|  |  |
| --- | --- |
| Картина, която съдържа текст, вентилатор, устройство  Описанието е генерирано автоматично | **ВИСШЕ ВОЕННОМОРСКО УЧИЛИЩЕ „Н. Й. ВАПЦАРОВ“**  ***9002 Варна, ул. „В. Друмев“ 73, тел.052/632-015, факс 052/303-163*** |
| ***“FILII MARIS SUMUS”***  **ФАКУЛТЕТ „ИНЖЕНЕРЕН“**  **КАТЕДРА „ИНФОРМАЦИОННИ ТЕХНОЛОГИИ“** |
| A picture containing text  Description automatically generated |  |

**KУРСОВ ПРОЕКТ**

**по дисциплината „Основи на програмирането и алгоритмизация“**

**теМА: „**Clinical trials DATA

management program**“**

**гр. Варна**

**2024 г.**

**Студент: Борислав Атанасов Дянков**

**Специалност: Софтуерни и мрежови технологии (DevOps)**

**фак. №: 141-24108**

СЪДЪРЖАНИЕ

1. Какво прави програмата (обобщение)
2. Файлове и папки включени в папката на проекта
3. Библиотеки
4. Код
5. UML DIAGRAM/ DATA FLOWCHART
6. FUTURE IMPROVEMENTS
7. SQL STRUCTURE
8. Какво прави програмата (обобщение)
9. Програмата има за цел да улесни събирането или редактирането на данни от клинични тествания, като ги вкарава или редактира в база данни чрез попълване на текстови полета.
10. Файлове и папки включени в папката на проекта
11. .vscode – тази папка съдържа 3 .json файла, който съдържат инструкциите за това как програмата да бъде компилирана. c\_cpp\_properties.json съдържа информация, например кой компилатор да се използва (в случая gcc-x64 компилатора), информация за това кои външни файлове да бъдат включени при компилация на проекта, за да няма грешки при включването на допълнителни библиотеки (напр. sqlite3 API) и други. settings.json съдържа главно файл асоциациите (библиотеките, които програма използва). tasks.json дава инструкции как точно компилатора да компилира програмата от .c --> .o(binary)/.exe файл за да може да има и програма която се пуска чрез executable а не чрез ./... от терминала.
12. src папката съдържа една подпапка SQLite3, която просто включва .c и .h файлове, в който са описани предефинирани функции и променливи за работа с API при извикване на #include <sqlite3.h>.
13. compile.txt е просто текстови файл където самата compile команда е записана, тъй като не ми се пишеше всеки път като правя промени в кода :). Първо е името на компилатора (gcc), второ е кой .c файл ще бъде компилиран, трето е пътят на директорията на sqlite header файла, тъй като той не включен по default, а след него следва -o, което е просто команда компилатора да направи от .c в .о (т.е. машинен/binary code), следва името на .exe файла, и последно са .dll (dynamic link library) файловете, които са по-малки програми, които програмата ползва за връзка с WinAPI и SQLite API.
14. ct.exe е изпълнимата версия на програмата. С други думи това, което потребителят трябва да вижда при пускане на програмата, а не кода.
15. Trials.db e базата данни, с която програмата си комуникира при определените действия чрез SQLite API. Trials.sql е query файла който дефинира стуктурата на базата данни. Какви таблици да има, какви полета, какво да се случва при update (triggers) и т.н.
16. User\_Guide.docx е подробен наръчник за потребителя как да използва програмата.
17. Библиотеки
18. ctype.h – използвана главно при валидацията на символите при стринговете
19. stdio.h – няма как без нея, това е standart input/output библиотеката
20. stdbool.h – тази библиотека включва boolean променливите true и false, които са съществена част за проекта (валидации на данни и др.)
21. string.h – тази библиотека позволява манипулацията на стрингове (използвана е заради strlen() функцията, която връща дължината на стринга)
22. sqlite3.h – това е SQLite API библиотеката, която предоставя цялата фунцкионалност за манипулация на базата данни на заден фон чрез програмата.
23. windows.h – това е WinAPI, което позволява чрез код да се създават прозорци, бутони, textboxes, labеls, listviews (таблици) и много други графични елементи.
24. commctrl.h (common controls) – използвана е само с цел леко подобряване на listviews, като маркиране на цял ред както съм използвал в кода.

!!! Всички библиотеки са подредени азбучно, защото style50 ми се караше, че трябва да е така :(, и така свикнах. Изключение прави commctrl.h, защото тя зависи от windows.h и не може да е преди нея.

1. Код
2. Глобални дефиниции (помощни функции)

* с HBRUSH (handle to a brush) e дефиниранa четката, с която правя фона син.
* с HWND (handle to a window) са дефинирани бутони, textbox-ове и label-и като който се ползват за въвеждане на данни, вкарване на данните в базата чрез кликане на бутона или подсказки и описания на определни неща чрез labels.
* с HMENU (handle to a menu) саа дефинирани двата съответни изгледа на SQL таблиците

Тези променливи са нужни, за да може WinAPI да знае какъв вид елемент да генерира. Това са препратки към специфични ресурси на Windows applications.

* selectedOption e променлива тип int, която първоначално има стойност -1, тъй като в началото никоя опция не е избрана за работа. По-късно тя приема стойността от енумераторите Insert\_Options, EditView\_Options
* enums – има 4 енумератори като първите два са съответно за текстовите полета и бутоните. Стойностите вътре в тях са ID-та на тези елементи, използвани за разграничаване на елементите от един тип и, за да знае програмата с кой елемент работи. Другите два са за опциите споменати горе за вкарване и редактиране и разглеждане на информацията.

1. Добавени функции за улеснение и техните прототипи

Прототипте на фунцкиите са след глобалните дефиниции и преди главните функции WindowProc, WinMain

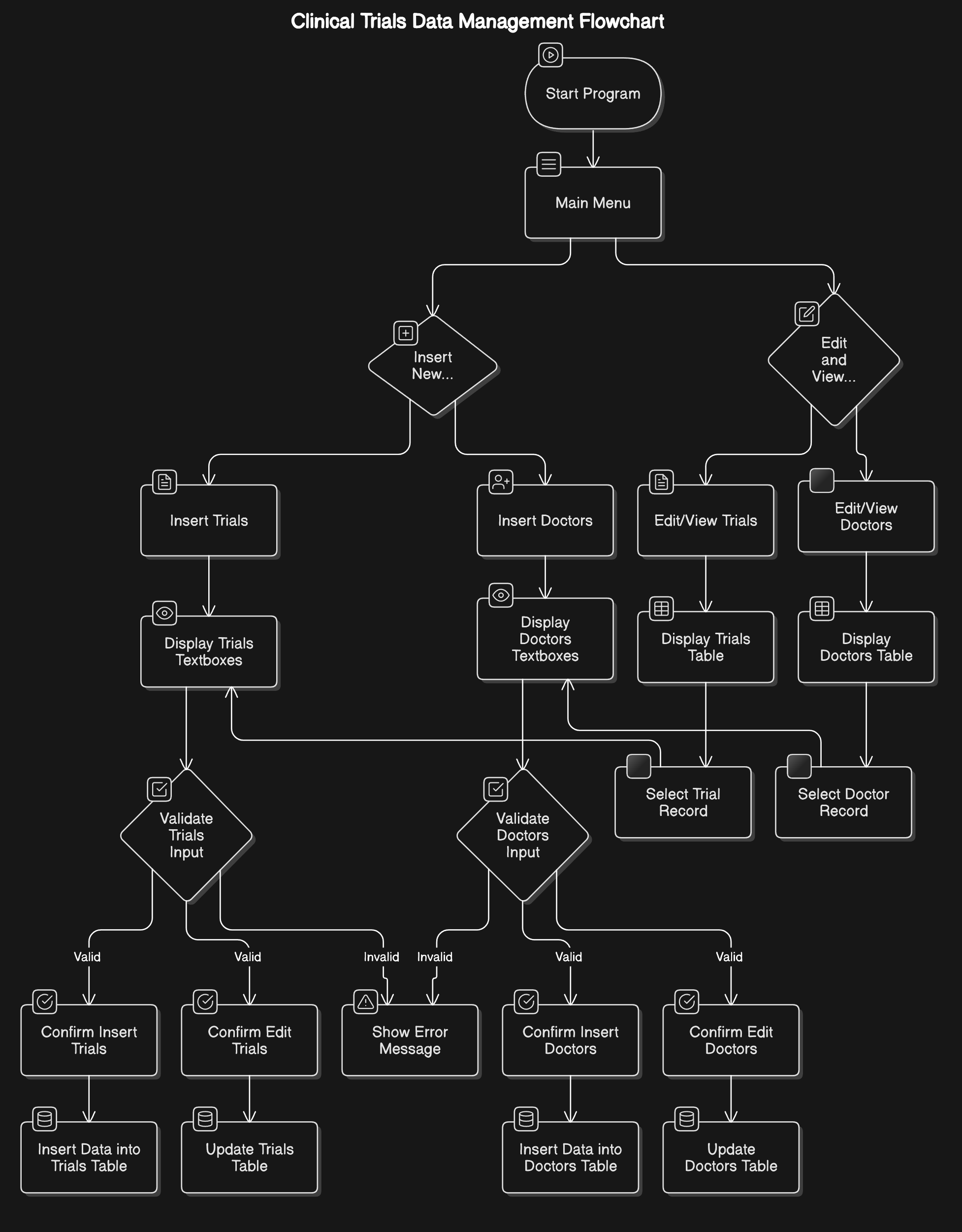
* CreateTextBox връща HWND textbox, който се предaва на съответната променлива за textbox. Има аргументи x y позиция спрямо главния прозорец, width height, HWND, HINSTANCE (което е просо handle за инстанцията на елемента), аргумент за ID на textbox (споменато по-горе при enum). Вътре в тази фунцкия е реалната функция (CreateWindow) за създаване на този елемент, просто аз съм я направил в съкратен вид, за да не се пише толкова дълго и защото някой аргументи се повтарят. „EDIT“ указва на програмата че това е текстово поле, „“ – празно за да няма начален текст при показ на полетата, стиловете на полето са вкарани в един аргумент чрез bitwise OR, като WS\_TABSTOP e малко quality of life, което да позволява на user да фокусира елементите като натиска tab на клавиатурата, WS\_VISIBLE се ползва, за да може прозореца да е видим при пускане на програмата (може да бъде махнато, тъй като те са скрити още в началото на програмата с друга функция, просто ми трябваше в началото за наместване на елементите), WS\_BORDER задава 1px граница на елемента за да може да има разграничение, нищо повече. WS\_CHILD указва, че елемента е подчинен на главния прозорец на програмата. Следващите 4 аргумента са за позицията и размера на елемента, после е hwnd, ID-то (задължително трябваше да е cast 2 пъти), hInst и накрая е аргумента за LPARAM който по default е NULL, тъй като с него се обработват messages от WindowProc (напр. за координати или pointers), а в случая на програмата това не е съществено.
* CreateButton и CreateLabel са на същият принцип просто първият аргумент е „BUTTON“ или „STATIC“ или някой аргументи са махнати или променени.
* SetFont функцията е с цел да се създаде, а после и зададе font на елементите инициализирани с фунцкиите описани горе. Тя е съкратен вид на CreateFont функцията, която приема за аргументи неща от вида – стил, удебеляване, име на шрифта, какъв стандарт за chars да ползва, с какво качество да го рендърва (в случая е използвано ANTIALIASED\_QUALITY за по-модерно и приятно за очите четене), как да го отрязва ако има overflow и т.н. В нея е използвана и друга функция – SendMessage, която просто праща съобщение до WinAPI, че това ще е цялостния вид на шрифта, който елементът ще ползва.
* ViewTrials и ViewDoctors функциите генерират таблиците (listviews) в прозореца като отново се използва споменатата преди фунцкия – CreateWindow. Аргументите на функцията са същите, както при създаването на бутони текстови полети и етикети, но с някои добаввки. LVS\_REPORT, LVS\_SINGLESEL, LVS\_SHOWSELALWAYS указват, че ще е във вид на таблица, че ще може да се селекитра по един ред, и че селекцията на реда е вианги видима. Функцията ListView\_SetExtendedListViewStyle след нея е от commctrl.h, която е използвана за указване, че при кликане ще се селектира целият ред, и че таблицата ще има разделителни линии между колоните. Следват дефинициите за структурата LV\_COLUMN, която за начало е зададен да е {0} (default), за да не стават обърквания при задаването на стойности по-нататък. После следват, задавания на mask (от какво реално се нуждае колоната като параметри – в случая, заглавие, подклетки за инфромация и параметър за ширина в пиксели). След това се задават ширината и заглавието на колоната и се вкарва колоната чрез ListView\_InsertColumn, като аргумента в средата е индекса по който тя да бъде вкарана в таблицата (0 до n-1 (n = брой колони)). Отваря се базата данни и sql statement, който ще бъде подготвян за заявки към db, дефинирана е променлива int row, която се отброява обработените редове от while(sqlite3\_step(stmt) == SQLITE\_ROW) чрез SELECT query-то. Вътре в цикъла се задават параметрите за реда (записа), текста на клетката като се feed-ват от базата данни. Първо е първата колона на реда отделно, а после с for loop се попълват и другите до края на реда. Накрая се затваря връзката с базата данни, за да не се получават нежелани промени при друго действие и приключваме statement т.е. query.

!!! След всяко действие към SQL се затваря db и финализира statement-а.

* HideElements е обобщаваща функция, която скрива елементите, като вътре има доста ShowWindow функции, като те приемат за аргументи елемента, който че бъде скрит и SW\_HIDE за скриване.
* IsNum функцията проверява дали въведеното от потребителя е само числена стойност (без празни места, букви и други знаци) и връща true ако всичко е както трябва (напр. за бройка).
* OnlyAlphaChars проверява дали въеденото от потребителя съдържа само букви или празни места (напр. за имена).
* IsLeapYear връща true ако годината е високосна и false ако не е. Тя е важна част при валидирането на дата в следващата функция при въвеждането ѝ. Проверката става като първо се проверява дали годината се дели на 4 без остатък, за да е високосна и дали се дели на 100 със остатък (за години, които не са начало на век – 2004, 2012 и т.н.), а второто условие проверява дали се дели на 400 без остатък за години, които са начало на век (2000, 2400), защото през 4 века тези години са високосни.
* ValidateDate проверява за точно въведен формат във вида – DD/MM/YYYY. Първо проверява дали дължината на датата е 10 символа (заедно с наклонените черти), после проверява дали тези черти са на правилните позиции (съответно 2-ра и 5-та), след това извлича от целия string деня, месеца и годината. Първият if проверява дали цифрите са реално само цифри, след това, се проверява дали месеца в датата е междду 1 и 12, след това е зададена граница за годината от 1950 г. до 9999 г. Масивът daysInMonth е с 12 стойности съответно за всеки месец и ако въведеният месец е февруари и годината високосна тогава 2-рата стойност от 28 става 29 за да позволява дати като 29/02/2024 например. Накрая се проверява дали денят е в границата от 1 до съответните дни за този месец от масива. Ако въведената дата е валидна тя минава проверката и се вкарва в базата данни.

1. Главни функции – WindowProc и WinMain

* WinMain – това main функцията задължителна за WinAPI. Тя замества int main(). Аргументът hInst е за инстаницята на даден елемент, hPrevInst е за предишна инстаниця и рядко се използва (не е използвано в тази програма също), защото, lpStr е същото като char\*, но е дефинирано така в WinAPI (замества argv) и накрая е show, което се използва за показването на елементи. За да се създаде главния прозорец, задължително се задава име на класа и се инициализира този клас чрез WNDCLASS. Задават се стойности за инстанцията - hInst, коя е функцията, която ще обработва процесите от различни действия в програмата, задава се и вече дефинираното име на класа за прозореца, като накрая съм добавил син фон чрез RGB, за различен цвят от default бял. Проверява се дали успешно е зададен класа на прозореца. Ако всичко е наред, минава към създаването на прозореца чрез CreateWindowEx, която е подобна на CreateWindow. Задават се координати на прозореца за това къде да се появи спрямо дисплея (CW\_USEDEFAULT в случая го поставя леко вляво и нагоре от центъра), размери в пиксели, заглавие, и се използва и името на класа, с който е инициализиран този прозорец. Отново се прави проверка дали създаването му е наред. Ако всичко до тук е минало без грешки се добавят подобаващите елементи разположени в прозореца, скриват се първоначално чрез HideElements, После се показва прозореца и се обновява, ако са правени промени по него. While цикълът след това, праща всяко съобщение/event (кликане, появяване на елементи и други), за да може то да бъде обработено от WindowProc. Накрая на WinMain фунцкията се връща return(int)msg.wParam, което е като return 0, за да покаже че всичко е успешно изпълнено с програмата.
* WindowProc – Тази функция обработва цялата комуникация на програмата от елементите до базата данни. Тя съдържа switch блок, като cases са процесите в програмата (msg). Първият case е за правене на фона на всеки етикет прозрачен (по default e бял), вторият е за начертаване на черната разделителна линия (като се използва HDC за графично изобразяване), третият е за изпращане на съобщението за затваряне на програмата към return(int)msg.wParam – 0. Следващият case е по-особен, тъй като в него са обособени важните процеси свързани с обмяната на данни от програмата към SQL и обратно. В него има нов switch, в който всеки case е селектирана опция или при натискане на бутон, какво да се случва. Първите 3 са за опциите за вкарване на данни в двете таблици. 1 – показва textbox-ове за Trials и бутона (скрива за Doctors), 2 – показва за Doctors (скрива за Trials). Третият case вече изпълнява същинската идея на програмата. Първо се проверявва коя опция е селектирана (Insert\_Into\_Trials/ Insert\_Into\_Doctors).
* При Insert\_Into\_Trials се правят buffer стрингове, които приемат стойностите от съответните textbox-овете чрез GetWindowText, след това се разделят двете дати на ден, месец и година и се събират тези стойности обръщат в int, за да се сравняват (ако началната дата е след крайната). Следват чекове, за да се валидират всички стойности чрез вече създадените помощни функции. Проверяват се дали има празни полета, дали имената са само с букви и spaces, дали броя на участниците е само числена стойност (без spaces и до 6 цифри), дали датите са валидни като формат и времево. Ако проверките не върнат грешка при input (MessageBox-овете), се отваря базата данни за работа с въведените данни. Проверява се дали успешно се отваря базата данни. След като е успешно отворена се проверява дали въведеният доктор за LeadDr е във Doctors, ако го има като ред в таблицата се използва името му за trial, в противен случай се добавя и тогава се използва (има и проверка ако процеса е неуспешен - за всеки случай). След това се проверява дали има такъв запис за trial вече, ако не, се вкарват съответните данни в съответните колони в таблицата Trials и се появява MessageBox за потвърждаване на успешен insert.
* Идеята за Insert\_Into\_Doctors e същата като при Trials.
* При EditView\_Trials и EditView\_Doctors идеята е същата за показване на полета (+1 допълнително за ID на записа), а бутона вместо „Confirm Insert“ е „Confirm Edit“. Проверките са същите. Разликите откъм заявките е, че накрая се използва UPDATE вместо INSERT INTO, за да се осъществи редактирането на данни в определен запис (селектиран по input-a в полето за Trial/Doctor ID). Допълнително се показват и таблиците с данните при избиране на опция от („Edit and View…“), за да може потребителят да разглежда и знае кой ред променя. Refresh на изгледа на таблицата при добавяне или редактиране става като се избере отново една от опциите от „Edit and View…”, като това се осъщесвява чрез FindWindowEx, което търси вече съществуващо listview, унищожава го (DestroyWindow), и го изкарва наново с новите попълнения.

1. UML DIAGRAM/ DATA FLOWCHART
2. FUTURE IMPROVEMENTS

Могат да се добави търсене по специални критерии - т.е SELECT query със специални критерии (дата, име на доктор, специалност, име на болестта и т.н)

Могат да се добавят още полета за данни в двете таблици.

Може да се добави ORDER BY query, което да е възможност чрез селектиране на опция за подреждане - по възходящ/низходящ ред на имена, участници, дати, като цяло по всеки критерии), а може би даже и комбинирано сортиране.

1. SQL STRUCTURE

A screenshot of a computer program

AI-generated content may be incorrect.