;

}

void printGrades(short \* input) {

cout << input[0] << ' ' << input[1] << ' ' << input[2] << ' ' << input[3] << ' ' << input[4] << ' ' << input[5] << ' ' << input[6] << ' ' << input[7] << ' ' << endl;

}

#include <iostream>

#include <fstream>

#include <iomanip>

#include <string>

#include <chrono>

using namespace std::chrono;

using namespace std;

struct list

{

int value;

list \* before;

list \* after;

};

// main functions

int countOfElements(ifstream \* file);

// array functions

void fillArrayFromFile(ifstream \* file, int \* arr, int arrSize);

int getArrayElement(int \* arr, int size, int input);

int \* increaseArray(int \* obj, int \* size);

int \* decreaseArray(int \* obj, int \* size);

void insertToArr(int \* arr, int size, int value, int index);

void deleteArrElement(int \* arr, int size, int index);

void printArr(int \* arr, int size);

// list functions

list \* createList(int size);

void fillListFromFile(ifstream \* file, list \* list);

void printList(list \*list);

int getListElementByIndex(list \*dlList, int index);

int getListElementByValue(list \*dlList, int value);

list \* insertToList(list \*dlList, int value, unsigned index);

void deleteListElementByIndex(list \* &dlList, int index);

void deleteListElementByValue(list \* &dlList, int value);

//secondary functions

bool isDigit(char input);

float stopSecondsTimer(time\_point<steady\_clock> startTimer);

long long stopNanoSecondsTimer(time\_point<steady\_clock> startTimer);

bool choiseNextAction();

void practicalWork2() {

// Timer

srand((unsigned)time(NULL));

time\_point<steady\_clock> startTimer = steady\_clock::now();

float stopDynTime = stopSecondsTimer(startTimer);

float stopListTime = stopSecondsTimer(startTimer);

long long stopDynNanoTime = stopNanoSecondsTimer(startTimer);

long long stopListNanoTime = stopNanoSecondsTimer(startTimer);

system("CLS");

cout << "Solution of task \"Working with Dynamic Massives and Doubly Linked Lists\". \n\n" <<

"\tReading file and creating dynamic massive and doubly linked list. \n";

ifstream ifile;

ofstream ofile;

string path;

int numberOfElements;

int \* dynArr = 0;

list \* dlList = 0;

do { // file opening

cout << "\nEnter the path to the file. \n"

<< "Only english words in the file and path! Example: C:\\anime\\pr2testFile.txt \n>> ";

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

getline(cin, path);

ifile.open(path);

if (!ifile.is\_open()) {

cout << "Error opening file! Please restart the program! \n";

throw - 1;

}

ifstream \*pifile = &ifile;

numberOfElements = countOfElements(pifile);

if (numberOfElements == 0) {

cout << "Error! No elements founds.\n";

system("pause");

continue;

}

// working with an array

ifile.close();

ifile.open(path);

startTimer = steady\_clock::now();

dynArr = new int[numberOfElements];

fillArrayFromFile(pifile, dynArr, numberOfElements);

stopDynTime = stopSecondsTimer(startTimer);

// working with a doubly linked list

ifile.close();

ifile.open(path);

startTimer = steady\_clock::now();

dlList = createList(numberOfElements);

fillListFromFile(pifile, dlList);

stopListTime = stopSecondsTimer(startTimer);

cout << "\nDynamic array has been initialized and filled in " << fixed << stopDynTime << " second(s).";

cout << "\nDoubly linked list has been initialized and filled in " << fixed << stopListTime << " second(s). \n";

system("pause");

// uncomment to debug

//printList(dlList);

//printArr(dynArr, numberOfElements);

do {

cout << "\n\tDynamic array and doubly linked list operations. \n"

<< "1) Getting and element\n"

<< "2) Element insertion\n"

<< "3) Deleting an element\n"

<< "Enter option number... \n>> ";

int input;

cin >> input;

switch (input) {

case 1:

cout << "\nTypes of getting: \n"

<< "1) On index\n"

<< "2) On value\n"

<< "Enter option number... \n>> ";

cin >> input;

switch (input) {

case 1:

cout << "\nEnter the index of element... \n>> ";

cin >> input;

if ((input < numberOfElements) && (input >= 0))

{

// working with an array

startTimer = steady\_clock::now();

int dynArrResult = dynArr[input];

stopDynNanoTime = stopNanoSecondsTimer(startTimer);

// working with a doubly linked list

startTimer = steady\_clock::now();

int dlListResult = getListElementByIndex(dlList, input);

stopListNanoTime = stopNanoSecondsTimer(startTimer);

cout << "\n" << input << " element has containing the value " << dynArrResult << ". \n";

// uncomment to debug

//cout << input << " element has containing the value " << dlListResult << ". \n";

cout << "\nElement found in dynamic array for " << fixed << stopDynNanoTime << " nanosecond(s). \n";

cout << "Element found in doubly linked list for " << fixed << stopListNanoTime << " nanosecond(s). \n";

}

else cout << "Error! Array has not contain that index!\n";

break;

case 2:

cout << "\nEnter the value of element... \n>> ";

cin >> input;

// working with an array

startTimer = steady\_clock::now();

int dynArrResult = getArrayElement(dynArr, numberOfElements, input);

stopDynNanoTime = stopNanoSecondsTimer(startTimer);

// working with a doubly linked list

startTimer = steady\_clock::now();

int dlListResult = getListElementByValue(dlList, input);

stopListNanoTime = stopNanoSecondsTimer(startTimer);

if ((dynArrResult != -1) && (dlListResult != -1)) {

cout << "\nThe first found element with a value of " << input << " has an index " << dynArrResult << ".\n";

// uncomment to debug

//cout << "\nThe first found element with a value of " << input << " has an index " << dlListResult << ".\n";

cout << "Element found in dynamic array for " << fixed << stopDynNanoTime << " nanosecond(s). \n";

cout << "Element found in doubly linked list for " << fixed << stopListNanoTime << " nanosecond(s). \n";

}

else cout << "Error! The element has not found. \n";

break;

}

break;

case 2:

cout << "\nEnter the insertion value... \n>> ";

cin >> input;

int inputIndex;

cout << "\nEnter the index of insertion... \n>> ";

cin >> inputIndex;

if ((inputIndex >= 0) && (inputIndex <= numberOfElements))

{

// working with an array

startTimer = steady\_clock::now();

dynArr = increaseArray(dynArr, &numberOfElements);

insertToArr(dynArr, numberOfElements, input, inputIndex);

stopDynNanoTime = stopNanoSecondsTimer(startTimer);

// working with a doubly linked list

startTimer = steady\_clock::now();

dlList = insertToList(dlList, input, inputIndex);

stopListNanoTime = stopNanoSecondsTimer(startTimer);

cout << "Element has been inserted in dynamic array for " << fixed << stopDynNanoTime << " nanosecond(s). \n";

cout << "Element has been inserted in doubly linked list for " << fixed << stopListNanoTime << " nanosecond(s). \n";

// uncomment to debug

//printArr(dynArr, numberOfElements);

//printList(dlList);

}

else cout << "Error! Array has not contain that index!\n"

<< "If you need an insert after all elements, the index must be equal to the number of elements.\n";

break;

case 3:

cout << "\nTypes of deleting: \n"

<< "1) On index\n"

<< "2) On value\n"

<< "Enter option number... \n>> ";

cin >> input;

switch (input) {

case 1:

cout << "\nEnter the index of the element... \n>> ";

cin >> input;

if ((input < numberOfElements) && (input >= 0))

{

// working with an array

startTimer = steady\_clock::now();

deleteArrElement(dynArr, numberOfElements, input);

dynArr = decreaseArray(dynArr, &numberOfElements);

stopDynNanoTime = stopNanoSecondsTimer(startTimer);

// working with a doubly linked list

startTimer = steady\_clock::now();

deleteListElementByIndex(dlList, input);

stopListNanoTime = stopNanoSecondsTimer(startTimer);

cout << "Element has been deleted from dynamic array for " << fixed << stopDynNanoTime << " nanosecond(s). \n";

cout << "Element has been deleted from doubly linked list for " << fixed << stopListNanoTime << " nanosecond(s). \n";

// uncomment to debug

//printArr(dynArr, numberOfElements);

//printList(dlList);

}

else cout << "Error! Array has not contain that element\n";

break;

case 2:

cout << "\nEnter the value of the element... \n>> ";

cin >> input;

// working with an array

startTimer = steady\_clock::now();

int index = getArrayElement(dynArr, numberOfElements, input);

if (index != -1)

{

deleteArrElement(dynArr, numberOfElements, index);

dynArr = decreaseArray(dynArr, &numberOfElements);

stopDynNanoTime = stopNanoSecondsTimer(startTimer);

// working with a doubly linked list

startTimer = steady\_clock::now();

deleteListElementByValue(dlList, input);

stopListNanoTime = stopNanoSecondsTimer(startTimer);

cout << "Element has been deleted from dynamic array for " << fixed << stopDynNanoTime << " nanosecond(s). \n";

cout << "Element has been deleted from doubly linked list for " << fixed << stopListNanoTime << " nanosecond(s). \n";

// uncomment to debug

//printArr(dynArr, numberOfElements);

//printList(dlList);

}

else cout << "Error! The element has not found. \n";

break;

}

break;

}

cout << "\nEnd of dynamic array and doubly linked list operations. \n";

} while (choiseNextAction());

} while (!ifile.is\_open());

ifile.close();

}

int countOfElements(ifstream \* file)

{

string t;

int count = 0;

while (std::getline(\*file, t)) {

if (t.length() > 1)

for (unsigned i = 0; i < static\_cast<unsigned>(t.length() - 1); i++)

if (((t[i] == ' ') || (t[i] == '\t')) && (isDigit(t[i + 1])))

count++; // Counting the number of "spaces/tabs+digits"

count++; // counting the number of linebreakes

}

return count;

}

void fillArrayFromFile(ifstream \* file, int \* arr, int arrSize)

{

int arrPos = 0;

while ((arrPos < arrSize) && (!file->eof()))

{

// Getting a string with numbers from file

string tStr;

getline(\*file, tStr);

// Extracting a numbers from string to array

for (int unsigned tStrPos = 0; tStrPos < tStr.length(); tStrPos++)

{

char \* tNum = new char[tStr.length()];

int tNumPos = 0;

while (

(tStr[tStrPos] != ' ') &&

(tStr[tStrPos] != '\t') &&

(tStrPos < tStr.length())

) {

tNum[tNumPos] = tStr[tStrPos];

tStrPos++;

tNumPos++;

}

arr[arrPos] = stoi(tNum);

arrPos++;

}

}

}

int getArrayElement(int \* arr, int size, int input)

{

int result = -1;

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

if (arr[i] == input) {

result = i;

break;

}

return result;

}

int \* increaseArray(int \* obj, int \* size)

{

int \* tempObj = new int[\*size + 1];

for (int i = 0; i < \*size; i++) {

tempObj[i] = obj[i];

}

delete[] obj;

obj = tempObj;

\*size = \*size + 1;

return obj;

}

int \* decreaseArray(int \* obj, int \* size)

{

int \* tempObj = new int[\*size - 1];

for (int i = 0; i < \*size - 1; i++) {

tempObj[i] = obj[i];

}

delete[] obj;

obj = tempObj;

\*size = \*size - 1;

return obj;

}

void insertToArr(int \* arr, int size, int value, int index)

{

if (!arr[index])

{

arr[index] = value;

return;

}

for (int i = size - 1; i > index; i--)

arr[i] = arr[i - 1];

arr[index] = value;

}

void deleteArrElement(int \* arr, int size, int index)

{

if (index == (size - 1))

return;

for (int i = index; i < size; i++)

arr[i] = arr[i + 1];

}

void printArr(int \* arr, int size)

{

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

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

{

cout << arr[i] << ' ';

}

cout << "\n--- End of Debug ---\n";

system("pause");

}

bool isDigit(char input) {

if (input >= 48 && input <= 57)

return true;

else return false;

}

list \* createList(int size) {

list \* current = 0, \* after = 0;

for (int i = 1; i <= size; i++)

{

current = new list;

current->after = after; // adding an address of next element

if (after)

after->before = current; // writing an address of current element to next element

after = current; // changing position of current element

}

current->before = 0;

return current;

}

void fillListFromFile(ifstream \* file, list \* dlList)

{

list \* pos = dlList;

do {

if (file->eof())

return;

// Getting a string with numbers from file

string tStr;

getline(\*file, tStr);

// Extracting a numbers from string to array

for (int unsigned tStrPos = 0; tStrPos < tStr.length(); tStrPos++)

{

char \* tNum = new char[tStr.length()];

int tNumPos = 0;

while (

(tStr[tStrPos] != ' ') &&

(tStr[tStrPos] != '\t') &&

(tStrPos < tStr.length())

) {

tNum[tNumPos] = tStr[tStrPos];

tStrPos++;

tNumPos++;

}

pos->value = stoi(tNum);

pos = pos->after;

}

} while (pos);

}

void printList(list \*dlList)

{

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

list \*pos = dlList;

do {

cout << pos->value << ' ';

pos = pos->after;

} while (pos);

cout << "\n--- End of Debug ---\n";

system("pause");

}

int getListElementByIndex(list \*dlList, int index)

{

list \*pos = dlList;

do {

if (index == 0)

return pos->value;

else

pos = pos->after;

index--;

} while (pos);

return 0;

}

int getListElementByValue(list \*dlList, int value)

{

int index = 0;

list \*pos = dlList;

do {

if (pos->value == value)

return index;

else {

pos = pos->after;

index++;

}

} while (pos);

return -1;

}

list \* insertToList(list \*dlList, int value, unsigned index)

{

list \* newElement = new list;

newElement->value = value;

list \*pos = dlList;

do {

if (index == 0)

{

if (pos->before == 0)

{

newElement->before = 0;

newElement->after = pos;

pos->before = newElement;

return newElement;

}

else

{

newElement->before = pos->before;

newElement->after = pos;

pos->before->after = newElement;

pos->before = newElement;

return dlList;

}

}

else if ((index == 1) && (pos->after == 0)) {

newElement->after = 0;

newElement->before = pos;

pos->after = newElement;

return dlList;

}

else

pos = pos->after;

index--;

} while (pos);

return 0;

}

// from lesson, not worked

//list \* insertToList(list \*dlList, int value, unsigned index)

//{

// list \* newElement = new list;

// newElement->value = value;

//

// if (!index || !dlList)

// {

// dlList->before = dlList;

// newElement->before = 0;

// newElement->after = dlList;

// dlList = newElement;

// return dlList;

// }

//

// list \* beforePos = dlList;

// index--;

// while ((beforePos->after) && (index--))

// beforePos = beforePos->after;

//

// newElement->before = beforePos;

// newElement->after->before = newElement;

// newElement->after = beforePos->after;

// beforePos->after = newElement;

// return newElement;

//}

void deleteListElementByIndex(list \* &dlList, int index)

{

list \* pos = dlList;

do

{

if (index == 0)

if (pos->before == 0)

{

pos = dlList->after;

pos->before = 0;

delete dlList;

dlList = pos;

return;

}

else if (pos->after == 0)

{

pos->before->after = 0;

delete pos;

return;

}

else

{

pos->before->after = pos->after;

pos->after->before = pos->before;

delete pos;

return;

}

else

{

pos = pos->after;

index--;

}

} while (pos);

}

void deleteListElementByValue(list \*&dlList, int value)

{

list \* pos = dlList;

do

{

if (pos->value == value)

if (pos->before == 0)

{

pos = dlList->after;

pos->before = 0;

delete dlList;

dlList = pos;

return;

}

else if (pos->after == 0)

{

pos->before->after = 0;

delete pos;

return;

}

else

{

pos->before->after = pos->after;

pos->after->before = pos->before;

delete pos;

return;

}

else {

pos = pos->after;

}

} while (pos);

}