Національний університет “Львівська політехніка”

Кафедра програмного забезпечення

**КУРСОВА РОБОТА**

**з дисципліни «Об’єктно-орієнтоване програмування»**

**На тему:  
«***Таблиця пацієнтів-донорів крові***»**

Студентки групи  ПЗ-23

спеціальності 6.121

“Інженерія програмного забезпечення”

Ломницької С. І.

Керівник: доцент кафедри ПЗ,

к.т.н., доцент Коротєєва Т. О.

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів\_\_\_Оцінка ECTS\_\_\_

Члени комісії          \_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Львів 2022

ЗАВДАННЯ

до курсової роботи з дисципліни «Об’єктно-орієнтоване програмування»

студентки групи ПЗ-23 Ломницької Соломії

***Тема: «Таблиця пацієнтів-донорів крові»***

**Завдання:** (Варіант 14)

Створити таблицю у візуальному середовищі

№ | Прізвище | Вік | Група крові | Резус-фактор | Артеріальний тиск | Пульс

1. Швидким алгоритмом відсортувати записи за показником артеріального тиску.
2. Згрупувати людей за однаковими групами крові та однаковими резус-факторами.
3. Згрупувати людей за однаковими резус-факторами та відсортувати кожну групу за показником Пульсу.
4. Визначити людей, які є універсальними донорами, а які є універсальними реципієнтами та сформувати загальну таблицю донорів та реципієнтів.
5. Для вказаного показника Вік визначити пацієнтів з підвищеними показниками артеріального тиску та пульсу.
6. Всім пацієнтам з нормальними артеріальним тиском вивести повідомлення «*Прізвище*---- Здоровий! ---»

Для [клас](http://vns.lpnu.ua/mod/glossary/showentry.php?eid=54317&displayformat=dictionary)у створити: 1) [Конструктор](http://vns.lpnu.ua/mod/glossary/showentry.php?eid=54330&displayformat=dictionary) за замовчуванням; 2) [Конструктор](http://vns.lpnu.ua/mod/glossary/showentry.php?eid=54330&displayformat=dictionary) з параметрами; 3) [конструктор](http://vns.lpnu.ua/mod/glossary/showentry.php?eid=54330&displayformat=dictionary) копій; 4) перевизначити операції >>, << для зчитування та запису у файл.

**Зміст завдання та календарний план його виконання**

|  |
| --- |
|  |
| **№ з/п** | **Зміст завдання** | **Дата** |
| 1 | Здійснити аналiтичний огляд лiтератури за заданою темою та обгрунтувати вибір інструментальних засобів реалізації. | 12.09.2022 |
| 2 | Побудова UML діаграм | 28.09.2022 |
| 3 | Розробка алгоритмів реалізації | 29.09.2022 |
| 4 | Реалізація завдання (кодування) | 14.10.2022 |
| 5 | Формування інструкції користувача | 20.10.2022 |
| 6 | Оформлення звіту до курсової роботи згідно з вимогами Міжнародних стандартів, дотримуючись такої структури:  ·        зміст;  ·        алгоритм розв‘язку задачі у покроковому представленні;  ·        діаграми UML класів, прецедентів, послідовності виконання;  ·        код розробленої програми з коментарями;  ·        протокол роботи програми для кожного пункту завдання  ·        інструкція користувача та системні вимоги;  ·        опис виняткових ситуацій;  ·        структура файлу вхідних даних;  ·        висновки;  ·        список використаних джерел. | 21.10.2022 |

Завдання прийнято до виконання: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Ломницька С.І.)

Керівник роботи: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ Коротєєва Т. О./

**Зміст**

1. [Алгоритм розв'язку задачі у покроковому представленні 5](#_Toc117690035)
2. [Діаграми UML класів, прецедентів, послідовності виконання 11](#_Toc117690036)
3. [Код розробленої програми з коментарями 14](#_Toc117690037)
4. [Протокол роботи для кожного пункту завдання 48](#_Toc117690038)
5. [Інструкція користувача та системні вимоги 58](#_Toc117690039)
6. [Опис виняткових ситуацій 68](#_Toc117690040)
7. [Структура файлу вхідних даних 71](#_Toc117690041)

[Висновки 72](#_Toc117690042)

[Список використаних джерел 73](#_Toc117690043)

# Алгоритм розв'язку задачі у покроковому представленні

Алгоритм розв'язку задачі загалом

Алгортим ALL використано для роботи з таблицею пацієнтів-донорів крові.

ALL1. Ввід даних у таблицю алгоритмом .

ALL2. Виконання бажаних дій з таблицею. Використовувати алгоритми

ALL3. Збереження результатів у текстовий файл алгоритмом .

ALL4. Кінець. Вихід.

Швидким алгоритмом відсортувати записи за заданим показником.

Алгоритм QSA впорядковує отриманий вектор значень у порядку зростання шляхом рекурсивного розбиття його на дві. Задано масив елементів A1, A2, … An, що містить n елементів. Для усіх елементів має бути справедливим співвідношення Ai< Ai+1 - для всіх i=0,.. n-1, де Ai< Ai+1 визначається за значенням заданого показника. Символи i та j – індекси елементів масиву A. A0 – базовий елемент, p – індекс базового елемента.

QSA1.Початкове присвоєння значень. i = 1, j = n, left = 1, right = n.

QSA2.Перевірка умови – чи і>j. Якщо і>j виконуємо кроки QSA3- QSA10.

QSA3.Визначити базовий елемент A0. A0 = Aj, p = j.

QSA4. Цикл з післяумовою. Поки i<=j повторювати кроки QSA5 – QSA7. Інакше -> QSA8.

QSA5. Цикл. Поки Аi< A0, то i=i+1.

QSA6. Цикл. Поки Аj> A0, то j=j+1.

QSA7. Перевірка умови. Якщо Аj> A0, то j=j-1, інакше переставити місцями елементи Аi та Аj, та присвоїти значення змінним i=i+1, j=j-1.

QSA8. Сортування для лівої частини. i = i; j = p – 1. -> QSA2.

QSA9. Сортування для правої частини. i = p +1; j = j. -> QSA2.

QSA10.Кінець. Вихід.

Формування групи людей з однаковими групами крові та однаковими резус-факторами

Алгоритм FBR дозволяє сформувати масив пацієнтів, що мають дану користувачем групу крові та резус фактор. Задано масив A1, A2, … An, що містить n елементів та результуючий масив В, розмір якого m = 0. Символ i – індекс елементів масиву A, bt та rh – задані група крові та резус фактор відповідно.

FBR1. Перевірка умови – чи вектор не порожній? Якщо n < 1, то видаємо помилку, -> FBR8.

FBR2. Початкове присвоєння значень. i = 1.

FBR3. Цикл для проходження по масиву A. Поки i <= n виконуємо кроки FBR4 - FBR6.

FBR4. Перевірка умови – чи значення полів інформації пацієнта рівні даним користувачем. Якщо група крові A[i] рівна bt і резус фактор A[i] рівний rh, то переходимо до кроку FBR5.

FBR5. Елемент масиву A, що задовольняє умові додаємо в кінець масиву B, m=m+1;

FBR6. Інкрементування. i = i +1.

FBR7. Перевірка умови – чи вектор B - порожній. Якщо m<1, виводимо відповідне повідомлення.

FBR8. Кінець. Вихід.

Формування групи людей з однаковими резус-факторами та її сортування.

Алгоритм FRS дозволяє сформувати посортований масив пацієнтів, що мають даний користувачем резус фактор. Задано масив A1, A2, … An, що містить n елементів та результуючий масив В, розмір якого m = 0. Символ i – індекс елементів масиву A, rh – заданий резус фактор.

FRS1. Перевірка умови – чи вектор не порожній? Якщо n < 1, то видаємо помилку, -> FRS9.

FRS2. Початкове присвоєння значень. i = 1.

FRS3. Цикл для проходження по масиву A. Поки i <= n виконуємо кроки FBR4 та FRS5.

FRS4. Перевірка умови – чи значення полів інформації пацієнта рівні даним користувачем. Якщо резус фактор A[i] рівний rh, то переходимо до кроку FBR5.

FRS5. Елемент масиву A, що задовольняє умові додаємо в кінець масиву B. m=m+1;

FRS6. Інкрементування. i = i +1.

FRS7. Перевірка умови – чи вектор B - порожній. Якщо m<1, виводимо відповідне повідомлення, -> FRS9.

FRS8. Сортування масиву B в порядку зростання за показником пульсу в порядку зростання довільним алгоритмом (наприклад, алгоритмом QSA).

FRS9. Кінець. Вихід.

Пошук універсальних донорів

Алгоритм UD дозволяє сформувати масив пацієнтів, що є універсальними донорами. Задано масив A1, A2, … An, що містить n елементів та результуючий масив В, розмір якого m = 0. Символ i – індекс елементів масиву A.

UD1. Перевірка умови – чи вектор не порожній? Якщо n < 1, то видаємо помилку, -> UD8.

UD2. Початкове присвоєння значень. i = 1.

UD3. Цикл для проходження по масиву A. Поки i <= n виконуємо кроки UD4 – UD6.

UD4. Перевірка умови – чи значення полів інформації пацієнта рівні даним користувачем. Якщо група резус фактор A[i] негативний(«-»), а група крові – 1(O), то переходимо до кроку UD5.

UD5. Елемент масиву A, що задовольняє умові додаємо в кінець масиву B, m=m+1;

UD6. Інкрементування. i = i +1.

UD7. Перевірка умови – чи вектор B - порожній. Якщо m<1, виводимо відповідне повідомлення.

UD8. Кінець. Вихід.

Пошук універсальних реципієнтів

Алгоритм UR дозволяє сформувати масив пацієнтів, що є універсальними реципієнтами. Задано масив A1, A2, … An, що містить n елементів та результуючий масив В, розмір якого m = 0. Символ i – індекс елементів масиву A.

UR1. Перевірка умови – чи вектор не порожній? Якщо n < 1, то видаємо помилку, -> UR8.

UR2. Початкове присвоєння значень. i = 1.

UR3. Цикл для проходження по масиву A. Поки i <= n виконуємо кроки UR4 – UR6.

UR4. Перевірка умови – чи значення полів інформації пацієнта рівні даним користувачем. Якщо група резус фактор A[i] позитивний(«+»), а група крові – 4(AB), то переходимо до кроку UR5.

UR5. Елемент масиву A, що задовольняє умові додаємо в кінець масиву B, m=m+1;

UD6. Інкрементування. i = i +1.

UR7. Перевірка умови – чи вектор B - порожній. Якщо m<1, виводимо відповідне повідомлення.

UR8. Кінець. Вихід.

Пошук пацієнтів з підвищеним тиском та пульсом за заданим віком

Алгоритм FIND дозволяє сформувати масив пацієнтів, що з підвищеним тиском та заданим віком. Задано масив A1, A2, … An, що містить n елементів та результуючий масив В, розмір якого m = 0. Символ i – індекс елементів масиву A, age – заданий користувачем вік.

FIND1. Перевірка умови – чи вектор не порожній? Якщо n < 1, то видаємо помилку, -> FIND8.

FIND2. Початкове присвоєння значень. i = 1.

FIND3. Цикл для проходження по масиву A. Поки i <= n виконуємо кроки FIND4 - FIND6.

FIND4. Перевірка умови – чи значення поля «Вік» інформації студента рівні даним користувачем. Якщо вік пацієнта A[i] рівний заданому, а пульс та тиск більший за верхню межу норми (100 ударів за хвилину для пульсу та 140/90 для тиску) то переходимо на крок FIND5.

FIND5. Елемент масиву A, що задовольняє умові додаємо в кінець масиву B, m=m+1;

FIND6. Інкрементування. i = i +1.

FIND7. Перевірка умови – чи вектор B - порожній. Якщо m<1, виводимо відповідне повідомлення, -> FIND8.

FIND8. Кінець. Вихід.

Перевірка, чи пацієнт може бути донором для іншого пацієнта

Алгоритм СHECK перевіряє, чи може бути пацієнт A донором для пацієнта B.

СHECK1. Перевірка умови – чи сумісні резуси. Якщо резус фактор донора А позитивний, а реципієнта В – негативний, то пацієнт А не може бути донором. -> CHECK5.

СHECK2. Перевірка умови – чи група крові донора перша негативна. Якщо група крові донора А == O - , то пацієнт А може бути донором для даного реципієнта В. -> CHECK5.

СHECK3. Перевірка умови – чи група крові реципієнта четверта позитивна. Якщо група крові реципієнта В == АВ + , то пацієнт А може бути донором для даного реципієнта В. -> CHECK5.

СHECK4. Перевірка умови – чи групи крові обох пацієнтів однакові. Якщо група крові реципієнта В == групі крові донора А , то пацієнт А може бути донором для даного реципієнта В. -> CHECK5.

CHECK5. Кінець. Вихід.

Додавання нового пацієнта

Алгоритм ADD дозволяє додати нового пацієнта у масив A1, A2, … An, що містить n елементів.

ADD1. Перевірка умови – чи зчитування відбувається з файлу. Якщо обрано зчитування з файлу переходимо на крок ADD2, інакше -> ADD7.

ADD2. Відкриваємо заданий користувачем файл.

ADD3. Перевірка умови – чи файл відкрито. Якщо файл не відкрито, то виводимо повідомлення про помилку, -> ADD9.

ADD4. Перевірка умови – чи файл порожній. Якщо файл порожній, то виводимо повідомлення про помилку, -> ADD9.

ADD5. Цикл зчитування об’єктів класу. Поки не досягнуто кінця файлу (EOF) повторювати крок ADD6, інакше -> ADD9 .

ADD6. Зчитати стрічку та зберегти її дані в поля. Додаємо елемент в кінець масиву A1, A2, … An, , n = n+1.

ADD7. Вводимо дані полів з клавіатури.

ADD8. Додаємо елемент в кінець масиву A1, A2, … An, n = n+1.

ADD9.Кінець. Вихід.

Збереження інформації в файл

Алгоритм SAVE дозволяє зберегти інформацію про пацієнтів масиву A1, A2, … An у файл. Символ i – індекс елементів масиву A, що містить n елементів.

SAVE1. Відкриваємо заданий користувачем файл.

SAVE2. Перевірка умови – чи файл відкрито. Якщо файл не відкрито, то виводимо повідомлення про помилку, -> SAVE7.

SAVE3. Початкове присвоєння значень. i = 1.

SAVE4. Цикл для проходження по масиву A. Поки i <= n виконуємо SAVE5.

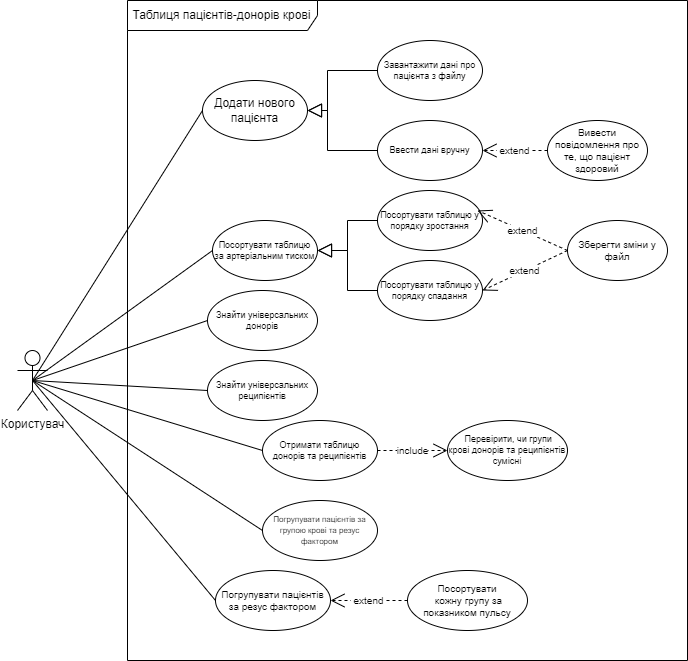
SAVE5. Записуємо значення полів у файл.

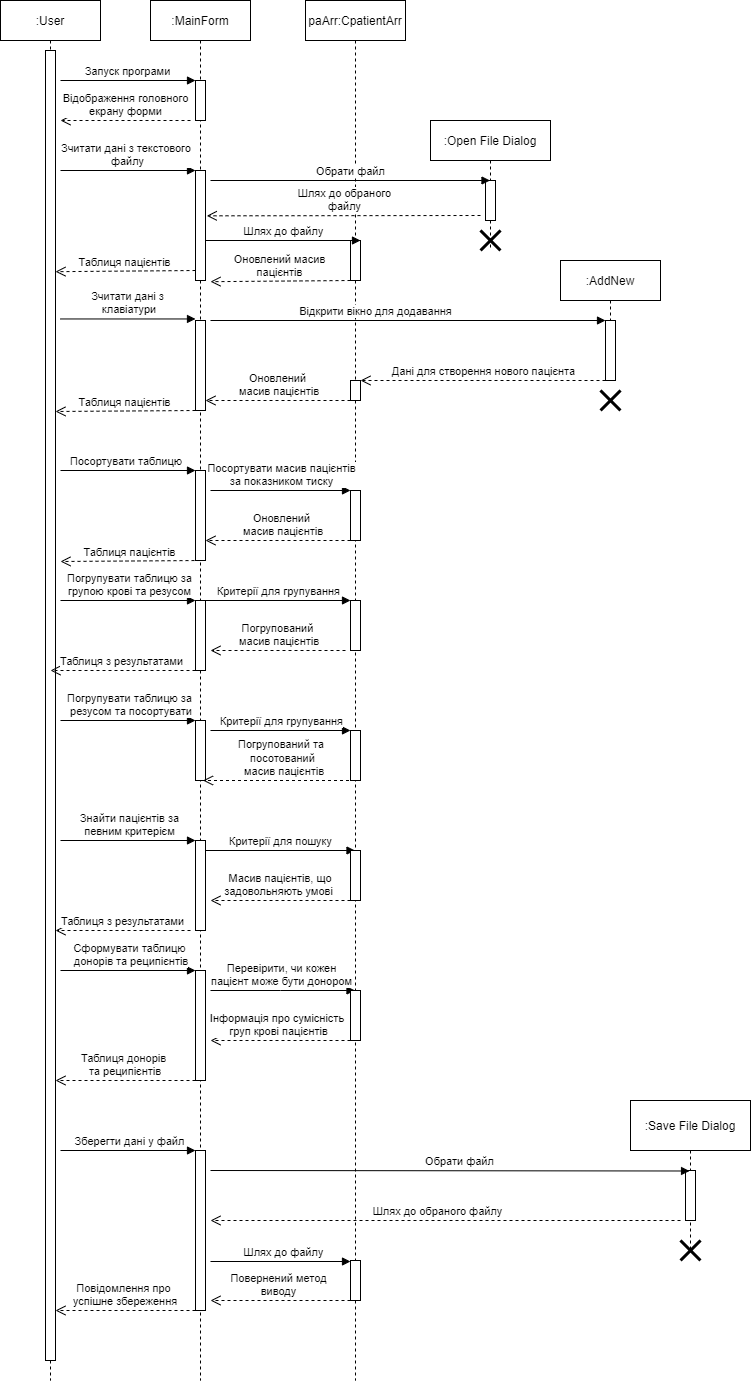
SAVE6.Інкрементування. i = i +1.

SAVE7. Кінець. Вихід.

# 2. Діаграми UML класів, прецедентів, послідовності виконання

*Рис.2.1.Діаграма класів рівня реалізації.*

*Рис.2.2.Діаграма прецедентів.*

*Рис.2.3.Діаграма послідовності.*

# 3. Код розробленої програми з коментарями

**Enumerations.h**

#ifndef EnumerationsH

#define EnumerationsH

#include <iostream>

#include <fstream>

#include <iomanip>

#include <vector>

#include <algorithm>

enum BloodType { O = 1, A = 2, B = 3, AB = 4 };

enum Rhesus { Positive = 0, Negative = 1 };

#endif

**CBloodPressure.h**

#ifndef BloodPressureH

#define BloodPressureH

#include "Enumerations.h"

#include "Cexception.h"

//Class CBloodPressure contains info about patient's blood pressure:

//contains fields systolic and diastolic pressure

class CBloodPressure

{

private:

int m\_iSystolic;

int m\_iDiastolic;

public:

CBloodPressure();

CBloodPressure(int iSystolic, int iDiastolic);

CBloodPressure(const CBloodPressure& bpSource);

CBloodPressure& operator=(const CBloodPressure &piSource);

friend std::ostream& operator<<(std::ostream& osOut, CBloodPressure& bpSource);

friend std::istream& operator>>(std::istream& isIn, CBloodPressure& bpDestination);

int GetSystolicPressure();

void SetSystolicPressure(int iSystolic);

int GetDiastolicPressure();

void SetDiastolicPressure(int iDiastolic);

int CheckPressure();

};

#endif

**CBloodPressure.cpp**

#include "CBloodPressure.h"

#define MAX\_DIASTOLIC 200

#define MAX\_SYTOLIC 300

#define MAX\_NORM\_D 90

#define MAX\_NORM\_S 140

#define MIN\_NORM\_D 60

#define MIN\_NORM\_S 90

//Default constructor without any parameters.

CBloodPressure::CBloodPressure()

{

this->SetDiastolicPressure(1);

this->SetSystolicPressure(1);

}

//Constructor with two parameters.

//Constructor has two integer parameters - systolic and distolic pressure values.

CBloodPressure::CBloodPressure(int iSystolic, int iDiastolic)

{

this->SetDiastolicPressure(iDiastolic);

this->SetSystolicPressure(iSystolic);

}

//Copy constructor.

//Has one parameter - constant reference to the source object.

CBloodPressure::CBloodPressure(const CBloodPressure& bpSource)

:m\_iSystolic(bpSource.m\_iSystolic), m\_iDiastolic(bpSource.m\_iDiastolic)

{

}

//Overloaded operator = which assigns all the values from source object to current object.

//Has one parameter - constant reference to the source object.

CBloodPressure& CBloodPressure::operator=(const CBloodPressure &piSource)

{

this->m\_iSystolic = piSource.m\_iSystolic;

this->m\_iDiastolic = piSource.m\_iDiastolic;

return \*this;

}

//Function GetSystolicPressure() to get value of systolic pressure.

//Returns integer value of systolic pressure.

int CBloodPressure::GetSystolicPressure()

{

return this->m\_iSystolic;

}

//Function SetSystolicPressure() that sets value of systolic pressure.

//Has one parameter - integer value of systiolc pressure.

void CBloodPressure::SetSystolicPressure(int iSystolic)

{

if (iSystolic>0&& iSystolic<MAX\_SYTOLIC)

this->m\_iSystolic = iSystolic;

else

throw Cexception("Range");

}

//Function GetDiastolicPressure() is to get value of diastolic pressure.

//Returns integer value of diastolic pressure.

int CBloodPressure::GetDiastolicPressure()

{

return this->m\_iDiastolic;

}

//Function SetDiastolicPressure() that sets value of diastolic pressure.

//Has one parameter - integer value of disstiolc pressure.

void CBloodPressure::SetDiastolicPressure(int iDiastolic)

{

if (iDiastolic >0 && iDiastolic < MAX\_DIASTOLIC)

this->m\_iDiastolic = iDiastolic;

else

throw Cexception("Range");

}

//Function CheckPressure() checks if current pressure is normal.

/\*Return values:

-1 -- low pressure - systolic pressure is <90 or diastolis pressure <60

0 -- normal pressure

1 -- high pressure - systolic pressure is >140 or diastolis pressure >90\*/

int CBloodPressure::CheckPressure()

{

int iDiast = this->GetDiastolicPressure();

int iSyst = this->GetSystolicPressure();

if (iDiast < MIN\_NORM\_D || iSyst < MIN\_NORM\_S)

return -1;

else if (iDiast > MAX\_NORM\_D || iSyst > MAX\_NORM\_S)

return 1;

else

return 0;

}

//Overloaded operator << to output class info to output stream.

//Has two parameters - reference to output stream and class object.

//Returns reference to output stream.

std::ostream& operator<<(std::ostream& osOut, CBloodPressure& bpSource)

{

osOut << bpSource.m\_iSystolic << "/" << bpSource.m\_iDiastolic;

return osOut;

}

//Overloaded operator >> to input class info from input stream.

//Has two parameters - reference to input stream and class object.

//Returns reference to input stream.

std::istream& operator>>(std::istream& isIn, CBloodPressure& bpDestination)

{

isIn >> bpDestination.m\_iSystolic;

isIn >> bpDestination.m\_iDiastolic;

return isIn;

}

**CpatientInfo.h**

#ifndef CpatientInfoH

#define CpatientInfoH

#include "CBloodPressure.h"

//Class CpatientInfo contains all the information about patient.

/\*Fields of this class contain data about:

Surname of patient;

Age of patient;

Blood Type of patient and its rhesus factor;

Arterial pressure;

Pulse - beats per minute.

\*/

class CpatientInfo

{

private:

std::string m\_sSurname;

int m\_iAge;

BloodType m\_iBloodType;

Rhesus m\_bResus;

CBloodPressure m\_Pressure;

int m\_iPulse;

public:

CpatientInfo();

CpatientInfo(std::string sSurname, int iAge, BloodType btBloodType,

Rhesus rhResus, CBloodPressure bpPressure, int iPulse);

CpatientInfo(const CpatientInfo& piSource);

CpatientInfo& operator=(const CpatientInfo &piSource);

friend std::ostream& operator<<(std::ostream& osOut, CpatientInfo& piSource);

friend std::istream& operator>>(std::istream& isIn, CpatientInfo& piDestination);

bool operator>(CpatientInfo& piVal);

bool operator<(CpatientInfo& piVal);

CBloodPressure GetPressure();

std::string GetSurname();

int GetAge();

BloodType GetBloodtype();

Rhesus GetResus();

int GetPulse();

void SetPressure(CBloodPressure bpPressure);

void SetSurname(std::string sSurname);

void SetAge(int iAge);

void SetBloodtype(BloodType btBloodType);

void SetResus(Rhesus rhResus);

void SetPulse(int iPulse);

std::string GetStrBloodType();

std::string GetStrRhesus();

int CheckPulse();

};

#endif

**CpatientInfo.cpp**

#include "CpatientInfo.h"

#define MAX\_AGE 100

#define MAX\_LEN 50

#define MIN\_PULSE 30

#define MAX\_PULSE 300

#define MAX\_NORM\_PULSE 100

#define MIN\_NORM\_PULSE 50

//Default constructor without any parameters.

CPatientInfo::CPatientInfo()

{

this->SetSurname(" ");

this->SetAge(1);

this->SetBloodtype(O);

this->SetResus(Negative);

this->m\_Pressure.SetDiastolicPressure(1);

this->m\_Pressure.SetSystolicPressure(1);

this->SetPulse(MIN\_NORM\_PULSE);

}

//Constructor with six parameters.

//Assign those parameters to class fields.

CPatientInfo::CPatientInfo(std::string sSurname, int iAge, BloodType btBloodType, Rhesus rhResus, CBloodPressure bpPressure, int iPulse)

{

this->SetSurname(sSurname);

this->SetAge(iAge);

this->SetBloodtype(btBloodType);

this->SetResus(rhResus);

this->SetPressure(bpPressure);

this->SetPulse(iPulse);

}

//Copy constructor.

//Has one parameter - constant reference to the source object.

CPatientInfo::CPatientInfo(const CPatientInfo& source)

:m\_sSurname(source.m\_sSurname), m\_iAge(source.m\_iAge), m\_iBloodType(source.m\_iBloodType), m\_bResus(source.m\_bResus), m\_iPulse(source.m\_iPulse), m\_Pressure(source.m\_Pressure)

{

}

//Overloaded operator = which assigns all the values from source object to current object.

//Has one parameter - constant reference to the source object.

CPatientInfo& CPatientInfo::operator=(const CPatientInfo &piSource)

{

m\_sSurname = piSource.m\_sSurname;

m\_iAge = piSource.m\_iAge;

m\_iBloodType = piSource.m\_iBloodType;

m\_bResus = piSource.m\_bResus;

m\_Pressure = piSource.m\_Pressure;

m\_iPulse = piSource.m\_iPulse;

return \*this;

}

//Function GetSurname() to get string value of blood pressure.

//Returns object of blood pressure.

CBloodPressure CPatientInfo::GetPressure()

{

return this->m\_Pressure;

}

//Function GetSurname() to get string value of surname.

//Returns string value of surname.

std::string CPatientInfo::GetSurname()

{

return this->m\_sSurname;

}

//Function GetAge() to get value of age.

//Returns integer value of age.

int CPatientInfo::GetAge()

{

return this->m\_iAge;

}

//Function GetBloodtype() to get value of blood type.

//Returns enum value of blood type.

BloodType CPatientInfo::GetBloodtype()

{

return m\_iBloodType;

}

//Function GetResus() to get value of rhesus.

//Returns enum value of rhesus.

Rhesus CPatientInfo::GetResus()

{

return this->m\_bResus;

}

//Function GetAge() to get value of pulse.

//Returns integer value of pulse.

int CPatientInfo::GetPulse()

{

return this->m\_iPulse;

}

//Function SetPressure() that sets value of pressure.

//Has one parameter - object of blood pressure.

void CPatientInfo::SetPressure(CBloodPressure bpPressure)

{

this->m\_Pressure = bpPressure;

}

//Function SetSurname() that sets string value of surname.

//Has one parameter - string value of surname.

void CPatientInfo::SetSurname(std::string sSurname)

{

if (sSurname.length()> MAX\_LEN||sSurname.length()< 1)

{

throw Cexception("Strlen");

return;

}

std::string::const\_iterator it = sSurname.begin();

std::locale loc2("en\_US.UTF8");

while (it != sSurname.end() && !std::isdigit(\*it)&&!std::ispunct(\*it))

++it;

if (it == sSurname.end())

{

this->m\_sSurname = sSurname;

}

else

throw Cexception("Alphanum");

}

//Function SetAge() that sets value of age.

//Has one parameter - integer value of age.

void CPatientInfo::SetAge(int iAge)

{

if (iAge > 0 && iAge < MAX\_AGE)

{

this->m\_iAge = iAge;

}

else

throw Cexception("Range");

}

//Function SetBloodtype() that sets value of blood type.

//Has one parameter - enum value of blood type.

void CPatientInfo::SetBloodtype(BloodType btBloodType)

{

this->m\_iBloodType = btBloodType;

}

//Function SetResus() that sets value of rhesus.

//Has one parameter - enum value of rhesus.

void CPatientInfo::SetResus(Rhesus rhResus)

{

this->m\_bResus = rhResus;

}

//Function SetPulse() that sets value of pulse.

//Has one parameter - integer value of pulse.

void CPatientInfo::SetPulse(int iPulse)

{

if (iPulse > MIN\_PULSE && iPulse < MAX\_PULSE)

{

this->m\_iPulse = iPulse;

}

else

throw Cexception("Range");

}

//GetStrBloodType() is function to convert enum value of blood type.

//Returns string value.

std::string CPatientInfo::GetStrBloodType()

{

std::string sRes;

if(this->m\_iBloodType==O)

{

sRes = "O";

}

else if (this->m\_iBloodType== A )

{

sRes = "A";

}

else if(this->m\_iBloodType== B)

{

sRes = "B";

}

else if(this->m\_iBloodType== AB)

{

sRes = "AB";

}

return sRes;

}

//GetStrRhesus() is function to convert enum value of rhesus.

//Returns string value.

std::string CPatientInfo::GetStrRhesus()

{

std::string sRes;

if(this->m\_bResus==Negative)

{

sRes = "-";

}

else if (this->m\_bResus== Positive )

{

sRes = "+";

}

return sRes;

}

//Overloaded operator > compares two objects by pressure value.

//Has one parameter - object of class.

/\*If pressure of current patient is bigger than pressure of

parameter function returns 1;

If pressure of current patient is less or equal than pressure of

parameter function returns 0.

\*/

bool CPatientInfo:: operator>(CPatientInfo& piVal)

{

if (this->m\_Pressure.GetSystolicPressure() > piVal.m\_Pressure.GetSystolicPressure())

return 1;

else if (this->m\_Pressure.GetSystolicPressure() < piVal.m\_Pressure.GetSystolicPressure())

return 0;

else if (this->m\_Pressure.GetSystolicPressure() == piVal.m\_Pressure.GetSystolicPressure())

{

if (this->m\_Pressure.GetDiastolicPressure() > piVal.m\_Pressure.GetDiastolicPressure())

return 1;

else

return 0;

}

return 0;

}

//Overloaded operator < compares two objects by pressure value.

//Has one parameter - object of class.

/\*If pressure of current patient is less than pressure of

parameter function returns 1;

If pressure of current patient is bigger or equal than pressure of

parameter function returns 0.

\*/

bool CPatientInfo:: operator<(CPatientInfo& piVal)

{

if (this->m\_Pressure.GetSystolicPressure() < piVal.m\_Pressure.GetSystolicPressure())

return 1;

else if (this->m\_Pressure.GetSystolicPressure() > piVal.m\_Pressure.GetSystolicPressure())

return 0;

else if (this->m\_Pressure.GetSystolicPressure() == piVal.m\_Pressure.GetSystolicPressure())

{

if (this->m\_Pressure.GetDiastolicPressure() < piVal.m\_Pressure.GetDiastolicPressure())

return 1;

else

return 0;

}

return 0;

}

//Function CheckPulse() checks if current patient has normal pulse.

/\*Return values:

-1 -- low pulse - pulse is <50

0 -- normal pressure

1 -- high pressure - pulse is >100\*/

int CPatientInfo::CheckPulse()

{

if (this->GetPulse() < MIN\_NORM\_PULSE)

return -1;

else if(this->GetPulse()> MAX\_NORM\_PULSE)

return 1;

else

return 0;

}

//Overloaded operator << to output class info to output stream.

//Has two parameters - reference to output stream and class object.

//Returns reference to output stream.

std::ostream& operator<<(std::ostream& osOut, CPatientInfo& piSource)

{

osOut << std::setw(15) << piSource.m\_sSurname << "||" << std::setw(5) << piSource.m\_iAge << "||";

std::string temp;

if (piSource.m\_iBloodType == O)

{

temp = "O";

}

else if (piSource.m\_iBloodType == A)

{

temp = "A";

}

else if (piSource.m\_iBloodType == B)

{

temp = "B";

}

else if (piSource.m\_iBloodType == AB)

{

temp = "AB";

}

osOut << std::setw(5)<< temp << "||";

if (piSource.m\_bResus == Negative)

{

temp = "-";

}

else if (piSource.m\_bResus == Positive)

{

temp = "+";

}

osOut << std::setw(3) << temp << "||" << std::setw(8) << piSource.m\_Pressure << "||" << std::setw(5) << piSource.m\_iPulse << "\n";

return osOut;

}

//Overloaded operator >> to input class info from input stream.

//Has two parameters - reference to input stream and class object.

//Returns reference to input stream.

std::istream& operator>>(std::istream& isIn, CPatientInfo& piDestination)

{

int temp;

std::string temp1;

isIn >> piDestination.m\_sSurname;

isIn >> piDestination.m\_iAge;

isIn >> temp;

switch (temp)

{

case 1:

piDestination.m\_iBloodType = O;

break;

case 2:

piDestination.m\_iBloodType = A;

break;

case 3:

piDestination.m\_iBloodType = B;

break;

case 4:

piDestination.m\_iBloodType = AB;

break;

default:

break;

}

isIn >> temp1;

if (temp1 == "+")

piDestination.m\_bResus = Positive;

else if (temp1 == "-")

piDestination.m\_bResus = Negative;

isIn >> piDestination.m\_Pressure;

isIn >> piDestination.m\_iPulse;

return isIn;

}

**CpatientArr.h**

#ifndef CpatientArrH

#define CpatientArrH

#include "CpatientInfo.h"

void QuickSort(std::vector<CPatientInfo> &vecPatientArr, int iLeft, int iRight, bool (\*pCmpFunc)(CPatientInfo, CPatientInfo));

bool CmpPulseAsc(CPatientInfo a, CPatientInfo b);

bool CmpPressureAsc(CPatientInfo a, CPatientInfo b);

bool CmpPressureDesc(CPatientInfo a, CPatientInfo b);

//Class CPatientArr is created to help user with management of your patients.

//Contains vector of CPatientInfo objects.

class CPatientArr

{

private:

std::vector<CPatientInfo> m\_vecArray;

public:

CPatientArr();

CPatientArr(int iCount);

CPatientArr(const std::vector<CPatientInfo>& vecPatientArr);

CPatientArr(const CPatientArr& paPatientArr);

friend std::ostream& operator<<(std::ostream& osOut, CPatientArr& paSource);

friend std::istream& operator>>(std::istream& isIn, CPatientArr& paDestination);

CPatientInfo& operator[](int iIndex);

std::vector<CPatientInfo> GetVector();

void SetVector(std::vector<CPatientInfo> vecPatientArr);

void PushBack(CPatientInfo piNewElem);

void EraseInd(int iIndex);

std::vector<CPatientInfo> FindSameBloodType(BloodType btBloodType, Rhesus rhRhesus);

std::vector<CPatientInfo> FindSameRhesusSort(Rhesus rhRhesus, bool (\*pCmpFunc)(CPatientInfo, CPatientInfo));

std::vector<CPatientInfo> FindUniDonors();

std::vector<CPatientInfo> FindUniRecipients();

std::vector<CPatientInfo> FindHighPressureAndPulse(int iAge);

bool AreCompartible(int iDonor, int iRecipient);

friend void QuickSort(CPatientArr &vecPatientArr, int iLeft, int iRight, bool (\*pCmpFunc)(CPatientInfo, CPatientInfo));

};

#endif

**CpatientArr.cpp**

#include "CpatientArr.h"

#define MAX\_AGE 100

//Function CmpPulseAsc() compares two CPatientInfo objects by pulse.

//Has two parameters - two objects of CPatientInfo class.

/\* Return values:

if pulse of the first patient is lower than second -> return 1.

else return 0.

\*/

bool CmpPulseAsc(CPatientInfo piA, CPatientInfo piB)

{

if (piA.GetPulse() < piB.GetPulse())

return 1;

else

return 0;

}

//Function CmpPulseAsc() compares two CPatientInfo objects by pressure.

//Has two parameters - two objects of CPatientInfo class.

/\* Return values:

if pressure of the first patient is greater than second -> return 1.

else return 0.

\*/

bool CmpPressureAsc(CPatientInfo piA, CPatientInfo piB)

{

if (piA > piB)

return 1;

else

return 0;

}

//Function CmpPulseAsc() compares two CPatientInfo objects by pressure.

//Has two parameters - two objects of CPatientInfo class.

/\* Return values:

if pressure of the first patient is lower than second -> return 1.

else return 0.

\*/

bool CmpPressureDesc(CPatientInfo piA, CPatientInfo piB)

{

if (piA < piB)

return 1;

else

return 0;

}

//Function QuickSort() is recursive function to sort array.

/\*Has four parameters: array for sorting, the leftest element and the rightest

element, comparison function which returns bool and has

two parameters - two CPatientInfo objects.\*/

void QuickSort( CPatientArr& paPatientArr, int iLeft, int iRight, bool (\*pCmpFunc)(CPatientInfo, CPatientInfo)) {

if (iLeft >= iRight)

return;

CPatientInfo pivot = paPatientArr[iLeft];

int count = 0;

for (int i = iLeft + 1; i <= iRight; i++) {

if (!pCmpFunc(paPatientArr[i],pivot))

count++;

}

int pivotIndex = iLeft + count;

std::swap(paPatientArr[pivotIndex], paPatientArr[iLeft]);

int i = iLeft, j = iRight;

while (i < pivotIndex && j > pivotIndex) {

while (!pCmpFunc(paPatientArr[i],pivot)) {

i++;

}

while (pCmpFunc(paPatientArr[j],pivot)) {

j--;

}

if (i < pivotIndex && j > pivotIndex) {

std::swap(paPatientArr[i++], paPatientArr[j--]);

}

}

// partitioning the array

int p = pivotIndex;

// Sorting the left part

QuickSort(paPatientArr, iLeft, p - 1, pCmpFunc);

// Sorting the right part

QuickSort(paPatientArr, p + 1, iRight, pCmpFunc);

}

//Default constructor without any parameters.

CPatientArr::CPatientArr()

{

this->m\_vecArray.clear();

}

//Constructor with one parameter - integer for size of array.

//Create vector with passed size.

CPatientArr::CPatientArr(int iCount)

{

if (iCount>0){

this->m\_vecArray.resize(iCount);

for (int i = 0; i < m\_vecArray.size(); ++i)

{

CPatientInfo temp;

this->m\_vecArray[i] = temp;

}

}

else throw Cexception("Range");

}

//Copy constructor.

//Has one parameter - constant reference to the source object.

CPatientArr::CPatientArr(const std::vector<CPatientInfo>& vecSourceArr)

{

this->m\_vecArray.clear();

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

{

m\_vecArray.push\_back(vecSourceArr[i]);

}

}

//Constructor with one parameter - source vector.

//Create vector with the same length and values as source vector.

CPatientArr::CPatientArr(const CPatientArr& vecSourceArr)

{

this->m\_vecArray.clear();

for (int i = 0; i < vecSourceArr.m\_vecArray.size(); ++i)

{

this->m\_vecArray.push\_back(vecSourceArr.m\_vecArray[i]);

}

}

//Function GetVector() to get vector from this class.

//Returns vector of CPatientInfo objects.

std::vector<CPatientInfo> CPatientArr::GetVector()

{

return this->m\_vecArray;

}

//Function SetVector() to set source vector.

//Has one parameter - source vector.

//Makes current vector hsve the same length and values as source vector.

void CPatientArr::SetVector(std::vector<CPatientInfo> vecSourceArr)

{

if (!vecSourceArr.empty())

{

this->m\_vecArray.resize(vecSourceArr.size());

for (int i = 0; i < m\_vecArray.size(); ++i)

{

this->m\_vecArray.push\_back(vecSourceArr[i]);

}

} else throw Cexception("Empty");

}

//Function PushBack() adds new element to the end of current vector.

//Has one parameter - CPatientInfo object.

void CPatientArr::PushBack(CPatientInfo piNewElem)

{

this->m\_vecArray.push\_back(piNewElem);

}

void CPatientArr::EraseInd(int iIndex)

{

if (iIndex>=0&& iIndex < m\_vecArray.size())

this->m\_vecArray.erase(this->m\_vecArray.begin()+iIndex);

else throw Cexception("Index");

}

//Overloaded operator [] to get element of current vector by index.

//Has one parameter - integer index.

//Returns reference to CPatientInfo object.

CPatientInfo & CPatientArr::operator[](int iIndex)

{

CPatientInfo piRes;

if (iIndex>(-1)&&iIndex<m\_vecArray.size())

return this->m\_vecArray[iIndex];

else throw Cexception("Index");

}

//Function FindSameBloodType() finds patients with given by user blood type and rhesus.

//Has two parameters - blood type and rhesus.

//Returns vector of patients with passed blood type and rhesus.

std::vector<CPatientInfo> CPatientArr::FindSameBloodType(BloodType btBloodType, Rhesus rhRhesus)

{

if(!this->m\_vecArray.empty()){

std::vector<CPatientInfo> piRes;

for (int i = 0; i < m\_vecArray.size(); ++i)

{

if (this->m\_vecArray[i].GetBloodtype() == btBloodType && this->m\_vecArray[i].GetResus() == rhRhesus)

piRes.push\_back(this->m\_vecArray[i]);

}

return piRes;

}else throw Cexception("Empty");

}

//Function FindSameRhesusSort() finds patients with given by user rhesus.

/\*Has two parameters - rhesus and comparison function which returns boolean

value and accepts two CPatientInfo objects.\*/

//Returns sorted vector of CPatientInfo objects.

std::vector<CPatientInfo> CPatientArr::FindSameRhesusSort(Rhesus rhRhesus, bool (\*pCmpFunc)(CPatientInfo, CPatientInfo))

{

if(!this->m\_vecArray.empty()){

std::vector<CPatientInfo> piRes;

for (int i = 0; i < m\_vecArray.size(); ++i)

{

if (this->m\_vecArray[i].GetResus() == rhRhesus)

piRes.push\_back(this->m\_vecArray[i]);

}

std::sort(piRes.begin(), piRes.end(), pCmpFunc);

return piRes;

}else throw Cexception("Empty");

}

//Function FindUniDonors() finds universal donors (people with O- blood type) in currrent vector.

//Returns vector of CPatientInfo objects.

std::vector<CPatientInfo> CPatientArr::FindUniDonors()

{

if(!this->m\_vecArray.empty()){

std::vector<CPatientInfo> piRes;

for (int i = 0; i < m\_vecArray.size(); ++i)

{

if (this->m\_vecArray[i].GetResus() == Negative && this->m\_vecArray[i].GetBloodtype() == O)

piRes.push\_back(this->m\_vecArray[i]);

}

return piRes;

}else throw Cexception("Empty");

}

//Function FindUniRecipients() finds universal recipients (people with AB+ blood type) in currrent vector.

//Returns vector of CPatientInfo objects.

std::vector<CPatientInfo> CPatientArr::FindUniRecipients()

{ if(!this->m\_vecArray.empty()){

std::vector<CPatientInfo> piRes;

for (int i = 0; i < m\_vecArray.size(); ++i)

{

if (this->m\_vecArray[i].GetResus() == Positive && this->m\_vecArray[i].GetBloodtype() == AB)

piRes.push\_back(this->m\_vecArray[i]);

}

return piRes;

}else throw Cexception("Empty");

}

//Function FindHighPressureAndPulse() finds patient with hight pressure AND pulse by age.

//Has one parameter - integer value of age.

//Returns vector of CPatientInfo objects.

std::vector<CPatientInfo> CPatientArr::FindHighPressureAndPulse(int iAge)

{

if(iAge>=0&&iAge<MAX\_AGE)

{

if(!this->m\_vecArray.empty()){

std::vector<CPatientInfo> piRes;

for (int i = 0; i < m\_vecArray.size(); ++i)

{

if (this->m\_vecArray[i].GetAge() == iAge && this->m\_vecArray[i].GetPressure().CheckPressure() > 0

&& this->m\_vecArray[i].CheckPulse() > 0)

piRes.push\_back(this->m\_vecArray[i]);

}

return piRes;

}else throw Cexception("Empty");

}else throw Cexception("Range");

}

//Function AreCompartible checks if the first given patient can be donor to the second.

//Has two parameters - integer index of donor and of reipient.

/\*If the first given patient can be donor to the second -> return 1.

Else -> return 0. \*/

bool CPatientArr::AreCompartible(int iDonor, int iRecipient)

{

if(iDonor>=0&&iDonor<this->m\_vecArray.size()&&iRecipient>=0&&iRecipient<this->m\_vecArray.size())

{

if( this->m\_vecArray[iDonor].GetResus()== Positive&& this->m\_vecArray[iRecipient].GetResus() == Negative)

return 0;

else

{

if(this->m\_vecArray[iDonor].GetBloodtype() == O)

return 1;

else if(this->m\_vecArray[iRecipient].GetBloodtype() == AB)

return 1;

else if(this->m\_vecArray[iDonor].GetBloodtype()==this->m\_vecArray[iRecipient].GetBloodtype())

return 1;

}

}else throw Cexception("Index");

return 0;

}

//Overloaded operator << to output class info to output stream.

//Has two parameters - reference to output stream and class object.

//Returns reference to output stream.

std::ostream& operator<<(std::ostream& osOut, CPatientArr& paSource)

{

for (int i = 0; i<paSource.m\_vecArray.size();++i)

{

osOut<<paSource[i]<<std::endl;

}

return osOut;

}

//Overloaded operator >> to input class info from input stream.

//Has two parameters - reference to input stream and class object.

//Returns reference to input stream.

std::istream& operator>>(std::istream& isIn, CPatientArr& paDestination)

{

if(isIn.peek() == EOF)

{

throw Cexception("EmptyFile");

}else{

CPatientInfo temp;

isIn >> temp;

paDestination.PushBack(temp);

while(!isIn.eof())

{

isIn>>temp;

paDestination.PushBack(temp);

}

return isIn;

}

}

**Cexception.h**

#ifndef CexceptionH

#define CexceptionH

#include <iostream>

//Class CException handles errors and warnings.

class CException: public std::exception

{

public:

CException(const char\* cErr) :std::exception(cErr) {}

virtual const char\* what() const noexcept override; };

#endif

**Cexception.cpp**

//---------------------------------------------------------

#pragma hdrstop

#include "Cexception.h"

//---------------------------------------------------------

#pragma package(smart\_init)

//Function what() returns string with explanation of thrown exception.

const char\* CException::what() const noexcept

{

if (!strcmp(this->std::exception::what(), "Alphanum"))

{

return "Surname contains invalid characters. Please, check input data.";

}

else if (!strcmp(this->std::exception::what(), "Strlen"))

{

return "Surname has wrong lenght. Please, check input data. Surname can be between 1 and 50 symbols.";

}

else if (!strcmp(this->std::exception::what(), "Range"))

{

return "One of your integer inputs is out of range. Please, check input data.";

}

else if (!strcmp(this->std::exception::what(), "Index"))

{

return "Action is accessing data using invalid index.";

}

else if (!strcmp(this->std::exception::what(), "EmptyFile"))

{

return "Chosen file is empty. Check input file.";

}

else if (!strcmp(this->std::exception::what(), "Empty"))

{

return "Your vector is empty. Add items to table and try again.";

}

else if (!strcmp(this->std::exception::what(), "Open"))

{

return "Problem with opening file for obtaining data has occurred. Please, check if your file exists and isn`t corrupted.";

}

else if (!strcmp(this->std::exception::what(), "Save"))

{

return "Problem with opening file for saving data has occurred. Please, check if your file exists and isn`t corrupted.";

}

else if (!strcmp(this->std::exception::what(), "Empty Field"))

{

return "One of your fields is empty. All the fields have to be filled.";

}

return this->std::exception::what();

}

**MainFormFunctions.h**

//---------------------------------------------------------

#ifndef MainFunctionsH

#define MainFunctionsH

#include "MainForm.h"

#include "AddNew.h"

void FirstLineFill();

void TableOfDonorsSettings();

void TableOfPatientsSettings();

void PrintArray(CPatientArr& paArray);

//---------------------------------------------------------

#endif

**MainFormFunctions.cpp**

//---------------------------------------------------------------------------

#pragma hdrstop

#include "MainFormFunctions.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

void FirstLineFill(){

MainFormCW->TableOfPatients->Cells[0][0] = "#";

MainFormCW->TableOfPatients->Cells[1][0] = "Surname";

MainFormCW->TableOfPatients->Cells[2][0] = "Age";

MainFormCW->TableOfPatients->Cells[3][0] = "Blood Type";

MainFormCW->TableOfPatients->Cells[4][0] = "Rhesus";

MainFormCW->TableOfPatients->Cells[5][0] = "Pressure";

MainFormCW->TableOfPatients->Cells[6][0] = "Pulse";

}

void TableOfDonorsSettings()

{

MainFormCW->Shape1->Brush->Color = (TColor)0x00EAC19D;

MainFormCW->Shape2->Brush->Color = (TColor)0x00EAC19D;

MainFormCW->Shape3->Brush->Color = (TColor)0x00EAC19D;

MainFormCW->Shape1->Pen->Color = (TColor)0x00EAC19D;

MainFormCW->Shape2->Pen->Color = (TColor)0x00EAC19D;

MainFormCW->Shape3->Pen->Color = (TColor)0x00EAC19D;

MainFormCW->Panel1->Visible = false;

MainFormCW->TableOfPatients->Visible = false;

MainFormCW->TableOfDonors->Visible = true;

MainFormCW->ToolBar1->Visible = false;

MainFormCW->Image2->Visible = true;

MainFormCW->Image3->Visible = true;

MainFormCW->Label1->Caption = "Table of Donors and Recipients";

MainFormCW->TableOfDonors-> RowCount = paArr.GetVector().size()+1;

MainFormCW->TableOfDonors-> ColCount = paArr.GetVector().size()+1;

CPatientInfo temp;

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

MainFormCW->TableOfDonors->Cells[0][i+1] = paArr[i].GetSurname().c\_str();

MainFormCW->TableOfDonors->Cells[0][i+1] += " ";

MainFormCW->TableOfDonors->Cells[0][i+1] += paArr[i].GetStrBloodType().c\_str();

MainFormCW->TableOfDonors->Cells[0][i+1] += paArr[i].GetStrRhesus().c\_str();

}

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

MainFormCW->TableOfDonors->Cells[i+1][0] = paArr[i].GetSurname().c\_str();

MainFormCW->TableOfDonors->Cells[i+1][0] += " ";

MainFormCW->TableOfDonors->Cells[i+1][0] += paArr[i].GetStrBloodType().c\_str();

MainFormCW->TableOfDonors->Cells[i+1][0] += paArr[i].GetStrRhesus().c\_str();

}

}

void TableOfPatientsSettings()

{

MainFormCW->Shape1->Brush->Color = (TColor)0x00FBE4D7;

MainFormCW->Shape2->Brush->Color = (TColor)0x00FBE4D7;

MainFormCW->Shape3->Brush->Color = (TColor)0x00FBE4D7;

MainFormCW->Shape1->Pen->Color = (TColor)0x00FBE4D7;

MainFormCW->Shape2->Pen->Color = (TColor)0x00FBE4D7;

MainFormCW->Shape3->Pen->Color = (TColor)0x00FBE4D7;

MainFormCW->Panel1->Visible = true;

MainFormCW->TableOfPatients->Visible = true;

MainFormCW->TableOfDonors->Visible = false;

MainFormCW->ToolBar1->Visible = true;

MainFormCW->Image2->Visible = false;

MainFormCW->Image3->Visible = false;

MainFormCW->Label1->Caption = "Table of Patients";

MainFormCW->TableOfPatients ->RowCount = paArr.GetVector().size()+1;

}

void PrintArray(CPatientArr& paArray)

{

for (int i = 0; i < paArray.GetVector().size();++i)

{

MainFormCW->TableOfPatients->Cells[0][i+1] = i+1;

MainFormCW->TableOfPatients->Cells[1][i+1] = paArray[i].GetSurname().c\_str();

MainFormCW->TableOfPatients->Cells[2][i+1] = paArray[i].GetAge();

MainFormCW->TableOfPatients->Cells[3][i+1] = paArray[i].GetStrBloodType().c\_str();

MainFormCW->TableOfPatients->Cells[4][i+1] = paArray[i].GetStrRhesus().c\_str();

int iSyst = paArray[i].GetPressure().GetSystolicPressure();

int iDiast = paArray[i].GetPressure().GetDiastolicPressure();

MainFormCW->TableOfPatients->Cells[5][i+1] = IntToStr(iSyst)+"/"+IntToStr(iDiast);

MainFormCW->TableOfPatients->Cells[6][i+1] = paArray[i].GetPulse();

}

}

**MainForm.cpp**

//---------------------------------------------------------

#include <vcl.h>

#pragma hdrstop

#include "MainFunctions.h"

//---------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

#include "Help.h"

TMainFormCW \*MainFormCW;

bool ischanged = 0;

bool issaved = 0;

//---------------------------------------------------------

\_\_fastcall TMainFormCW::TMainFormCW(TComponent\* Owner)

: TForm(Owner)

{

FirstLineFill();

}

//---------------------------------------------------------

void \_\_fastcall TMainFormCW::DonRecTableTabClick(TObject \*Sender)

{

TableOfDonorsSettings();

for(int i = 0; i<paArr.GetVector().size();++i)

{

for(int j = 0; j<paArr.GetVector().size();++j)

{

if (i==j)

{

TableOfDonors->Cells[i+1][j+1] = "Same Person";

}

else if( paArr.AreCompartible(j,i))

{

TableOfDonors->Cells[j+1][i+1] = "+";

}

else

TableOfDonors->Cells[j+1][i+1] = "-";

}

}

}

//---------------------------------------------------------

void \_\_fastcall TMainFormCW::PatientsTableTabClick(TObject \*Sender)

{

TableOfPatientsSettings();

PrintArray(paArr);

}

//---------------------------------------------------------

void \_\_fastcall TMainFormCW::AddNewClick(TObject \*Sender)

{

AddNewPatientForm->ShowModal();

TableOfPatients ->RowCount = paArr.GetVector().size()+1;

PrintArray(paArr);

issaved = 0;

}

//---------------------------------------------------------

void \_\_fastcall TMainFormCW::SortPressureAscClick(TObject \*Sender)

{

try{

if (!paArr.GetVector().empty()){

QuickSort(paArr, 0, paArr.GetVector().size()-1, &CmpPressureAsc);

TableOfPatients ->RowCount = paArr.GetVector().size()+1;

PrintArray(paArr);

}else throw CException("Empty");

}catch(std::exception & ex)

{

Application->MessageBox(UnicodeString(ex.what()).c\_str(), \_D("Error"), MB\_ICONERROR|MB\_OK);

}

catch(...)

{

Application->MessageBox(UnicodeString("Unknown Error!").c\_str(), \_D("Error"), MB\_ICONERROR|MB\_OK);

}

issaved = 0;

}

//---------------------------------------------------------

void \_\_fastcall TMainFormCW::SortPressureDescClick(TObject \*Sender)

{

try{

if (!paArr.GetVector().empty()){

QuickSort(paArr, 0, paArr.GetVector().size()-1, &CmpPressureDesc);

TableOfPatients ->RowCount = paArr.GetVector().size()+1;

PrintArray(paArr);

}else throw CException("Empty");

}catch(std::exception & ex)

{

Application->MessageBox(UnicodeString(ex.what()).c\_str(), \_D("Error"), MB\_ICONERROR|MB\_OK);

}

catch(...)

{

Application->MessageBox(UnicodeString("Unknown Error!").c\_str(), \_D("Error"), MB\_ICONERROR|MB\_OK);

}

issaved= 0;

}

//---------------------------------------------------------

void \_\_fastcall TMainFormCW::GroupClick(TObject \*Sender)

{

BloodType bloodtype;

Rhesus rhesus;

bool allTypes = 0;

bool allRhesuses = 0;

if(RadioGroup1->ItemIndex == 0)

bloodtype = O;

else if(RadioGroup1->ItemIndex == 1)

bloodtype = A;

else if(RadioGroup1->ItemIndex == 2)

bloodtype = B;

else if(RadioGroup1->ItemIndex == 3)

bloodtype = AB;

else

{

allTypes =1;

}

if(RadioGroup2->ItemIndex == 0)

rhesus = Positive;

else if(RadioGroup2->ItemIndex == 1)

rhesus = Negative;

else

{

allRhesuses = 1;

}

std::vector<CPatientInfo> res;

std::vector<CPatientInfo> temp;

try{

if (allTypes)

{

if (allRhesuses)

{

for(int i = 1; i<=4; i++)

{

temp = paArr.FindSameBloodType((BloodType)i, Negative);

res.insert(res.end(),temp.begin(), temp.end());

temp = paArr.FindSameBloodType((BloodType)i, Positive);

res.insert(res.end(),temp.begin(), temp.end());

}

}

else

{

for(int i = 1; i<=4; i++)

{

temp = paArr.FindSameBloodType((BloodType)i, rhesus); res.insert(res.end(),temp.begin(), temp.end());

}

}

}

else if (allRhesuses)

{

temp = paArr.FindSameBloodType(bloodtype, Negative);

res.insert(res.end(),temp.begin(), temp.end());

temp = paArr.FindSameBloodType(bloodtype, Positive);

res.insert(res.end(),temp.begin(), temp.end());

}

else

{

res = paArr.FindSameBloodType(bloodtype, rhesus);

}

TableOfPatients->RowCount = res.size()+1;

CPatientArr resultArr(res);

PrintArray(resultArr);

}catch(std::exception & ex)

{

Application->MessageBox(UnicodeString(ex.what()).c\_str(), \_D("Error"), MB\_ICONERROR|MB\_OK);

}

catch(...)

{

Application->MessageBox(UnicodeString("Unknown Error!").c\_str(), \_D("Error"), MB\_ICONERROR|MB\_OK);

}

ShowAll1->Visible = true;

}

//---------------------------------------------------------

void \_\_fastcall TMainFormCW::GroupAndSortClick(TObject \*Sender)

{

Rhesus rhesus;

bool allRhesuses = 0;

if(RadioGroup3->ItemIndex == 0)

rhesus = Positive;

else if(RadioGroup3->ItemIndex == 1)

rhesus = Negative;

else

{

allRhesuses = 1;

}

std::vector<CPatientInfo> res;

std::vector<CPatientInfo> temp;

try{

if (allRhesuses)

{

temp = paArr.FindSameRhesusSort(Negative, &CmpPulseAsc);

res.insert(res.end(),temp.begin(), temp.end());

temp = paArr.FindSameRhesusSort(Positive, &CmpPulseAsc);

res.insert(res.end(),temp.begin(), temp.end());

}

else

{

res = paArr.FindSameRhesusSort(rhesus, &CmpPulseAsc);

}

TableOfPatients->RowCount = res.size()+1;

CPatientArr resultArr(res);

PrintArray(resultArr);

}catch(std::exception & ex)

{

Application->MessageBox(UnicodeString(ex.what()).c\_str(), \_D("Error"), MB\_ICONERROR|MB\_OK);

}

catch(...)

{

Application->MessageBox(UnicodeString("Unknown Error!").c\_str(), \_D("Error"), MB\_ICONERROR|MB\_OK);

}

ShowAll1->Visible = true;

}

//---------------------------------------------------------

void \_\_fastcall TMainFormCW::ShowClick(TObject \*Sender)

{

std::vector<CPatientInfo> res;

try{

res = paArr.FindHighPressureAndPulse(StrToInt(Edit1->Text));

TableOfPatients->RowCount = res.size()+1;

CPatientArr resultArr (res);

PrintArray(resultArr);

}catch(std::exception & ex)

{

Application->MessageBox(UnicodeString(ex.what()).c\_str(), \_D("Error"), MB\_ICONERROR|MB\_OK);

}

catch(...)

{

Application->MessageBox(UnicodeString("Unknown Error!").c\_str(), \_D("Error"), MB\_ICONERROR|MB\_OK);

}

ShowAll1->Visible = true;

}

//---------------------------------------------------------

void \_\_fastcall TMainFormCW::DeleteClick(TObject \*Sender)

{

int index = TableOfPatients->Row;

try{

paArr.EraseInd(index-1);

}catch(std::exception & ex)

{

Application->MessageBox(UnicodeString(ex.what()).c\_str(), \_D("Error"), MB\_ICONERROR|MB\_OK);

}

catch(...)

{

Application->MessageBox(UnicodeString("Unknown Error!").c\_str(), \_D("Error"), MB\_ICONERROR|MB\_OK);

}

TableOfPatients ->RowCount = paArr.GetVector().size()+1;

PrintArray(paArr);

issaved = 0;

}

//---------------------------------------------------------

void \_\_fastcall TMainFormCW::UniDonorsClick(TObject \*Sender)

{

try{

std::vector<CPatientInfo> resvector;

resvector = paArr.FindUniDonors();

TableOfPatients->RowCount = resvector.size()+1;

CPatientArr res (resvector);

PrintArray(res);

}catch(std::exception & ex)

{

Application->MessageBox(UnicodeString(ex.what()).c\_str(), \_D("Error"), MB\_ICONERROR|MB\_OK);

}

catch(...)

{

Application->MessageBox(UnicodeString("Unknown Error!").c\_str(), \_D("Error"), MB\_ICONERROR|MB\_OK);

}

ShowAll1->Visible = true;

}

//---------------------------------------------------------

void \_\_fastcall TMainFormCW::UniRecepientsClick(TObject \*Sender)

{

std::vector<CPatientInfo> res;

try{

res = paArr.FindUniRecipients();

TableOfPatients->RowCount = res.size()+1;

CPatientArr resultArr (res);

PrintArray(resultArr);

}catch(std::exception & ex)

{

Application->MessageBox(UnicodeString(ex.what()).c\_str(), \_D("Error"), MB\_ICONERROR|MB\_OK);

}

catch(...)

{

Application->MessageBox(UnicodeString("Unknown Error!").c\_str(), \_D("Error"), MB\_ICONERROR|MB\_OK);

}

ShowAll1->Visible = true;

}

//---------------------------------------------------------

void \_\_fastcall TMainFormCW::Help1Click(TObject \*Sender)

{

HelpForm->Show();

}

std::string filenametosave;

bool fileopened = 0;

void \_\_fastcall TMainFormCW::OpenFIleClick(TObject \*Sender)

{

if (OpenTextFileDialog1->Execute())

{

try{

if (FileExists(OpenTextFileDialog1->FileName))

{

std::wstring my\_wstr = OpenTextFileDialog1->FileName.c\_str();

std::string filename = std::wstring\_convert<std::codecvt\_utf8\_utf16<System::Char>>{}.to\_bytes(my\_wstr);

std::fstream file;

file.open(filename);

if (file.is\_open())

{

file>>paArr;

issaved = 1;

}

else throw CException("Open");

file.close();

}

else throw CException("Open");

}catch(std::exception & ex)

{

Application->MessageBox(UnicodeString(ex.what()).c\_str(), \_D("Error"), MB\_ICONERROR|MB\_OK);

}

catch(...)

{

Application->MessageBox(UnicodeString("Unknown Error!").c\_str(), \_D("Error"), MB\_ICONERROR|MB\_OK);

}

}

TableOfPatients ->RowCount = paArr.GetVector().size()+1;

PrintArray(paArr);

issaved = 0;

}

//---------------------------------------------------------

void \_\_fastcall TMainFormCW::SaveClick(TObject \*Sender)

{ try{

if (fileopened)

{

std::fstream file;

file.open(filenametosave);

if (file.is\_open())

{

file<<paArr;

issaved = 1;

}

else throw CException("Save");

file.close();

}

else

{

if (SaveTextFileDialog1->Execute())

{

std::wstring my\_wstr = SaveTextFileDialog1->FileName.c\_str();

filenametosave = std::wstring\_convert<std::codecvt\_utf8\_utf16<System::Char>>{}.to\_bytes(my\_wstr);

std::fstream file;

file.open(filenametosave, std::ios::app);

if (file.is\_open())

{

file<<paArr;

ShowMessage("Saved!");

} else throw CException("Save");

file.close();

fileopened = 1;

issaved = 1;

}

}

}catch(std::exception & ex)

{

Application->MessageBox(UnicodeString(ex.what()).c\_str(), \_D("Error"), MB\_ICONERROR|MB\_OK);

}

catch(...)

{

Application->MessageBox(UnicodeString("Unknown Error!").c\_str(), \_D("Error"), MB\_ICONERROR|MB\_OK);

}

}

//---------------------------------------------------------

void \_\_fastcall TMainFormCW::Save3Click(TObject \*Sender)

{

try{

if (SaveTextFileDialog1->Execute())

{

std::wstring my\_wstr = SaveTextFileDialog1->FileName.c\_str();

filenametosave = std::wstring\_convert<std::codecvt\_utf8\_utf16<System::Char>>{}.to\_bytes(my\_wstr);

std::fstream file;

file.open(filenametosave, std::ios::app);

if (file.is\_open())

{

file<<paArr;

ShowMessage("Saved!");

} else throw ("Save");

file.close();

fileopened = 1;

}

}catch(std::exception & ex)

{

Application->MessageBox(UnicodeString(ex.what()).c\_str(), \_D("Error"), MB\_ICONERROR|MB\_OK);

}

catch(...)

{

Application->MessageBox(UnicodeString("Unknown Error!").c\_str(), \_D("Error"), MB\_ICONERROR|MB\_OK);

}

}

//---------------------------------------------------------

void \_\_fastcall TMainFormCW::ShowAll1Click(TObject \*Sender)

{

TableOfPatients ->RowCount = paArr.GetVector().size()+1;

PrintArray(paArr);

ShowAll1->Visible = false;

}

//---------------------------------------------------------void \_\_fastcall TMainFormCW::FormClose(TObject \*Sender, TCloseAction &Action)

{

if (!issaved)

{

if(Application->MessageBox(UnicodeString("You have unsaved changes. Do you want to save them").c\_str(), \_D("Attention"), MB\_ICONWARNING|MB\_YESNO) == IDNO)

{

return;

}

else

{

try{

if (fileopened)

{

std::fstream file;

file.open(filenametosave);

if (file.is\_open())

{

file<<paArr;

}

else throw CException("Save");

file.close();

}

else

{

if (SaveTextFileDialog1->Execute())

{

std::wstring my\_wstr = SaveTextFileDialog1->FileName.c\_str();

filenametosave = std::wstring\_convert<std::codecvt\_utf8\_utf16<System::Char>>{}.to\_bytes(my\_wstr);

std::fstream file;

file.open(filenametosave, std::ios::app);

if (file.is\_open())

{

file<<paArr;

ShowMessage("Saved!");

} else throw CException("Save");

file.close();

fileopened = 1;

}

}

}catch(std::exception & ex)

{

Application->MessageBox(UnicodeString(ex.what()).c\_str(), \_D("Error"), MB\_ICONERROR|MB\_OK);

}

catch(...)

{

Application->MessageBox(UnicodeString("Unknown Error!").c\_str(), \_D("Error"), MB\_ICONERROR|MB\_OK);

}

}

}

}

//---------------------------------------------------------

**AddNew.cpp**

//---------------------------------------------------------

#include <vcl.h>

#pragma hdrstop

#include "AddNew.h"

//---------------------------------------------------------

#pragma package(smart\_init)

#pragma link "cspin"

#pragma resource "\*.dfm"

#include <iostream>

#include <string>

#include "Help.h"

#define DEF\_AGE 18

TAddNewPatientForm \*AddNewPatientForm;

//---------------------------------------------------------

\_\_fastcall TAddNewPatientForm::TAddNewPatientForm(TComponent\* Owner)

: TForm(Owner)

{

}

//---------------------------------------------------------

CPatientArr paArr;

void \_\_fastcall TAddNewPatientForm::SaveAndCloseClick(TObject \*Sender)

{

try{

if( (SurnameEdit->Text == "")||(RadioGroupBloodType->ItemIndex == -1)||(RadioGroupRhesus->ItemIndex == (-1))

|| SystPressureEdit->Text == ""||DiastPressureEdit->Text==""||PulseEdit->Text =="")

throw CException("Empty Field");

else{

CPatientInfo piTemp;

std::wstring my\_wstr = SurnameEdit->Text.c\_str();

std::string my\_str = std::wstring\_convert<std::codecvt\_utf8\_utf16<System::Char>>{}.to\_bytes(my\_wstr);

std::string str = my\_str;

piTemp.SetSurname(str);

piTemp.SetAge(SpinEditAge->Value);

BloodType btTmpBloodType;

if(RadioGroupBloodType->ItemIndex == 0)

btTmpBloodType = O;

else if(RadioGroupBloodType->ItemIndex == 1)

btTmpBloodType = A;

else if(RadioGroupBloodType->ItemIndex == 2)

btTmpBloodType = B;

else if(RadioGroupBloodType->ItemIndex == 3)

btTmpBloodType = AB;

piTemp.SetBloodtype(btTmpBloodType);

Rhesus rhTmpRhesus;

if(RadioGroupRhesus->ItemIndex == 0)

rhTmpRhesus = Positive;

else if(RadioGroupRhesus->ItemIndex == 1)

rhTmpRhesus = Negative;

piTemp.SetResus(rhTmpRhesus);

CBloodPressure bpPress(StrToInt(SystPressureEdit->Text),StrToInt(DiastPressureEdit->Text));

piTemp.SetPressure(bpPress);

piTemp.SetPulse(StrToInt(PulseEdit->Text));

paArr.PushBack(piTemp);

if(piTemp.GetPressure().CheckPressure()==0)

ShowMessage (SurnameEdit->Text+" is healthy!");

}

}catch(std::exception & ex)

{

Application->MessageBox(UnicodeString(ex.what()).c\_str(), \_D("Error"), MB\_ICONERROR|MB\_OK);

}

catch(...)

{

Application->MessageBox(UnicodeString("Unknown Error!").c\_str(), \_D("Error"), MB\_ICONERROR|MB\_OK);

}

AddNewPatientForm->Close();

}

//---------------------------------------------------------------------------

void \_\_fastcall TAddNewPatientForm::FormShow(TObject \*Sender)

{

SurnameEdit->Text = "";

SpinEditAge->Value = DEF\_AGE;

RadioGroupBloodType->ItemIndex = -1;

RadioGroupRhesus->ItemIndex = -1;

SystPressureEdit->Text = "";

DiastPressureEdit->Text="";

PulseEdit->Text ="";

}

//---------------------------------------------------------------------------

void \_\_fastcall TAddNewPatientForm::Image1Click(TObject \*Sender)

{

HelpForm->Show();

}

**Help.cpp**

//---------------------------------------------------------------

#include <vcl.h>

#pragma hdrstop

#include "Help.h"

//---------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

THelpForm \*HelpForm;

//---------------------------------------------------------

\_\_fastcall THelpForm::THelpForm(TComponent\* Owner)

: TForm(Owner)

{

}

//---------------------------------------------------------

void \_\_fastcall THelpForm::ExitClick(TObject \*Sender)

{

HelpForm->Close();

}

//---------------------------------------------------------

void \_\_fastcall THelpForm::ListBox1Click(TObject \*Sender)

{

if (ListBox1->ItemIndex == 1)

{

HelpForm->Caption ="About";

RichEdit1->Visible = true;

RichEdit2->Visible = false;

RichEdit3->Visible = false;

RichEdit4->Visible = false;

Image1->Visible=false;

}

else if (ListBox1->ItemIndex == 2)

{

HelpForm->Caption ="Blood Types Info";

RichEdit1->Visible = false;

RichEdit2->Visible = false;

RichEdit3->Visible = false;

RichEdit4->Visible = true;

Image1->Visible=true;

}

else if (ListBox1->ItemIndex == 3)

{

HelpForm->Caption ="Blood Pressure Info";

RichEdit1->Visible = false;

RichEdit2->Visible = true;

RichEdit3->Visible = false;

RichEdit4->Visible = false;

Image1->Visible=false;

}else if (ListBox1->ItemIndex ==4)

{

HelpForm->Caption ="Pulse Info";

RichEdit1->Visible = false;

RichEdit2->Visible = false;

RichEdit3->Visible = true;

RichEdit4->Visible = false;

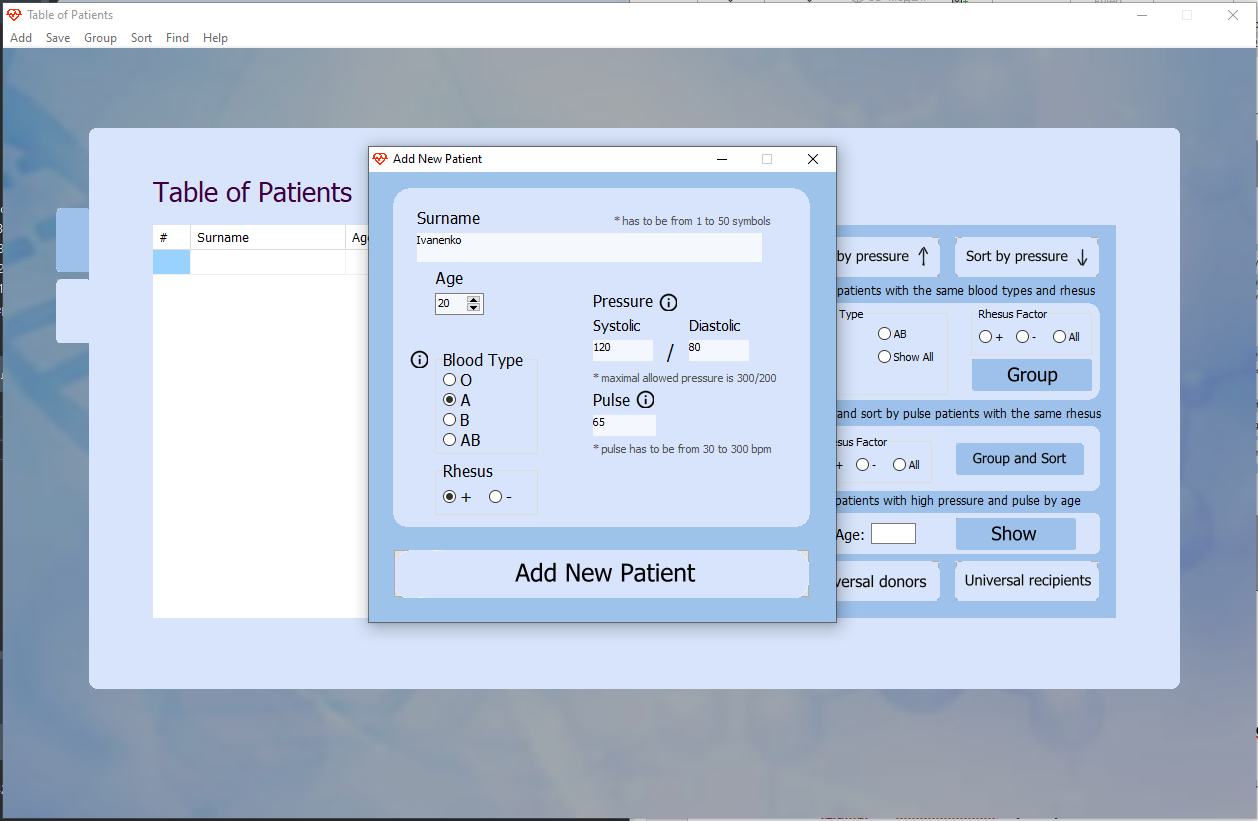
Image1->Visible=false;

}

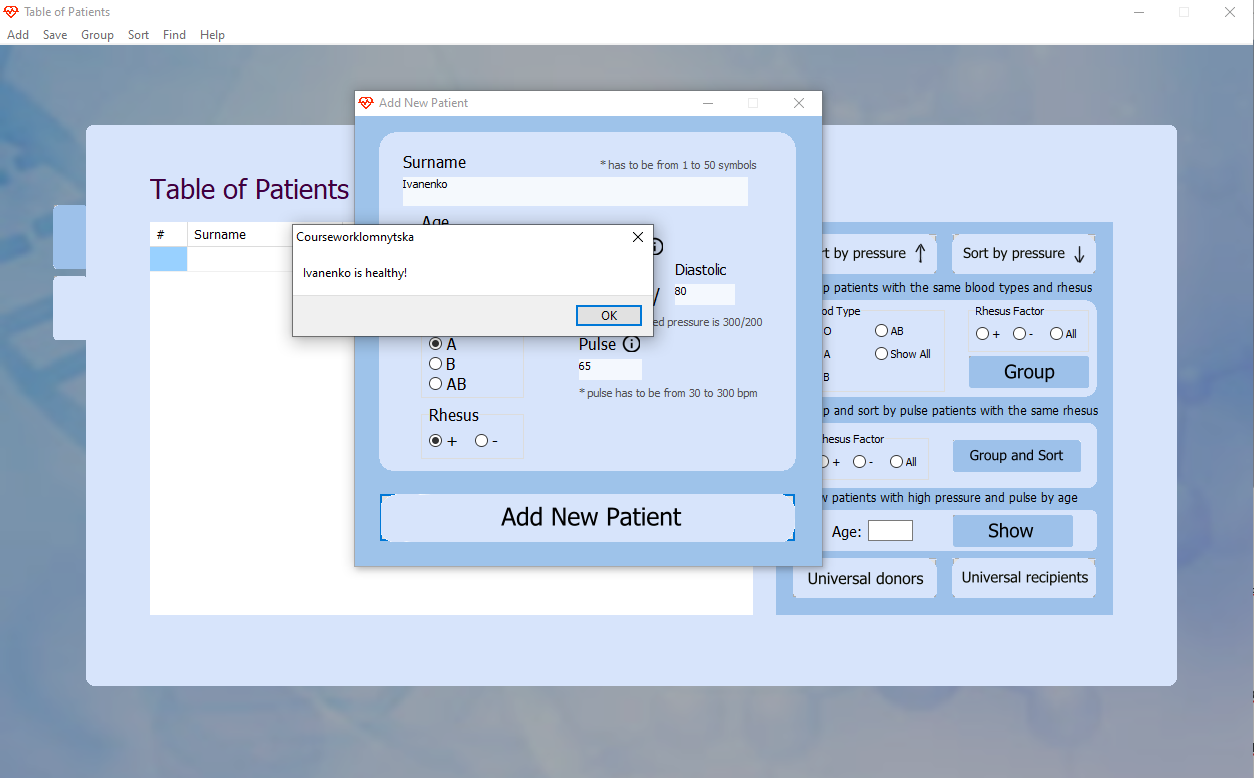
}

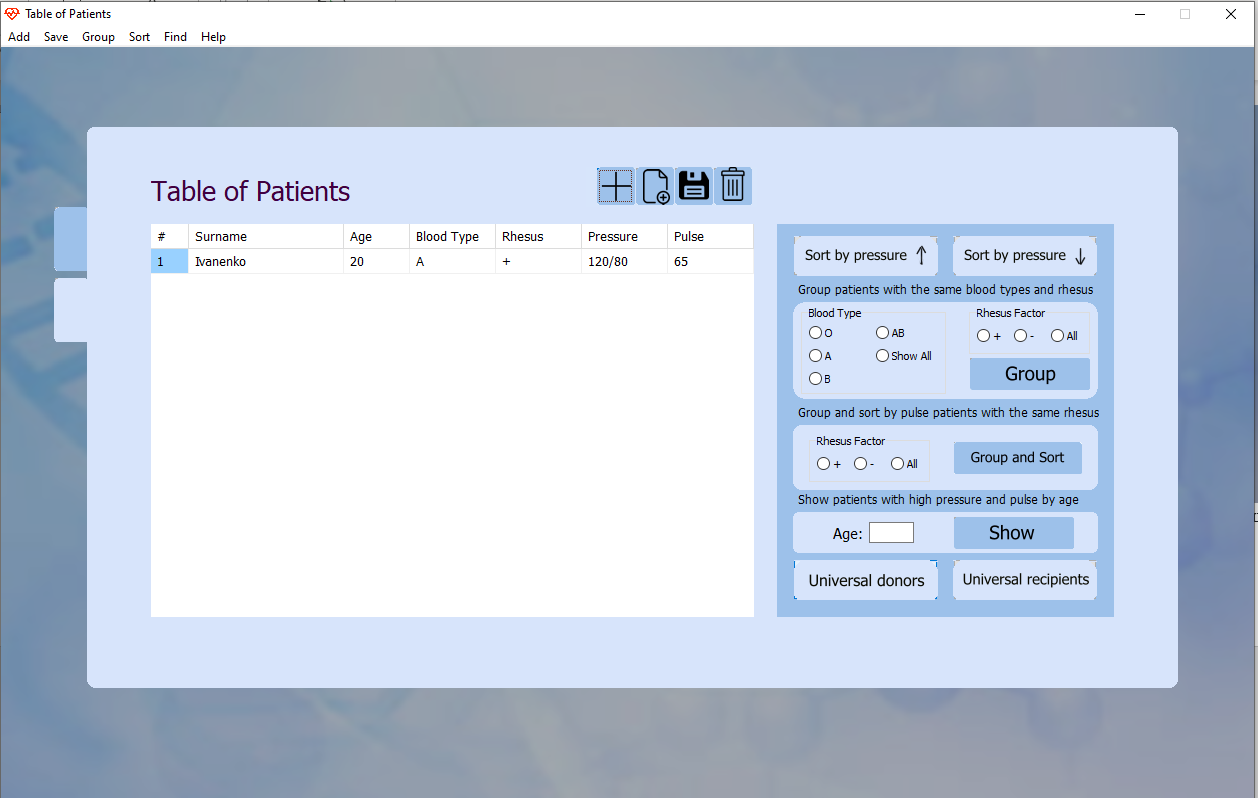
# 4. Протокол роботи для кожного пункту завдання

*Рис.4.1.Головний екран програми.*

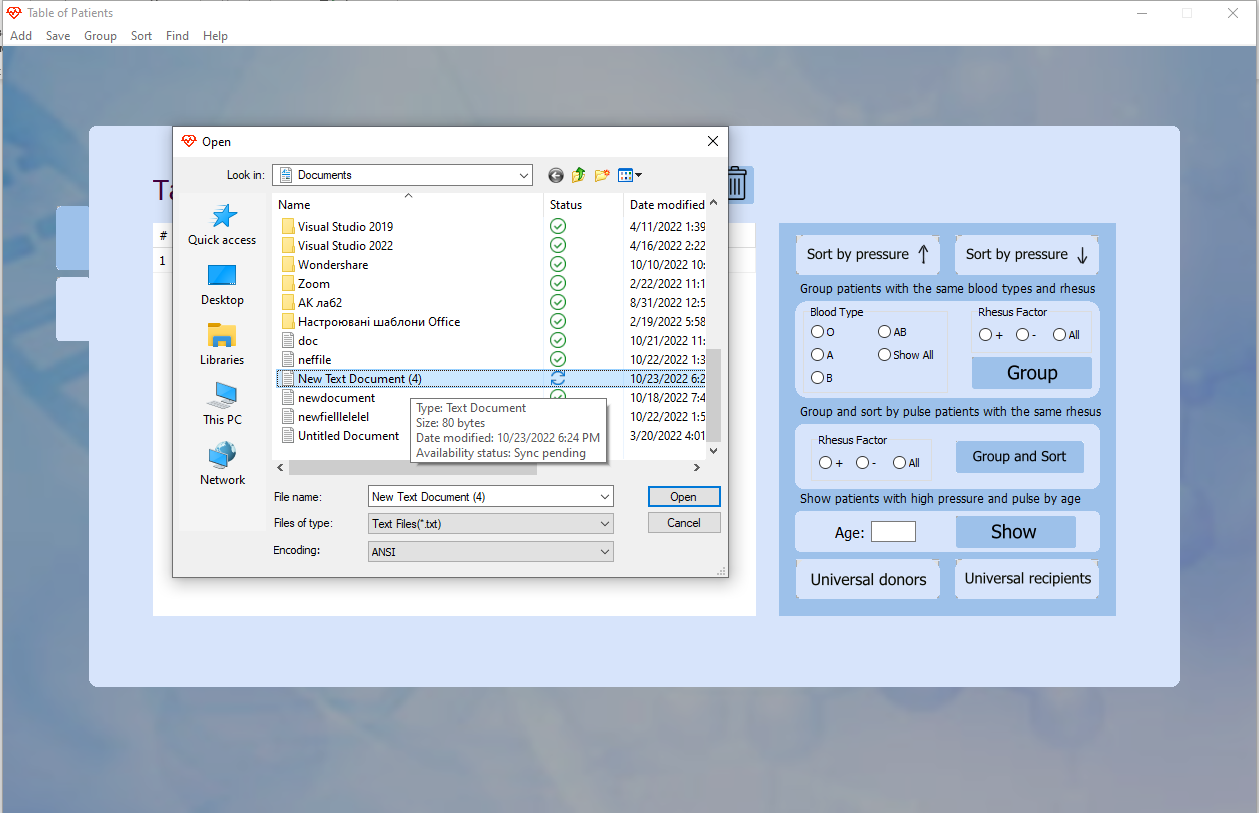
**

*Рис.4.2.Додавання пацієнта шляхом вводу з клавіатури.*

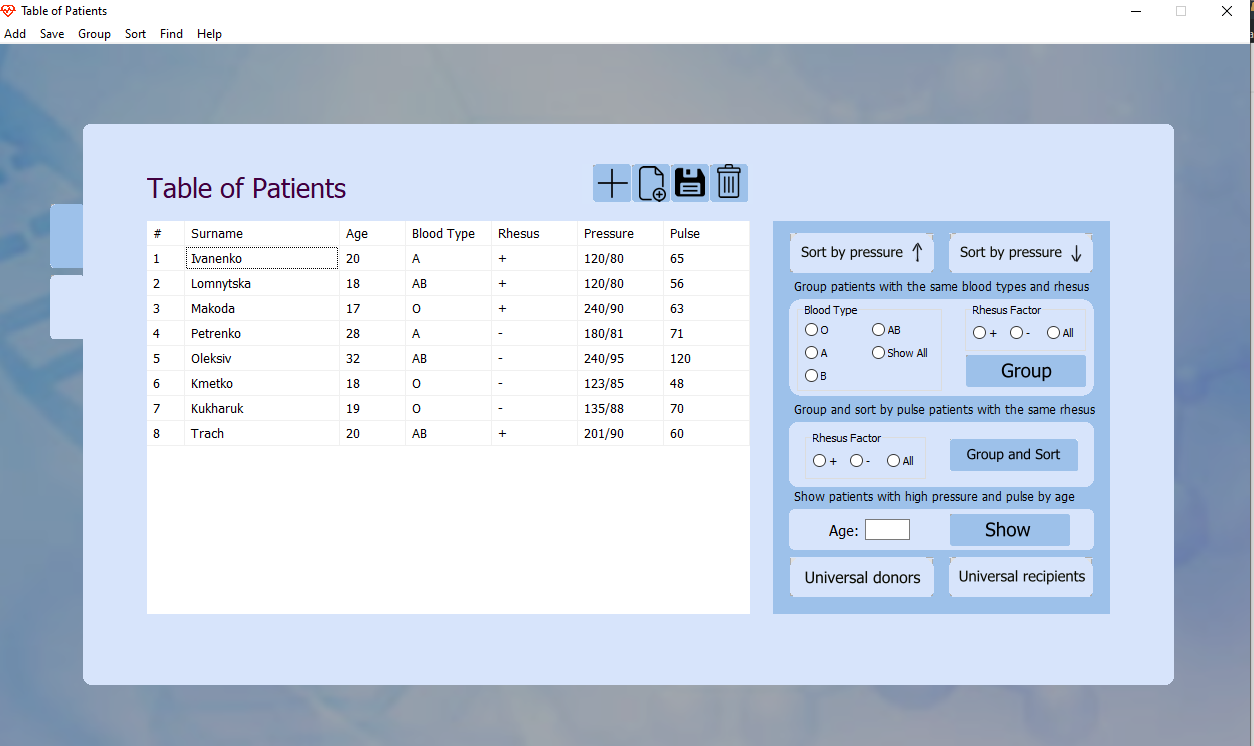
*Рис.4.3.Виведення повідомлення для пацієнта з нормальним тиском та пульсом про те, що він здоровий.*

**

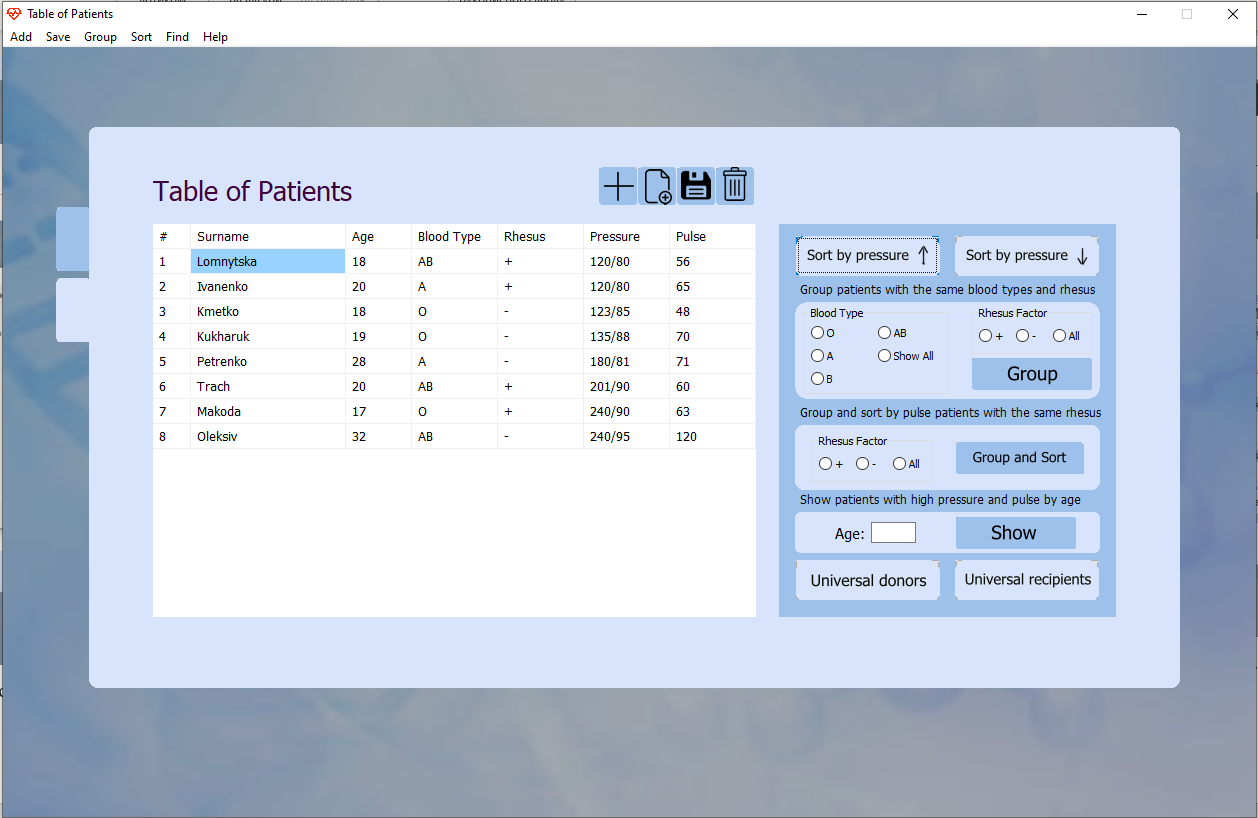
*Рис.4.4.Доданий пацієнт в таблиці.*

**

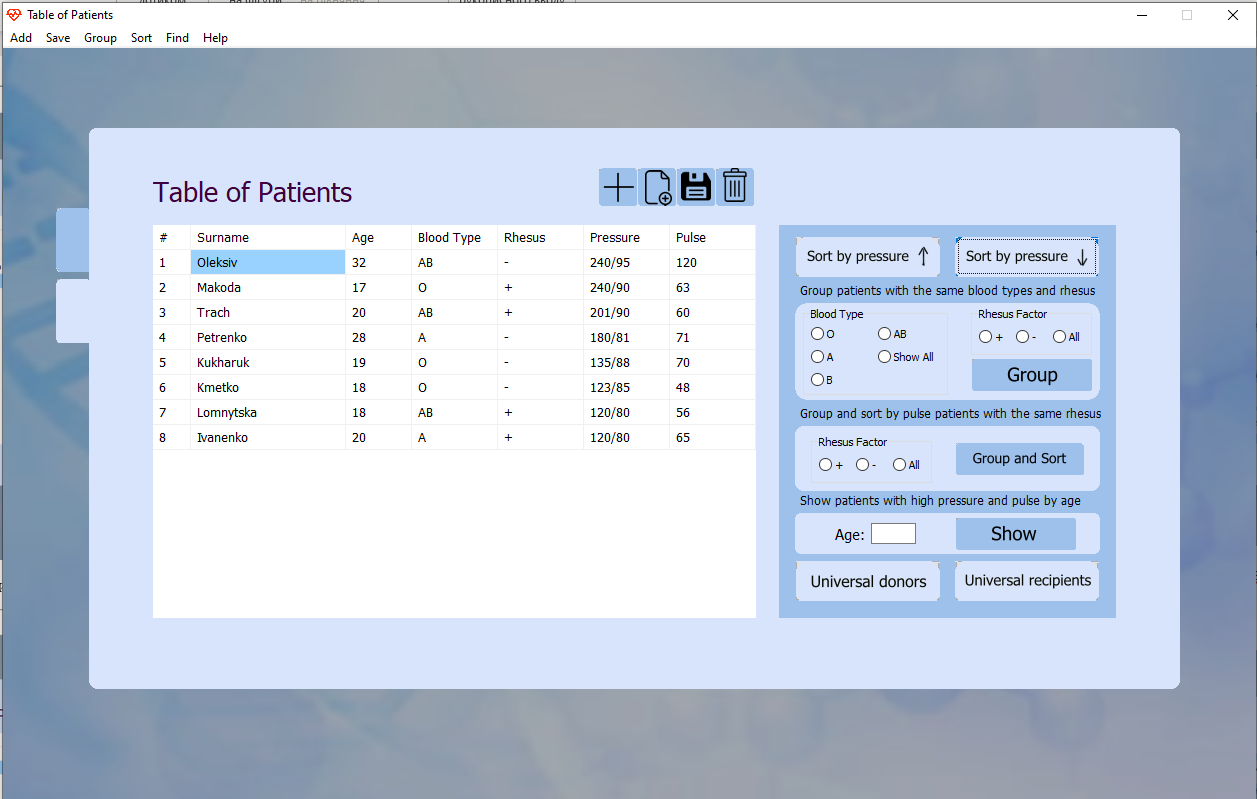
*Рис.4.5.Додавання нового пацієнта з текстового файлу.*

**

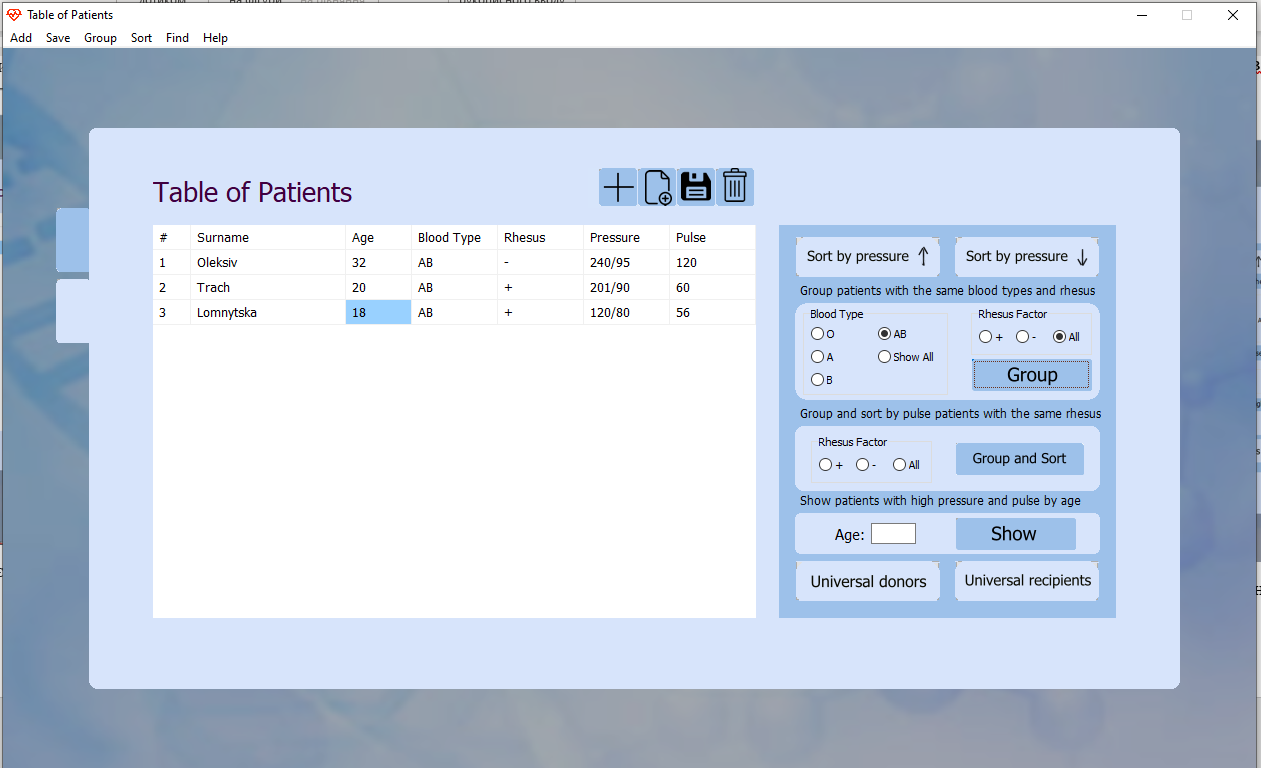
*Рис.4.6.Додані пацієнти в таблиці.*

**

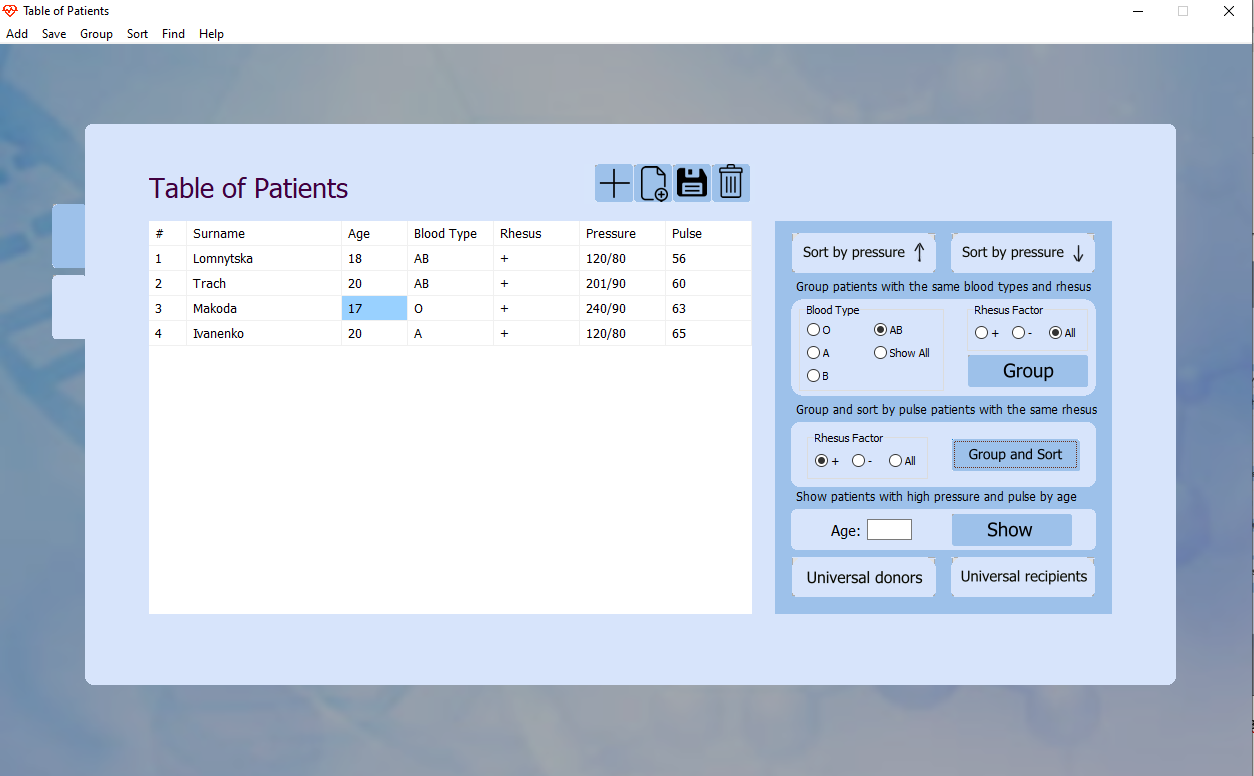
*Рис.4.7.Сортування таблиці за показником тиску в порядку зростання.*

**

*Рис.4.8.Сортування таблиці за показником тиску в порядку спадання.*

**

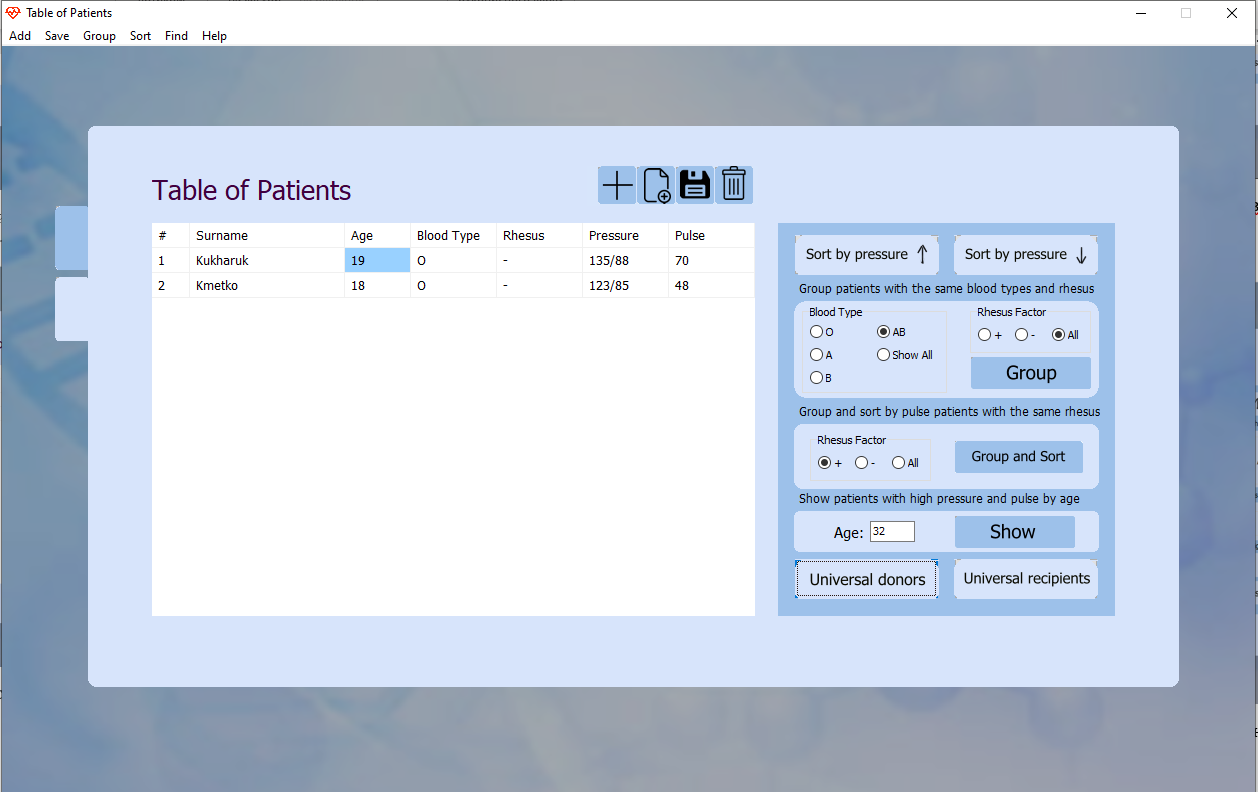
*Рис.4.9.Групування пацієнтів за показником групи крові та резус-фактору.*

**

*Рис.4.10.Групування пацієнтів за показником резус-фактору та сортування кожної з груп за показником пульсу в порядку зростання.*

**

*Рис.4.11.Визначення пацієнтів з підвищеним тиском та пульсом для даного показника віку.*

**

*Рис.4.12.Пошук людей, які є універсальними донорами.*

**

*Рис.4.13.Пошук людей, які є універсальними реципієнтами.*

*Зображення, що містить стіл

Автоматично згенерований опис*

*Рис.4.14.Загальна таблиця донорів та реципієнтів.*

**

*Рис.4.15.Збереження таблиці у текстовий файл.*

*Зображення, що містить текст

Автоматично згенерований опис*

*Рис.4.16.Отриманий текстовий файл.*

*Зображення, що містить стіл

Автоматично згенерований опис*

*Рис.4.17.Демонстрація видалення виділеного елемента: таблиця до видалення.*

*Зображення, що містить текст, знімок екрана, монітор

Автоматично згенерований опис*

*Рис.4.18.Демонстрація видалення виділеного елемента: таблиця після видалення.*

*Зображення, що містить стіл

Автоматично згенерований опис*

*Рис.4.19.Контекстне меню.*

*Зображення, що містить текст

Автоматично згенерований опис*

*Рис.4.20.Вікно довідки.*

# 5. Інструкція користувача та системні вимоги

1. **Компоненти ПЗ.**

Пакет розроблено на мові програмування С++ у середовищі розробки C++ Bulider 10.4 і може експлуатуватися під управлінням сімейства операційних систем Windows. При розробці використовувались стандартні інструменти мови С++. Під час програмування підсистем використовувались принципи об’єктно-орієнтованого програмування. Для коректної роботи пакету необхідна користувацька машина з процесором із тактовою частотою 1 ГГц, оперативою пам’яттю не менше 2 ГБ. Для експлуатації пакету під управлінням сімейства операційних систем Windows необхідно мати в наявності всі необхідні файли бібліотек і налаштувань:

CourseWorkLomnytska.exe – виконавчий файл програми.

1. **Встановлення ПЗ.**

Для роботи пакету необхідно встановити виконавчий файл StudentList2022.exe.

1. Встановити пакет CourseWorkLomnytska.
2. Запустити файл CourseWorkLomnytska.exe.
3. **Налаштування ПЗ.**

Жодних налаштувань системи не потребується.

Для коректної роботи програми з текстовим файлом заповніть його даними у форматі, наведеному у розділі 7.

1. **Базові функції ПЗ.**

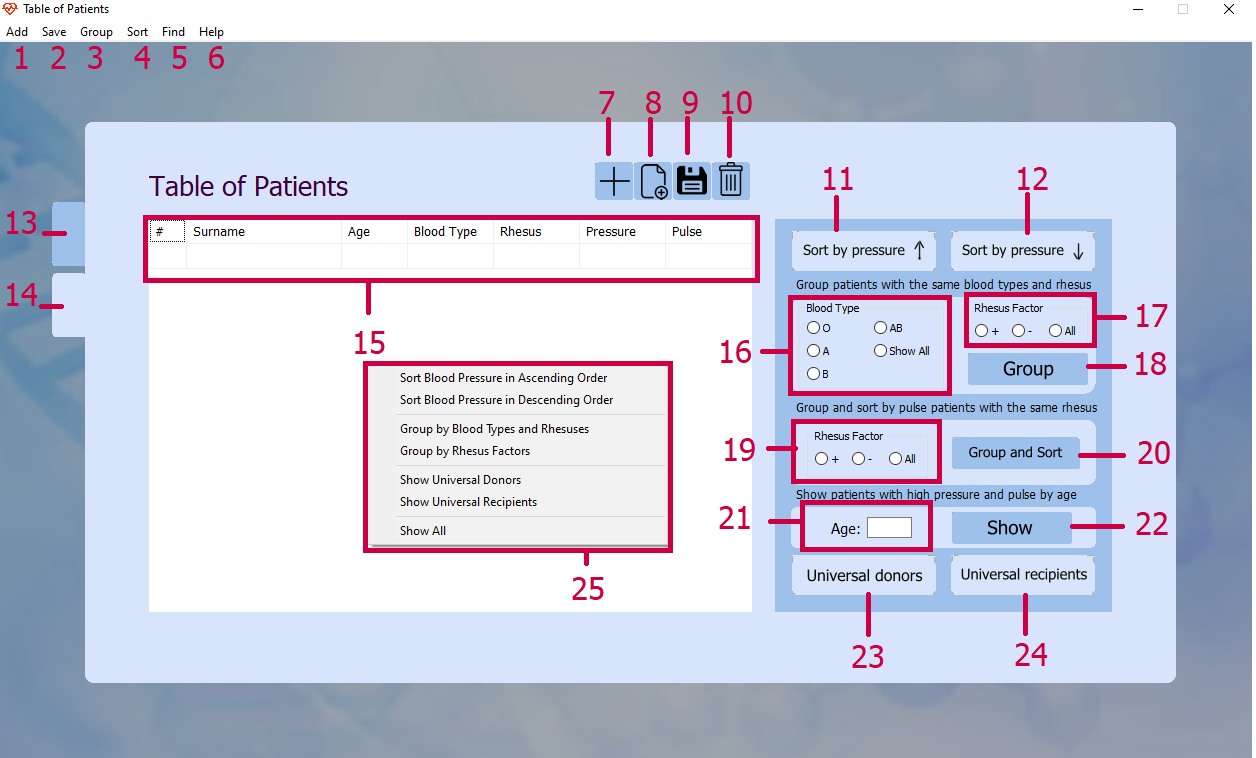
На рис.5.1:

(1) – Розділ головного меню «Add», що містить два варіанти додавання пацієнтів (з файлу та з клавіатури).

(2) – Розділ головного меню «Save», що містить два варіанти збереження таблиці («Зберегти» та «Зберегти як»).

(3) – Розділ головного меню «Group», що містить два варіанти групування таблиці (за групою крові та резус-фактором та тільки резус-фактором).

(4) – Розділ головного меню «Sort», що містить два варіанти сортування таблиці (за зростанням та за спаданням).



*Рис.5.1.Головний екран програми.*

(5) – Розділ головного меню «Find», що містить опції пошуку (пошук універсальних донорів або універсальних реципієнтів).

(6) – Розділ головного меню «Help», що дозволяє відкрити вікно довідки.

(7) – Кнопка для додавання нового пацієнта вручну.

(8) – Кнопка для додавання нових пацієнтів з текстового файлу.

(9) – Кнопка для збереження таблиці у текстовий файл.

(10) – Кнопка для видалення виділеного елемента з таблиці.

(11) – Кнопка для сортування таблиці за показником артеріального тиску в порядку зростання.

(12) – Кнопка для сортування таблиці за показником артеріального тиску в порядку спадання.

(13) – Вкладка з загальною таблицею донорів та реципієнтів.

(14) – Вкладка з таблицею пацієнтів.

(15) – Таблиця пацієнтів, куди виводяться результати користувацьких дій.

(16) – Поле для вибору групи крові для групування.

(17) – Поле для вибору резус-фактору для групування.

(18) – Кнопка для виведення погрупованого за вказаними у полях 16 та 17 критеріями списку у таблицю під номером 15.

(19) – Поле для вибору резус-фактору для групування та сортування.

(20) – Кнопка для виведення погрупованого за критеріями, вказаними у полі 19, списку у таблицю, кожна група з яких посортована в порядку зростання за показником пульсу.

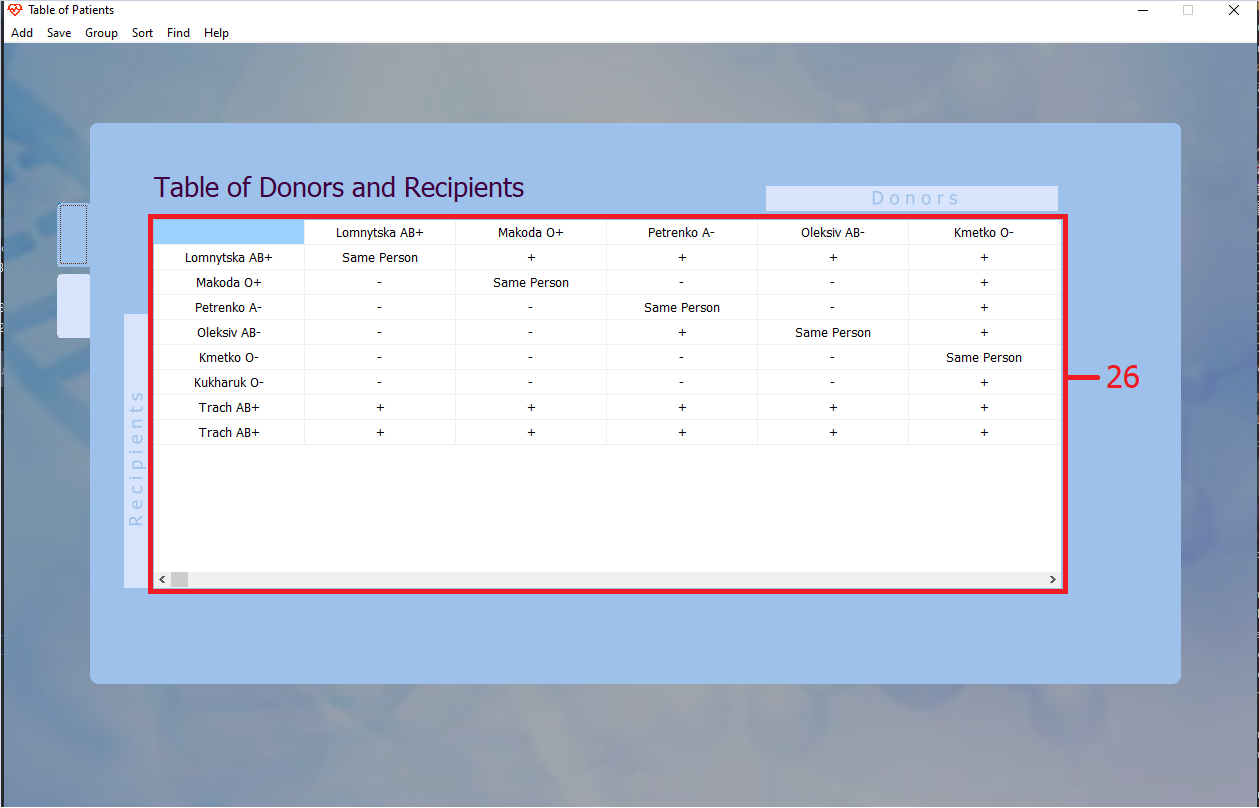
(21) – Поле для вводу віку для пошуку пацієнтів.

(22) – Кнопка для виведення пацієнтів з заданим віком, у яких підвищені пульс і тиск.

(23) – Кнопка для виведення універсальних донорів.

(24) – Кнопка для виведення універсальних реципієнтів.

(25) – Контекстне меню таблиці пацієнтів.



*Рис.5.2.Вкладка таблиці донорів та реципієнтів.*

(26) – Загальна таблиця донорів та реципієнтів, де «+» означає, що кров донора можна переливати реципієнту.



*Рис.5.3.Форма додавання нового пацієнта.*

(27) – Поле вводу прізвища.

(28) – Поле вибору віку.

(29) – Поле вводу артеріального тиску.

(30) – Поле для вибору групи крові.

(31) – Поле для вибору резус-фактору.

(32) – Поле для вводу значення пульсу.

(33) – Кнопка для додавання нового пацієнта в таблицю.



*Рис.5.4.Довідка.*

(34) – Вкладка, що містить інформацію про програму.

(35) – Вкладка, що містить інформацію про групи крові.

(36) – Вкладка, що містить інформацію про артеріальний тиск.

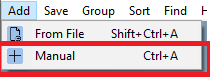
(37) – Вкладка, що містьть інформацію про пульс.

(38) – Кнопка для закривання форми.

Додати нового пацієнта можна двома способами – з клавіатури та з файлу.

Додавання пацієнта з клавіатури:

Для додавання з клавіатури можна натиснути на кнопку  (під номером (7) на рис.5.1) та у відкритому вікні, що зображене на рис.5.3 ввести необхідні дані в поля. Після цього потрібно натиснути на кнопку **«Add New Patient»** (на рис.5.3 під номером (33)). Також можна використати розділ головного меню **«Add»** під номером 1 на рис.5.1 та обрати наступний варіант додавання пацієнта **(«Manual»):**

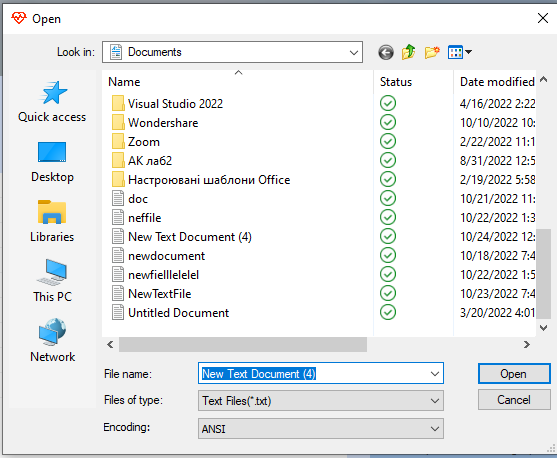


*Рис.5.5. Додавання вручну за допомогою головного меню.*

Після цього відкриється вікно додавання нового пацієнта, що зображене на рис.5.3. Наступні дії аналогічні до додавання за допомогою кнопки .

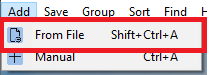
Додавання пацієнта з файлу:

Для додавання інформації з файлу можна натиснути на кнопку . Після цього в діалозі для вибору файлу обрати бажаний та натиснути **«Open»:**



*Рис.5.6. Діалог відкривання файлу.*

Також можна використати розділ головного меню **«Add»** під номером (1) на рис.5.1 та обрати наступний варіант додавання пацієнта **(«From File»):**



*Рис.5.7. Додавання з файлу за допомогою головного меню.*

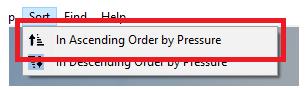
Після цього відкриється діалогове вікно для вибору файлу, що зображене на рис.5.6. Наступні дії аналогічні до додавання за допомогою кнопки .

Для того, щоб відобразити загальну таблицю донорів та реципієнтів необхідно натиснути на вкладку під номером (13) на рисунку 5.1. Після чого відобразиться таблиця, зображена на рис.5.2.

Для того, щоб відобразити таблицю пацієнтів необхідно натиснути на вкладку під номером (14) на рисунку 5.1.

Для того, щоб видалити елемент з таблиці необхідно виділити бажаного пацієнта в області (15) на рис.5.1 та натиснути на кнопку  ( (10) на рис.5.1).

Для сортування таблиці у порядку зростання за показником артеріального тиску необхідно натиснути на кнопку **«Sort by pressure↑»** ( (11) на рис.5.1). Другий варіант виконання дії – використовуючи контекстне меню натиснувши ПКМ на таблиці. В меню, що з’явилось ((25) на рис.5.1) вибрати опцію **«Sort Blood Pressure in Ascending order»**. Третій варіант виконання дії – використовуючи розділ головного меню **«Sort»** під номером (4) на рис.5.1 та обрати наступний варіант сортування:



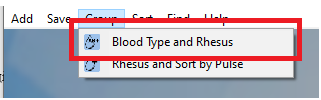
*Рис.5.8.Сортування таблиці за допомогою головного меню.*

Для сортування таблиці у порядку спадання за показником артеріального тиску необхідно натиснути на кнопку **«Sort by pressure ↓»** ( (12) на рис.5.1). Другий варіант виконання дії – використовуючи контекстне меню натиснувши ПКМ на таблиці. В меню, що з’явилось ((25) на рис.5.1) вибрати опцію **«Sort Blood Pressure in Descending order»**. Третій варіант виконання дії – використовуючи розділ головного меню **«Sort»** під номером (4) на рис.5.1 та обрати наступний варіант сортування:



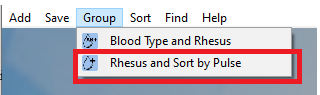
*Рис.5.9.Сортування таблиці за допомогою головного меню.*

Для групування таблиці за групою крові та резус-фактором необхідно обрати бажані параметри для відображення у полях (16) та (17) на рис.5.1 та натиснути кнопку **«Group»** ((18) на рис.5.1). Другий варіант виконання дії – використовуючи контекстне меню натиснувши ПКМ на таблиці. В меню, що з’явилось ((25) на рис.5.1) вибрати опцію **«Group by Blood Types and Rhesuses»**. Третій варіант виконання дії – використовуючи розділ головного меню **«Group»** під номером (3) на рис.5.1 та обрати наступний варіант групування:



*Рис.5.10.Групування таблиці за групою крові та резус-фактором за допомогою головного меню*.

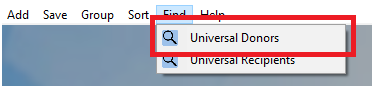
Для групування таблиці за резус-фактором та одночасного сортування кожної групи за значенням пульсу необхідно обрати бажані параметри для відображення у полі (19) на рис.5.1 та натиснути кнопку **«Group and Sort»** (номер (20) на рис.5.1). Другий варіант виконання дії – використовуючи контекстне меню натиснувши ПКМ на таблиці. В меню, що з’явилось ((25) на рис.5.1) вибрати опцію **«Group by Rhesus Factors»**. Третій варіант виконання дії – використовуючи розділ головного меню **«Group»** під номером (3) на рис.5.1 та обрати наступний варіант групування:

**

*Рис.5.11.Групування таблиці за резус-фактором за допомогою головного меню*.

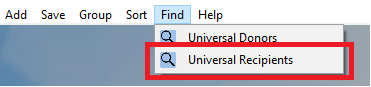
Для пошуку в таблиці пацієнтів з підвищеним пульсом та тиском за вказаним у полі (21) на рис.5.1 віком необхідно натиснути кнопку **«Show»** (номер (22) на рис.5.1).

Для виведення універсальних донорів у таблицю натиснути на кнопку **«Universal donors»** ((23) на рис.5.1). Другий варіант виконання дії – використовуючи контекстне меню натиснувши ПКМ на таблиці. В меню, що з’явилось ((25) на рис.5.1) вибрати опцію **«Show Universal Donors**». Третій варіант виконання дії – використовуючи розділ головного меню **«Find»** під номером (5) на рис.5.1 та обрати наступний варіант пошуку:



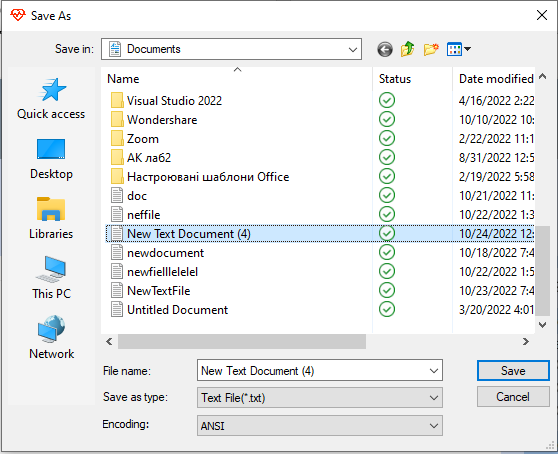
*Рис.5.12.Пошук універсальних донорів за допомогою головного меню*.

Для виведення універсальних реципієнтів у таблицю натиснути на кнопку **«Universal recipients»** ((24) на рис.5.1). Другий варіант виконання дії – використовуючи контекстне меню натиснувши ПКМ на таблиці. В меню, що з’явилось ((25) на рис.5.1) вибрати опцію **«Show Universal Recipients**». Третій варіант виконання дії – використовуючи розділ головного меню **«Find»** під номером (5) на рис.5.1 та обрати наступний варіант пошуку:



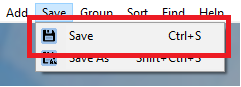
*Рис.5.13. Пошук універсальних реципієнтів за допомогою головного меню*.

Для збереження таблиці у текстовий файл можна натиснути на кнопку  ((9) на рис.5.1). Якщо збереження відбувається вперше, то після цього в діалозі для вибору файлу обрати бажаний або ввести назву нового файлу та натиснути **«Save»:**



*Рис.5.14.Діалог збереження у файл.*

Також можна використати розділ головного меню **«Save»** під номером (2) на рис.5.1 та обрати наступний варіант збереження таблці **(«Save»):**



*Рис.5.15. Збереження в файл за допомогою головного меню.*

Тоді таблиця зберігається в вже існуючий файл, в який вже виконувалось збереження або якщо збереження відбувається вперше, то відкривається діалогове вікно на рис.5.14.

Для збереження таблиці у новий текстовий файл можна використати розділ головного меню **«Save»** під номером (2) на рис.5.1 та обрати наступний варіант збереження таблці **(«Save As»):**

Зображення, що містить текст

Автоматично згенерований опис

*Рис.5.16. Збереження в новий файл за допомогою головного меню.*

Тоді відкривається діалогове вікно на рис.5.14. Якщо збереження відбувається вперше, то після цього в діалозі для вибору файлу обрати бажаний або ввести назву нового файлу та натиснути **«Save».**

1. **Аналіз можливих помилок.**

У випадку помилки зчитування даних перевірте відповідність введення інформації до вимог та пробуйте знову.

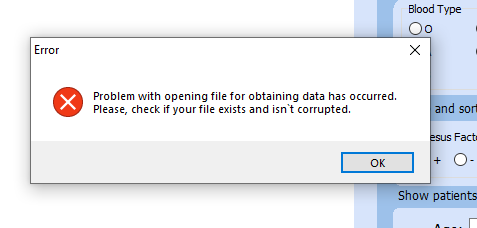
У випадку неможливості відкриття файлу перевірте наявність вказаного файлу або впевніться в його неушкодженісті.

Детальніше аналіз виняткових ситуацій наведено у розділі 6.

**Системні вимоги**

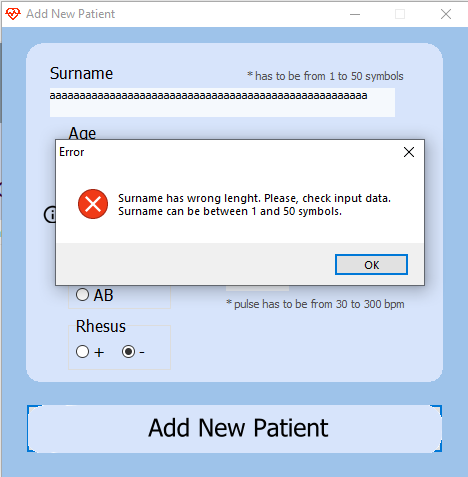
* Процесор: 32-розрядний або 64-розрядний процесор із тактовою частотою 1 ГГц або швидший (наприклад, IntelCore i3);
* ОЗП: 2 ГБ ОЗП (в т.ч. 50 МБ вільних, решта - для підтримки операційної системи) або більше.
* Інше обладнання: під’єднана клавіатура, або інший метод стандартного вводу, наявність монітора або іншого методу стандартного виводу.
* Графічний пристрій із підтримкою DirectX 9 і драйвером WDDM 1.0 або новішим.
* Рекомендована ОС: Windows 7 або вище/MacOS 10.14 або вище/аналогічні вищезгаданим ОС на ядрі Linux. 6
* Спеціальне програмне забезпечення: не вимагається.

# Опис виняткових ситуацій

При спробі відкрити для зчитування неіснуючий текстовий файл користувач отримає наступне повідомлення:

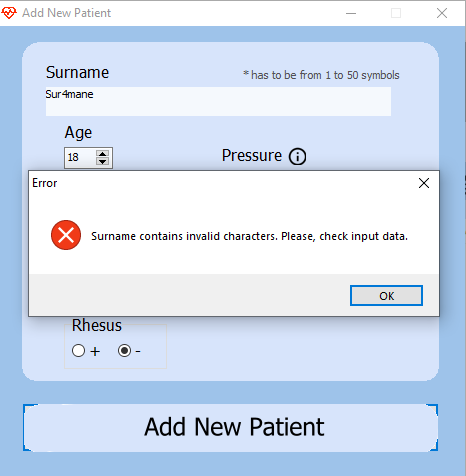
*Рис.6.1.Помилка відкриття файлу.*

При спробі ввести стрічку розміром більшим за 50 символів або меншим за 1 символ як прізвище пацієнта отримаємо наступне повідомлення:



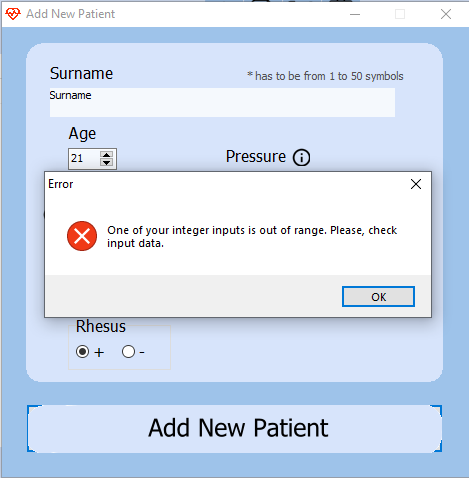
*Рис.6.2.Помилка вводу стрічки неправильної довжини.*

При спробі ввести стрічку з спеціальними символами або цифрами як прізвище пацієнта отримаємо наступне повідомлення:

**

*Рис.6.3.Помилка вводу стрічки зі спеціальними символами або числами.*

Якщо користувач введе значення тиску чи пульсу, що виходить за межі допустимих значень, то отримаємо наступне повідомлення:

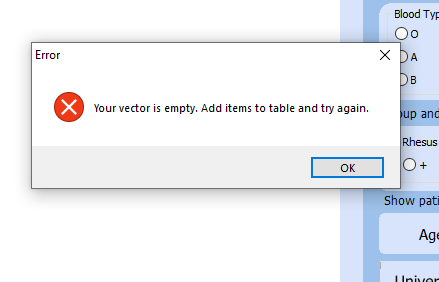


*Рис.6.4.Помилка вводу неправильного цілочисельного значення в одне з полів.*

Якщо користувач залишить одне з полів порожнім, то отримаємо наступне повідомлення:

*Рис.6.5.Помилка при відсутності даних у полі.*

При спробі виконання дії над порожньою таблицею отримаємо наступне повідомлення:

**

*Рис.6.6.Помилка при виконанні дій з порожньою таблицею.*

При спробі видалення елемента без виділення необхідного рядка з інформацією отримаємо повідомлення:

*Зображення, що містить текст

Автоматично згенерований опис*

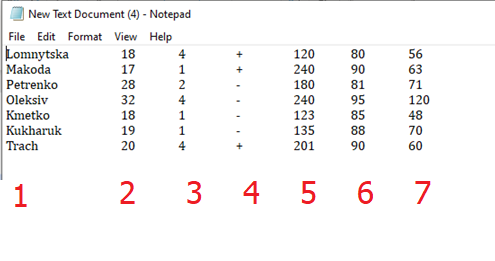
*Рис.6.7.Помилка виходу за межі масиву пацієнтів.*

Зображення, що містить текст

Автоматично згенерований описПри спробі зчитати інформацію пацієнта з порожнього файлу отримаємо повідомлення:

*Рис.6.8.Помилка зчитування з порожнього файлу.*

# Структура файлу вхідних даних

Дані повинні бути записані у текстовий файл з розширенням “.txt”. Приклад структури файлу вхідних даних зображено на рисунку:

*Рис.7.1.Приклад файлу з вхідними даними.*

Поля записуються послідовно і розділяються пробілами або клавішою TAB.

Поле під номером (1) відповідає за прізвище пацієнта – стрічка довжиною від 1 до 50 символів латинського алфавіту без використання спеціальних символів та цифр.

Поле під номером (2) - вік. Допустимими значеннями віку є цілі числа від 1 до 100.

Поле під номером (3) - група крові. Допустимими значеннями поля є цілі числа від 1 до 4, кожне з яких позначає відповідну групу крові 1 – O , 2 – A, 3 – B, 4 – AB.

Поле під номером (4) - резус фактор. Поле може мати лише значення символу «+» або «-».

Поле під номером (5) - верхній тиск, що називається систолічним. Допустимими значеннями верхнього артеріального тиску є цілі числа від 1 до 300.

Поле під номером (6) - нижній тиск, що називається діастолічним. Допустимими значеннями нижнього артеріального тиску є цілі числа від 1 до 200.

Поле під номером (7) - пульс (кількість ударів серця за хвилину). Допустимими значеннями пульсу є цілі числа від 30 до 300.

# Висновки

Виконуючи курсову роботу я створила програмне забезпечення для опрацювання таблиць пацієнтів-донорів крові. Програма дозволяє користувачу додавати нових пацієнтів з текстового документу або вручну за допомогою форми, видаляти пацієнтів, сортувати список, здійснювати пошук та групування. Для реалізації застосунку я використала основні принципи об’єктно-орієнтованого програмування, тобто реалізувала його роботу за допомогою класів CpatientInfo, CpatienArr, CBloodPressure та їх методів. На практиці я закріпила знання про основні теми мови С++, опрацьовані протягом вивчення предмету «Об’єктно-орієнтоване програмування» - функції, класи, наслідування, агрегація та композиція, обробка виняткових ситуацій, робота з файлами, стандартна бібліотека контейнерів, та ін, використавши ці елементи у власному програмному коді. Задокументувала основні структурні особливості програми за допомогою трьох видів UML-діаграм: діаграми класів на рівні реалізації, прецедентів та послідовності. Для створення курсової роботи було використано середовище C++ Builder 10.4 та онлайн-ресурс для створення діаграм – Draw.io.

# Список використаних джерел

1. Левус Є., Мельник Н. Вступ до інженерії програмного забезпечення : навч. посіб. Львів : Вид-во Львів. політехніки, 2018. 246 с.
2. Уроки програмування на С++ для початківців / aCode. aCode. URL: https://acode.com.ua/uroki-po-cpp/ (дата звернення: 26.10.2022).
3. C++ Builder. FunctionX Tutorials. URL: http://www.functionx.com/bcb/index.htm (дата звернення: 20.10.2022).
4. C++ reference. cppreference.com. URL: https://en.cppreference.com/w/cpp (дата звернення: 19.10.2022).
5. Prata S. C++ Primer Plus (4th Edition). 4-те вид. Sams, 2001. 800 с.
6. Stroustrup B. C++ Programming Language. 4-те вид. Addison Wesley, 2013. 1368 с.