Эта страница не нужна!!!

Её можно не печатать, а начинать со 2й.

Или здесь сделать титулку, не суть.

**Зміст**

[Вступ 3](#_Toc478433617)

[Основна частина 5](#_Toc478433618)

[Алгоритм роботи (російською) 5](#_Toc478433619)

[Вихідні файли 8](#_Toc478433620)

[Файл «main.cpp» 8](#_Toc478433621)

[Файл «inc.h» 22](#_Toc478433622)

[Файл «prototypes.h» 22](#_Toc478433623)

[Файл (конфігураційний) «mysql.ini» 22](#_Toc478433624)

[Файл «def.h» 23](#_Toc478433625)

[Файл-скрипт БД MySQL 25](#_Toc478433626)

[Скріншоти роботи програми 28](#_Toc478433627)

[Висновок 32](#_Toc478433628)

# Вступ

Темою дипломної роботи є «Автоматизація підбору ліків по діагнозу». Це означає, що потрібно розробити програму, що буде давати можливість лікарю вибрати правильні ліки для кожного пацієнта.

Під автоматизацією роботи можна розуміти певний зв'язок між лікарем і програмним середовищем. У якості такого цього на даний час використовуються різноманітні бази даних. Програма, яка дозволяє об’єднати дані, формує основу автоматизації підбору ліків.

**Бази даних** − це систематизовані сховища певної інформації. Вони створюються тоді, коли необхідно відслідковувати, аналізувати і зберігати інформацію за деякий час. Якщо поняття бази даних наблизити до інформаційних систем, які розміщуються на комп’ютерних носіях даних, то під відповідним терміном можна розуміти набір записів та файлів, які організовані спеціальним чином. Така організація дозволяє за будь-якої необхідності завжди отримати доступ до даних з якими працюємо. Задля зручності доступу до інформації використовуються додаткові графічні оформлення.

Будь-яка база даних, з точки зору її створення, в мовах програмування повинна складатися з трьох компонентів: файлу даних, об’єкту зв’язку і інтерфейсу. Файл даних містить структуру даних, що включає таблиці, запити, фільтри, функції, тощо. Об’єктом зв’язку виступає об’єкт мови програмування, що здійснює зв'язок між файлом даних і інтерфейсом. А інтерфейс включає вже комплекс засобів взаємодії з користувачем.

Тому, для використання та модифікації процесів в базах даних застосовують системи управління базами даних. Маючи на меті задачу реалізацій медичного страхування було вибрано середовищем розробки бази даних MySQL.

В якості мови програмування було обрано системну об’єктно-орієнтовану мову C++.

У зв’язку з тим, що мова C++ не підтримує обробку баз даних MySQL стандартними бібліотеками та методами, було застосовано так званий MySQL Connector. Для того щоб узгодити роботу середи розробки, «коннектора» та сервера бази даних необхідно встановлювати програмне забезпечення (ПЗ) тієї ж самої розрядності, що і використовуваний компілятор. Тобто, якщо компілятор має архітектуру x86, то відповідно необхідно встановити MySQL Connector та MySQL Server тієї ж самої розрядності, так само і для x64 архітектури.

Побудова БД виконувалась згідно із завданням.

**Завдання до дипломної роботи:**

Необхідно розробити базу даних «Ліків» , Відповідно користувачем моєї бази даних буде лікар.

Вхідні дані:

* Інформація про ліки;
* Інформація про діагноз;
* інформація про склад ліків;
* інформація про протипоказання;
* інформація про виробника;
* інформація про форму діагноза.

Вихідні дані:

* Список препаратів;
* Список діагнозів;
* Список виробників;
* Список протипоказань;
* Список складових препаратів;
* Список форм.

Розроблене програмне забезпечення повинно виконувати такі функції:

* Ведення списку ліків;
* Ведення списку виробників;
* Ведення списку діагнозів;
* Ведення списку складу ліків;
* Ведення списку протипоказань;
* Формування звітів.

Контекстна діаграма:

|  |
| --- |
|  |
| Рисунок 1 – Контекстна діаграма для БД «Інформаційна система підбору ліків за діагнозом» |

**Опис інформації:**

Для опису інформації, яка перетворюється процесами (таблиця 1).

Таблиця 1 — Словник даних контекстної діаграми ПЗ підбір ліків лікарем.

|  |  |
| --- | --- |
| Ім'я потоку | Структура потоку |
| 1 | 2 |
| Інформація від лікаря | * інформація про ліки; * інформація про діагноз; * інформація про склад ліків; * інформація про протипоказання; * інформація про виготовника; * запит на пошук ліків по:  1. по назві ліків; 2. по виготовнику; 3. по діагнозу. 4. По властивостям 5. По показанням складу 6. По групі 7. По фірмі 8. По передозуванню 9. Для вагітних 10. Для дітей 11. По алергічним реакціям 12. По побічним реакціям 13. По взаємодії з ліками 14. По режиму дозування 15. По застосуванню |
| Інформація для лікаря | * список ліків; * Діагноз; * список виготовників; * списки складу ліків; * Список протипоказання на ліки; |

**Концептуальна модель та її опис:**

В результаті вивчення предметної області та постанови задачі було виявлено основі сутності, атрибути та зв’язки між ними (представлені в таблиці 2,3 та рисунку 2).

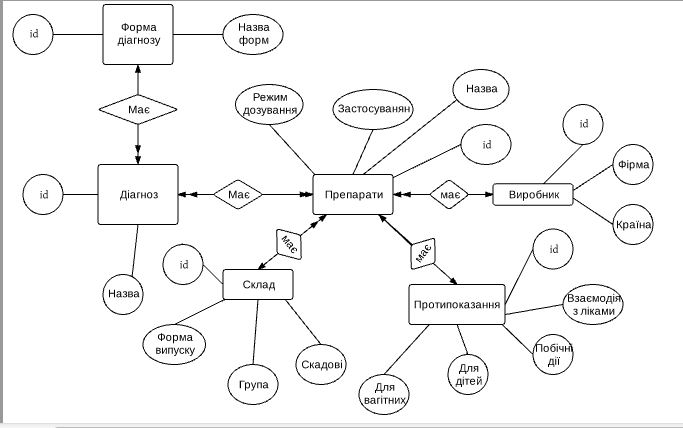
Таблиця 2 – Основні сутності та їх атрибути

|  |  |
| --- | --- |
| Сутність | Атрибути |
| Ліки | Id ліків  Назва  Застосування  Режим дозування |
| Діагноз | Id діагнозу  Назва  Форма |
| Форма діагнозу | Id  Назва форми |
| Склад | id  Форма випуску  Група  Складові |
| Виготовник | Id  Фірма  Країна |
| Протипоказання | Id  Передозування  Для вагітних  Для дітей  Побічні реакції  Взаємодія з ліками |

Таблиця 3 – Сутність-зв'язок

|  |  |  |
| --- | --- | --- |
| Сутності | Тип зв’язку | Зміст зв’язку |
| Форма діагнозу  Діагноз | 1:М | Одна форма може мати багато діагнозів |
| Діагноз  Препарати | 1:М | Один діагноз може мати багато препаратів |
| Препарати  Протипоказання | 1:М | Одні препарати можуть мати багато різних протипоказань |
| Препарати  Виробник | М:1 | Ліки можуть мати багато різних виробників |
| Препарати  Склад | 1:М | Одні ліки можуть мати багато складових |

**Діаграма «сутність-зв’язок»**

****

|  |
| --- |
|  |
| Рисунок 2 – Концептуальна модель |

**Логічна модель**

На основі концептуальної моделі описала та побудувала логічну модель, яку представила на рисунку 3.

Форма діагнозу(diagnosis\_form): ID форм – id, назва форм - name.

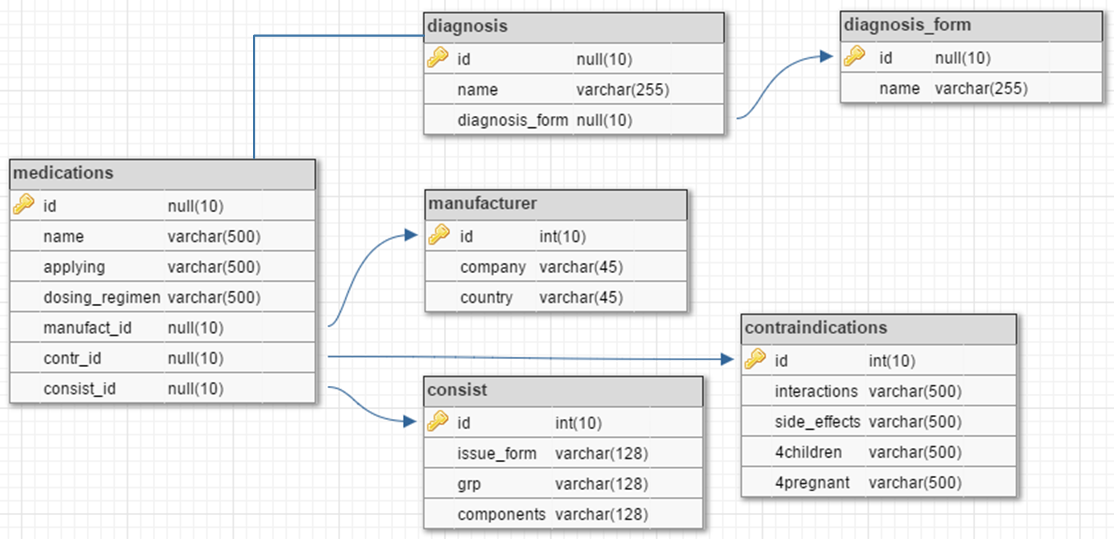
Діагноз(diagnosis): ID діагнозу – id, назва – name, форма– diagnosis\_form.

Склад (consist): ID – id, форма випуску – issue\_form, група – grp, складові - components.

Препарати (medications): назва – name, застосування – applying, режим дозування – dosing\_regimen.

Протипоказання (containdications): ID протипоказання – id, interactions – взаємодія с препаратами, побічні дії – side\_effects, для дітей – 4children, для вагітних – 4pregnant

Виробник (manufacturer): ID виробника - id, назва виробника - company, країна виробника - country.



**Фізична модель бази даних:**

Модель таблиці «diagnosis\_form»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Індефікатор поля | Ознака ключа | Тип даних | Обмеження | Примітка |
| id | PK | INT | NOT NULL, AUTO INCREMENTAL | Код форми діагнозу |
| name |  | VARCHAR(225) | NOT NULL | Назва форми діагнозу |

Модель таблиці «diagnosis»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Індефікатор поля | Ознака ключа | Тип даних | Обмеження | Примітка |
| id | PK | INT | NOT NULL, AUTO INCREMENTAL | Код діагнозу |
| name |  | VARCHAR(225) | NOT NULL | Назва форми діагнозу |
| form | FK | INT | NOT NULL | Посилання на поле  «id»таблиці «diagnosis\_form» |

Модель таблиці «medications»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Індефікатор поля | Ознака ключа | Тип даних | Обмеження | Примітка |
| id | PK | INT | NOT NULL, AUTO INCREMENTAL | Код препаратів |
| name |  | VARCHAR(500) | NOT NULL | Назва |
| applying |  | VARCHAR(500) | NOT NULL | Застосування |
| dosing\_regimen |  | VARCHAR(500) | NOT NULL | Режим дозування |
| manufact\_id | FK | INT | NOT NULL | Посилання на поле «id» таблиці «manufacturer» |
| contr\_id | FK | INT | NOT NULL | Посилання на поле «id» таблиці «contradications» |
| consist\_id | FK | INT | NOT NULL | Посилання на поле «id» таблиці «consist» |

Модель таблиці «maufacturer»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Індефікатор поля | Ознака ключа | Тип даних | Обмеження | Примітка |
| id | PK | INT | NOT NULL,  AUTO INCREMENTAL | Код виробника |
| company |  | VARCHAR(45) | NOT NULL | Фірма |
| country |  | VARCHAR(45) | NOT NULL | Країна |

Модель таблиці «contraindications»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Індефікатор поля | Ознака ключа | Тип даних | Обмеження | Примітка |
| id | PK | INT | NOT NULL, AUTO INCREMENTAL | Код протипоказаня |
| interactions |  | VARCHAR(500) | NOT NULL | Взаємодія  з препаратами |
| side\_effects |  | VARCHAR(500) | NOT NULL | Побічні дії |
| 4children |  | VARCHAR(500) | NOT NULL | Для дітей |
| 4pregnant |  | VARCHAR(500) | NOT NULL | Для вагітних |

Модель таблиці «consist»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Індефікатор поля | Ознака ключа | Тип даних | Обмеження | Примітка |
| id | PK | INT | NOT NULL,  AUTO INCREMENTAL | Код складу |
| issue\_form |  | VARCHAR(128) | NOT NULL | Форма випуску |
| grp |  | VARCHAR(128) | NOT NULL | Група |
| components |  | VARCHAR(128) | NOT NULL | Складові |

# Основна частина

В основній частині розглядається суть виконуваної роботи, алгоритм роботи програми, вихідні файли C++, та скрипт-модель БД MySQL.

Після того, як завантажилось середовище розробки Microsoft Visual C ++ 2010 Express, потрібно створити новий порожній проект "Консольний додаток Win32". Потім встановлюється "Release" як значення конфігурації рішення (Solution configurations). Далі потрібно відкрити сторінку властивостей (Property Pages) за допомогою головного меню (Проект-> Властивості).  
У вікні сторінки властивостей спочатку додається до параметру  
"Додаткові каталоги включення файл" (Властивості конфігураціі-> С/С ++ -> Загальні) рядок "C:\Program Files\MySQL\MySQL Server 5.1\include", після — до параметру "Додаткові каталоги бібліотек" (Властивості конфігураціі-> Компоновщік-> Загальні) рядок "C:\Program Files\MySQL\MySQL Server 5.1\lib\opt", далі - до параметру "Додаткові залежності" (Властивості конфігураціі-> Компоновщік-> Введення) додати  
значення "mysqlcppconn-static.lib" і "libmysql.lib", також до параметру "Визначення препроцесора" (Властивості конфігураціі-> С/С++-> Препроцесор) додати значення "CPPCONN\_PUBLIC\_FUNC=".  
Примітка. Файли mysqlcppconn-static.lib і libmysql.lib (знаходяться в архіві  
mysql-connector-c++-noinstall-1-1.1.0-win32.zip) потрібно помістити в каталог C:\Program Files\MySQL\MySQL Server 5.1\lib\opt.  
MySQL-сервер налаштовується таким чином. Вперше, майстер конфігурації сервера (Пуск> Все програми-> MySQL-> MySQL Server 5.1> MySQL Server Instance Config Wizard) і в вікні, потрібно крок за кроком вибирати:

* Reconfigure Instance
* Detailed Configuration
* Developer Machine
* Multifunctional Database
* Вибрати диск і шлях для зберігання БД типу "InnoDB"  
  Decision Support (DSS) / OLAP.
* Активувати Enable TCP / IP Networking Enable і Strict Mode, порт залишити за замовчуванням (3306)
* Standard Character Set
* Активувати Install as Windows Service (якщо цей сервіс був встановлений в процесі інсталяції, то цей елемент буде недоступним) і Include Bin Directory in Windows PATH.
* Визначити пароль адміністратора, якщо до цього часу він не був заданий

# Алгоритм роботи програми (російською)

1. Устанавливаем поддержку кириллицы в потоке вывода текущей консоли

2. Подключаемся к БД

2.1. Парсим (обрабатываем) файл с данными, необходимыми для подключения к БД

2.2. Подключаемся

2.3. Возникла ошибка подключения

2.3.1. «ДА» - повторяем попытку подключения

2.3.2. «НЕТ» - переходим ко следующему пункту

3. Отправляем запросы базе данных на установку кодировки, которая поддерживает кириллицу

4. Заполняем главное меню

5. Ожидаем нажатие клавиши (выбор пункта в меню)

5.1. Посылаем запрос в БД, основываясь на номере нажатой клавиши

5.1.1. «Вытаскиваем» имя БД из файла

5.1.2. Устанавливаем связь с выбранной БД (выбираем её для дальнейшей работы)

5.1.3. Произошла ошибка во время выполнения предыдущего действия

5.1.3.1. «ДА» - выводим на экран ошибку, которая произошла, и завершаем работу программы

5.1.3.2. «НЕТ» - переходим ко следующему пункту

5.1.4. Посылаем собственно сам запрос, в котором просим получить некоторые поля с таблицы

5.1.5. Произошла ошибка во время выполнения предыдущего действия

5.1.5.1. «ДА» - выводим на экран ошибку, которая произошла, и завершаем работу программы

5.1.5.2. «НЕТ» - переходим ко следующему пункту

5.2. Создаём меню, основываясь на номере нажатой клавиши (переходим в пункт 6 данного алгоритма).

5.3. Если нажат 0 (ноль), то завершаем работу программы

6. Узнаём кол-во строк в результирующем наборе

7. Максимальное кол-во страниц вычисляем по формуле:

<число\_строк> : <пунктов\_на\_страницу>

8. Если остаток от деления числа строк на кол-во пунктов на страницу равен нулю, тогда единожды декрементируем макс. число страниц

9. Если значение текущей страницы больше макс. числа страниц или меньше ноля (выходит за пределы), тогда присваиваем её макс. числу страниц или нолю соответственно.

10. Очищаем экран

11. Выводим название меню и ранее вычисленные значения страниц

12. Выводим пункты меню, текст которых состоит из соответствующих полей, которые «пришли к нам» в результате удачного запроса

13. Добавляем пункты «8. Назад» и «9. Вперёд» для перемещения между страницами

14. Добавляем пункт «0. Главное меню» для выхода в главное меню

15. Вызываем соответствующий хендлер

15.1. Ожидаем нажатие клавиши

15.2. Если значение клавиши равно 9 (девяти) или 8 (восьми), тогда перемещаемся по страницам, вызвав «по новой» то же самое меню, указав номер страницы больше/меньше на 1, чем текущий

15.3. Если значение равно 0 (нулю), тогда возвращаемся к пункту 4 этого алгоритма (открываем главное меню)

15.4. Если значение отличное, от предыдущих, но разрешено для нажатия, тогда, используя ранее полученные данные из БД, выводим информацию о пункте, который был выбран

(UPD: в листинге можно будет увидеть, что используется трёхмерный динамический массив (char\*\*\*), в котором нет разбиения на страницы, а есть лишь единая нумерация, поэтому «страничная нумерация» преобразуется в единую по формуле:

<номер\_выбран.\_пункта> + <страница\_выб.\_пункта> \* <число\_пунктов\_на\_стран.>)

15.5. Т.к. «мы прошли» по всем пунктам, каретка в БД находится на последней строке, и при следующем шаге выйдет за пределы БД. Для предотвращения подобной ошибки устанавливаем каретку на 0 (нулевую) строку

15.6. Задерживаем экран ожиданием нажатия на любую клавишу

15.7. Обновляем отображение текущей меню путём возврата к пункту 5.2 этого алгоритма

# Вихідні файли

## Файл «main.cpp»

#include "inc.h"

#pragma warning(disable:4244) // Отключаем вывод предупреждения 4244 при компиляции

#pragma warning(disable:4474) // Отключаем вывод предупреждения (безопасность printf) 4474 при компиляции

HANDLE hConsoleHandle = GetStdHandle(STD\_OUTPUT\_HANDLE); // Получаем дескриптор потока вывода текущей консоли для изменения цвета текста и фона

MYSQL \*connection, mysql; // Структуры, позволяющие взаимодействовать с БД

MYSQL\_RES \*result; // Структура, позволяющая обрабатывать результирующий набор

MYSQL\_ROW row; // Двумерный массив для записи строк

void main(void)

{

setlocale(LC\_ALL, "Russian"); // Поддержка кириллицы

unsigned int i = 0;

while(!connect\_to\_db())

{

system("cls");

printf(" Не удалось соединиться с базой данных!\n", SetErrorTextColor());

printf(" Ошибка: \"%s\"\n", mysql\_error(&mysql));

printf(" Неудачных попыток: %d\n\n", ++i);

printf(" Нажмите любую клавишу для повторной попытки подключения,\n", SetStandardTextColor());

printf(" либо нажмите Escape для завершения работы программы...\n\n");

if(\_getch() == KEYCODE\_ESC)

{

exit(EXIT\_FAILURE);

}

}

m\_main();

}

bool connect\_to\_db()

{

char sHost[32], sUser[32], sPass[32], sDB[32], sPort[32];

GetPrivateProfileStringA("mysql\_init", "host", NULL, sHost, charsmax(sHost), ".\\mysql.ini"); // Здесь и ниже парсим файл с данными для подключения к БД

GetPrivateProfileStringA("mysql\_init", "user", NULL, sUser, charsmax(sUser), ".\\mysql.ini");

GetPrivateProfileStringA("mysql\_init", "pass", NULL, sPass, charsmax(sPass), ".\\mysql.ini");

GetPrivateProfileStringA("mysql\_init", "db", NULL, sDB, charsmax(sDB), ".\\mysql.ini");

GetPrivateProfileStringA("mysql\_init", "port", NULL, sPort, charsmax(sPort), ".\\mysql.ini");

mysql\_init(&mysql); // Инициализация структуры

connection = mysql\_real\_connect(&mysql, sHost, sUser, sPass, sDB, atoi(sPort), NULL , 0); // Подключаемся к БД

if(connection == NULL) // Если соединение не удалось

{

return false;

}

mysql\_query(&mysql, "SET NAMES cp1251"); // Для кириллицы, если в БД не установлена нужная кодировка

mysql\_query(&mysql, "SET CHARACTER SET cp1251");

return true;

}

/\* Главное меню \*/

void m\_main(void)

{

int iKeys = 0;

mysql\_free\_result(result); // Очищаем результирующий набор

system("cls");

printf(" %s\n", sMenuNames[0], SetHeaderTextColor());

for(unsigned int i = 1; i < arraySize(sMenuNames); i++) // По всему массиву, кроме первого элемента ("Главное меню")

{

printf("\n %d. ", i, SetNumberTextColor()); // Выводим циферки специального цвета

printf("%s", sMenuNames[i], SetStandardTextColor()); // Выводим пункты обычного цвета

add\_key(i, iKeys); // Добавляем пункт в бит. сумму, чтобы можно было на него нажимать

}

printf("\n\n\n 0. ", SetNumberTextColor());

printf("Выйти", SetStandardTextColor());

add\_key(MENU\_KEY\_0, iKeys);

mh\_main(iKeys);

}

/\* Хендлер главного меню \*/

void mh\_main(int iKeys)

{

switch(get\_key(iKeys)) // Ожидаем нажатие клавиши, которая заключена в бит. сумме iKeys

{

case MENU\_KEY\_1:

{

if(!read\_medications()) // Если запрос удался, то

menu\_create(mMedications, 0); // Создаём меню лекарств. 0 - начальная страница

break;

}

case MENU\_KEY\_2:

{

if(!read\_diagnosis())

menu\_create(mDiagnosis, 0);

break;

}

case MENU\_KEY\_3:

{

if(!read\_compositions())

menu\_create(mCompositions, 0);

break;

}

case MENU\_KEY\_4:

{

if(!read\_cntraindications())

menu\_create(mContraindications, 0);

break;

}

case MENU\_KEY\_5:

{

if(!read\_manufacturers())

menu\_create(mManufacturers, 0);

break;

}

case MENU\_KEY\_0:

{

exit(1);

}

}

}

/\* Создаём меню \*/

void menu\_create(int iMenu, int iPage)

{

int iMaxPages = mysql\_num\_rows(result) / ITEMS\_PER\_PAGE, iKeys = 0; // Определяем максимальное кол-во строк

if(!(mysql\_num\_rows(result) % ITEMS\_PER\_PAGE)) iMaxPages--;

if(iPage > iMaxPages) // Корректируем номер текущей страницы, дабы он не вышел за рамки предела

iPage = iMaxPages;

if(iPage < 0)

iPage = 0;

/\* Начало: заполнение меню \*/

system("cls"); // Очищаем экран

printf(" %s", sMenuNames[iMenu + 1], SetHeaderTextColor()); // Выводим название меню

printf(" [%d/%d]\n", iPage + 1, iMaxPages + 1, SetDisableTextColor()); // номер текущей страницы и общее число страниц

char \*\*sRows = (char\*\*)malloc(sizeof(char\*) \* mysql\_num\_rows(result)); // Создаём двумерный динамический массив для получения результата запроса

for(unsigned int i = 0; row = mysql\_fetch\_row(result), i < mysql\_num\_rows(result); i++)

{

sRows[i] = (char\*)malloc(sizeof(char) \* strlen(\*row));

sRows[i] = \*row;

}

mysql\_data\_seek(result, 0); // Т.к. после предыдущего цикла каретка оказалась в конце таблицы, нужно её переместить в нулевую позицию для дальнейшей работы

SetStandardTextColor();

for(unsigned int i = 0; i < ITEMS\_PER\_PAGE; i++) // Выводим пункты

{

if((i + iPage \* ITEMS\_PER\_PAGE) < mysql\_num\_rows(result))

{

printf("\n %d. ", i + 1, SetNumberTextColor());

printf("%s", sRows[i + iPage \* ITEMS\_PER\_PAGE], SetStandardTextColor());

add\_key((i + 1), iKeys);

}

else

{

printf("\n");

}

}

if(!iPage) printf("\n\n 8. Назад", SetDisableTextColor()); // Если страница нулевая, тогда закрашиваем тёмносерым цетом

else

{

add\_key(MENU\_KEY\_8, iKeys);

printf("\n\n 8. ", SetNumberTextColor());

printf("Назад", SetStandardTextColor());

}

if(iPage >= iMaxPages) printf("\n 9. Вперёд", SetDisableTextColor());

else

{

add\_key(MENU\_KEY\_9, iKeys);

printf("\n 9. ", SetNumberTextColor());

printf("Вперёд", SetStandardTextColor());

}

add\_key(MENU\_KEY\_0, iKeys);

printf("\n\n 0. ", SetNumberTextColor());

printf("%s", sMenuNames[0], SetStandardTextColor());

/\* Конец: заполнение меню \*/

switch(iMenu) // Вызов хендлера

{

case mMedications : mh\_medications(iPage, iKeys);

case mDiagnosis : mh\_diagnosis(iPage, iKeys);

case mCompositions : mh\_compositions(iPage, iKeys);

case mContraindications : mh\_contraindications(iPage, iKeys);

case mManufacturers : mh\_manufacturers(iPage, iKeys);

}

}

/\* Хендлер меню лекарств \*/

void mh\_medications(int iPage, int iKeys = 0)

{

int iKey = get\_key(iKeys);

switch(iKey)

{

case MENU\_KEY\_8:

{

menu\_create(mMedications, --iPage);

break;

}

case MENU\_KEY\_9:

{

menu\_create(mMedications, ++iPage);

break;

}

case MENU\_KEY\_0:

{

m\_main();

break;

}

default:

{

iKey--;

MYSQL\_ROW \*sRows = (char\*\*\*)malloc(sizeof(char\*\*) \* mysql\_num\_rows(result));

for(unsigned int i = 0; row = mysql\_fetch\_row(result), i < mysql\_num\_rows(result); i++)

{

sRows[i] = (char\*\*)malloc(sizeof(char\*) \* mysql\_num\_fields(result));

for(unsigned int j = 0; j < mysql\_num\_fields(result); j++)

{

sRows[i][j] = (char\*)malloc(sizeof(char) \* strlen(row[j]));

sRows[i][j] = row[j];

}

}

printf("\n\n\t———————————————————————————————————————————————————————————————\n", SetSeparatorColor()); // Выводим информацию по пункту, который был нажат

printf("\t %21s | ", "Название", SetHeaderTextColor());

printf("%s\n", sRows[iKey + iPage \* ITEMS\_PER\_PAGE][0], SetStandardTextColor());

printf("\t———————————————————————————————————————————————————————————————\n", SetSeparatorColor());

printf("\t %21s | ", "Показания к применению", SetHeaderTextColor());

printf("%s\n", sRows[iKey + iPage \* ITEMS\_PER\_PAGE][1], SetStandardTextColor());

printf("\t———————————————————————————————————————————————————————————————\n", SetSeparatorColor());

printf("\t %21s | ", "Дозировка", SetHeaderTextColor());

printf("%s\n", sRows[iKey + iPage \* ITEMS\_PER\_PAGE][2], SetStandardTextColor());

printf("\t———————————————————————————————————————————————————————————————\n", SetSeparatorColor());

mysql\_data\_seek(result, 0);

\_getch();

menu\_create(mMedications, iPage);

}

}

}

/\* Хендлер меню диагнозов \*/

void mh\_diagnosis(int iPage, int iKeys = 0)

{

int iKey = get\_key(iKeys);

switch(iKey)

{

case MENU\_KEY\_8:

{

menu\_create(mDiagnosis, --iPage);

break;

}

case MENU\_KEY\_9:

{

menu\_create(mDiagnosis, ++iPage);

break;

}

case MENU\_KEY\_0:

{

m\_main();

break;

}

default:

{

iKey--;

MYSQL\_ROW \*sRows = (char\*\*\*)malloc(sizeof(char\*\*) \* mysql\_num\_rows(result));

for(unsigned int i = 0; row = mysql\_fetch\_row(result), i < mysql\_num\_rows(result); i++)

{

sRows[i] = (char\*\*)malloc(sizeof(char\*) \* mysql\_num\_fields(result));

for(unsigned int j = 0; j < mysql\_num\_fields(result); j++)

{

sRows[i][j] = (char\*)malloc(sizeof(char) \* strlen(row[j]));

sRows[i][j] = row[j];

}

}

printf("\n\n\t———————————————————————————————————————————————————————————————\n", SetSeparatorColor());

printf("\t %8s | ", "Название", SetHeaderTextColor());

printf("%s\n", sRows[iKey + iPage \* ITEMS\_PER\_PAGE][0], SetStandardTextColor());

printf("\t———————————————————————————————————————————————————————————————\n", SetSeparatorColor());

mysql\_data\_seek(result, 0);

\_getch();

menu\_create(mDiagnosis, iPage);

}

}

}

/\* Хендлер меню составов лекарств \*/

void mh\_compositions(int iPage, int iKeys = 0)

{

int iKey = get\_key(iKeys);

switch(iKey)

{

case MENU\_KEY\_8:

{

menu\_create(mCompositions, --iPage);

break;

}

case MENU\_KEY\_9:

{

menu\_create(mCompositions, ++iPage);

break;

}

case MENU\_KEY\_0:

{

m\_main();

break;

}

default:

{

iKey--;

MYSQL\_ROW \*sRows = (char\*\*\*)malloc(sizeof(char\*\*) \* mysql\_num\_rows(result));

for(unsigned int i = 0; row = mysql\_fetch\_row(result), i < mysql\_num\_rows(result); i++)

{

sRows[i] = (char\*\*)malloc(sizeof(char\*) \* mysql\_num\_fields(result));

for(unsigned int j = 0; j < mysql\_num\_fields(result); j++)

{

sRows[i][j] = (char\*)malloc(sizeof(char) \* strlen(row[j]));

sRows[i][j] = row[j];

}

}

printf("\n\n\t———————————————————————————————————————————————————————————————\n", SetSeparatorColor());

printf("\t %12s | ", "Состав", SetHeaderTextColor());

printf("%s\n", sRows[iKey + iPage \* ITEMS\_PER\_PAGE][0], SetStandardTextColor());

printf("\t———————————————————————————————————————————————————————————————\n", SetSeparatorColor());

printf("\t %12s | ", "Группа", SetHeaderTextColor());

printf("%s\n", sRows[iKey + iPage \* ITEMS\_PER\_PAGE][1], SetStandardTextColor());

printf("\t———————————————————————————————————————————————————————————————\n", SetSeparatorColor());

printf("\t %12s | ", "Форма выпуска", SetHeaderTextColor());

printf("%s\n", sRows[iKey + iPage \* ITEMS\_PER\_PAGE][2], SetStandardTextColor());

printf("\t———————————————————————————————————————————————————————————————\n", SetSeparatorColor());

mysql\_data\_seek(result, 0);

\_getch();

menu\_create(mCompositions, iPage);

}

}

}

/\* Хендлер меню противопоказаний \*/

void mh\_contraindications(int iPage, int iKeys = 0)

{

int iKey = get\_key(iKeys);

switch(iKey)

{

case MENU\_KEY\_8:

{

menu\_create(mContraindications, --iPage);

break;

}

case MENU\_KEY\_9:

{

menu\_create(mContraindications, ++iPage);

break;

}

case MENU\_KEY\_0:

{

m\_main();

break;

}

default:

{

iKey--;

MYSQL\_ROW \*sRows = (char\*\*\*)malloc(sizeof(char\*\*) \* mysql\_num\_rows(result));

for(unsigned int i = 0; row = mysql\_fetch\_row(result), i < mysql\_num\_rows(result); i++)

{

sRows[i] = (char\*\*)malloc(sizeof(char\*) \* mysql\_num\_fields(result));

for(unsigned int j = 0; j < mysql\_num\_fields(result); j++)

{

sRows[i][j] = (char\*)malloc(sizeof(char) \* strlen(row[j]));

sRows[i][j] = row[j];

}

}

printf("\n\n\t———————————————————————————————————————————————————————————————\n", SetSeparatorColor());

printf("\t %15s | ", "Другие препараты", SetHeaderTextColor());

printf("%s\n", sRows[iKey + iPage \* ITEMS\_PER\_PAGE][0], SetStandardTextColor());

printf("\t———————————————————————————————————————————————————————————————\n", SetSeparatorColor());

printf("\t %15s | ", "Побочные эффекты", SetHeaderTextColor());

printf("%s\n", sRows[iKey + iPage \* ITEMS\_PER\_PAGE][1], SetStandardTextColor());

printf("\t———————————————————————————————————————————————————————————————\n", SetSeparatorColor());

printf("\t %15s | ", "Для детей", SetHeaderTextColor());

printf("%s\n", sRows[iKey + iPage \* ITEMS\_PER\_PAGE][2], SetStandardTextColor());

printf("\t———————————————————————————————————————————————————————————————\n", SetSeparatorColor());

printf("\t %15s | ", "Для беременных", SetHeaderTextColor());

printf("%s\n", sRows[iKey + iPage \* ITEMS\_PER\_PAGE][3], SetStandardTextColor());

printf("\t———————————————————————————————————————————————————————————————\n", SetSeparatorColor());

mysql\_data\_seek(result, 0);

\_getch();

menu\_create(mContraindications, iPage);

}

}

}

/\* Хендлер меню изготовителей \*/

void mh\_manufacturers(int iPage, int iKeys = 0)

{

int iKey = get\_key(iKeys);

switch(iKey)

{

case MENU\_KEY\_8:

{

menu\_create(mManufacturers, --iPage);

break;

}

case MENU\_KEY\_9:

{

menu\_create(mManufacturers, ++iPage);

break;

}

case MENU\_KEY\_0:

{

m\_main();

break;

}

default:

{

iKey--;

MYSQL\_ROW \*sRows = (char\*\*\*)malloc(sizeof(char\*\*) \* mysql\_num\_rows(result));

for(unsigned int i = 0; row = mysql\_fetch\_row(result), i < mysql\_num\_rows(result); i++)

{

sRows[i] = (char\*\*)malloc(sizeof(char\*) \* mysql\_num\_fields(result));

for(unsigned int j = 0; j < mysql\_num\_fields(result); j++)

{

sRows[i][j] = (char\*)malloc(sizeof(char) \* strlen(row[j]));

sRows[i][j] = row[j];

}

}

printf("\n\n\t———————————————————————————————————————————————————————————————\n", SetSeparatorColor());

printf("\t %7s | ", "Компания", SetHeaderTextColor());

printf("%s\n", sRows[iKey + iPage \* ITEMS\_PER\_PAGE][0], SetStandardTextColor());

printf("\t———————————————————————————————————————————————————————————————\n", SetSeparatorColor());

printf("\t %7s | ", "Страна", SetHeaderTextColor());

printf("%s\n", sRows[iKey + iPage \* ITEMS\_PER\_PAGE][1], SetStandardTextColor());

printf("\t———————————————————————————————————————————————————————————————\n", SetSeparatorColor());

mysql\_data\_seek(result, 0);

\_getch();

menu\_create(mManufacturers, iPage);

}

}

}

/\* Ф-я отлова нажатия цифренных клавиш, которые указаны в iKeys \*/

int get\_key(int iKeys)

{

for(;;) // Бесконечный цикл. Эквивалентно while(1)

{

char cKey = \_getch() - 48; // Коды цифренных клавиш: 48-57. Чтобы получить настоящее число клавиши, отнимаем 48 (49-48=1 -- нажали клавишу 1)

if(key\_valid(cKey, iKeys)) // Разрешено ли нажатие этой клавиши? Т.е. находится ли этот бит в бит. сумме?

return cKey; // Если "да", то возвращаем цифру нажатой клавиши

}

}

int read\_medications()

{

char sDB[32];

GetPrivateProfileStringA("mysql\_init", "db", NULL, sDB, charsmax(sDB), ".\\mysql.ini");

if(mysql\_select\_db(&mysql, sDB)) // Выбираем нужную нам БД

{

printf("\n\n");

error\_db(); // Если во время выбора БД произошла ошибка, то вызываем соответствующую функцию

}

if(mysql\_query(&mysql, "SELECT name, applying, dosing\_regimen FROM medications")) // Посылаем запрос (просим поля name, applying, dosing\_regimen с таблицы medications

{

printf("\n\n");

error\_db();

}

if(!(result = mysql\_store\_result(&mysql))) // Получаем результирующий набор

{

printf("\n\n");

error\_db();

}

return NULL;

}

int read\_diagnosis()

{

char sDB[32];

GetPrivateProfileStringA("mysql\_init", "db", NULL, sDB, charsmax(sDB), ".\\mysql.ini");

if(mysql\_select\_db(&mysql, sDB))

{

printf("\n\n");

error\_db();

}

if(mysql\_query(&mysql, "SELECT name FROM diagnosis"))

{

printf("\n\n");

error\_db();

}

if(!(result = mysql\_store\_result(&mysql)))

{

printf("\n\n");

error\_db();

}

return NULL;

}

int read\_compositions()

{

char sDB[32];

GetPrivateProfileStringA("mysql\_init", "db", NULL, sDB, charsmax(sDB), ".\\mysql.ini");

if(mysql\_select\_db(&mysql, sDB))

{

printf("\n\n");

error\_db();

}

if(mysql\_query(&mysql, "SELECT components, grp, issue\_form FROM consist"))

{

printf("\n\n");

error\_db();

}

if(!(result = mysql\_store\_result(&mysql)))

{

printf("\n\n");

error\_db();

}

return NULL;

}

int read\_cntraindications()

{

char sDB[32];

GetPrivateProfileStringA("mysql\_init", "db", NULL, sDB, charsmax(sDB), ".\\mysql.ini");

if(mysql\_select\_db(&mysql, sDB))

{

printf("\n\n");

error\_db();

}

if(mysql\_query(&mysql, "SELECT interactions, side\_effects, 4children, 4pregnant FROM сontraindications"))

{

printf("\n\n");

error\_db();

}

if(!(result = mysql\_store\_result(&mysql)))

{

printf("\n\n");

error\_db();

}

return NULL;

}

int read\_manufacturers()

{

char sDB[32];

GetPrivateProfileStringA("mysql\_init", "db", NULL, sDB, charsmax(sDB), ".\\mysql.ini");

if(mysql\_select\_db(&mysql, sDB))

{

printf("\n\n");

error\_db();

}

if(mysql\_query(&mysql, "SELECT company, country FROM manufacturer"))

{

printf("\n\n");

error\_db();

}

if(!(result = mysql\_store\_result(&mysql)))

{

printf("\n\n");

error\_db();

}

return NULL;

}

void error\_db()

{

printf("\* Ошибка взаимодействия с базой данных:\n", SetErrorTextColor());

printf("\* \"%s\"\n\n", mysql\_error(&mysql)); // Ф-я возвращает строку с ошибкой

printf("\* Восстановите соединение с сервером MySQL,\n", SetStandardTextColor());

printf("\* программа автоматически продолжит работу...\n");

while(!connect\_to\_db());

}

## Файл «inc.h»

#include <stdio.h>

#include <conio.h>

#include <locale.h>

#include <windows.h>

#include "prototypes.h" // Файл с прототипами используемых функций

#include "def.h" // Файл с константами и макросами

#include "mysql.h" // Собственно...

## Файл «prototypes.h»

int get\_key(int);

bool connect\_to\_db();

void error\_db();

void m\_main(void);

void mh\_main(int);

void menu\_create(int, int);

void mh\_medications(int, int);

void mh\_diagnosis(int, int);

void mh\_compositions(int, int);

void mh\_contraindications(int, int);

void mh\_manufacturers(int, int);

int read\_medications();

int read\_diagnosis();

int read\_compositions();

int read\_cntraindications();

int read\_manufacturers();

## Файл (конфігураційний) «mysql.ini»

[mysql\_init]

host = "localhost"

user = "root"

pass = ""

db = "drugs"

port = "3306"

## Файл «def.h»

#define KEYCODE\_ESC 27 // Код клавиши Escape

char sMenuNames[][32] = // Массив с названиями меню

{

"Главное меню",

"Лекарства",

"Диагнозы",

"Составы лекарств",

"Противопоказания",

"Изготовители",

};

#define MENU\_KEY\_1 1 // Константы клавиш

#define MENU\_KEY\_2 2

#define MENU\_KEY\_3 3

#define MENU\_KEY\_4 4

#define MENU\_KEY\_5 5

#define MENU\_KEY\_6 6

#define MENU\_KEY\_7 7

#define MENU\_KEY\_8 8

#define MENU\_KEY\_9 9

#define MENU\_KEY\_0 0

#define ITEMS\_PER\_PAGE 7 // Число пунктов на 1 страницу

#define add\_key(bit, bit\_summ) (bit\_summ |= (1 << bit)) // Добавляет бит (bit) к бит. сумме (bit\_summ)

#define sub\_key(bit, bit\_summ) (bit\_summ &= ~(1 << bit)) // Удаляет бит (bit) из бит. суммы (bit\_summ)

#define key\_valid(bit, bit\_summ) (bit\_summ & (1 << bit)) // Проверяет, есть ли бит (bit) в бит. сумме (bit\_summ)

enum Menus // Перечисление идентификаторов меню

{

mMedications = 0,

mDiagnosis,

mCompositions,

mContraindications,

mManufacturers

};

enum ConsoleColor // Перечисление цветов

{

Black = 0,

Blue,

Green,

Cyan,

Red,

Magenta,

Brown,

LightGray,

DarkGray,

LightBlue,

LightGreen,

LightCyan,

LightRed,

LightMagenta,

Yellow,

White

};

#define SetErrorTextColor() SetConsoleTextAttribute(hConsoleHandle, LightRed | Black) // Для ошибок

#define SetNumberTextColor() SetConsoleTextAttribute(hConsoleHandle, LightRed | Black) // Для номеров пунктов

#define SetHeaderTextColor() SetConsoleTextAttribute(hConsoleHandle, LightCyan | Black) // Для заголовка

#define SetStandardTextColor() SetConsoleTextAttribute(hConsoleHandle, LightGray | Black) // Стандартный цвет

#define SetDisableTextColor() SetConsoleTextAttribute(hConsoleHandle, DarkGray | Black) // Тёмносерый цвет

#define SetSeparatorColor() SetConsoleTextAttribute(hConsoleHandle, Magenta | Black) // Фиолетовый для сепаратора (разделителя)

/\* Получаем размерность массива с неявной (опущенной) размерностью \*/

template<typename T, size\_t n>

inline size\_t arraySize(const T (&arr)[n])

{

return n;

}

/\* Получаем размер массива с учетом выделения места под нуль-терминальный символ \*/

#define charsmax(arr) (arraySize(arr) - 1)

## Файл-скрипт БД MySQL

/\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET NAMES utf8 \*/;

/\*!50503 SET NAMES utf8mb4 \*/;

/\*!40014 SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0 \*/;

/\*!40101 SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='NO\_AUTO\_VALUE\_ON\_ZERO' \*/;

-- Дамп структуры базы данных drugs

CREATE DATABASE IF NOT EXISTS `drugs` /\*!40100 DEFAULT CHARACTER SET utf8 \*/;

USE `drugs`;

-- Дамп структуры для таблица drugs.consist

CREATE TABLE IF NOT EXISTS `consist` (

`id` int(10) unsigned NOT NULL AUTO\_INCREMENT COMMENT 'Код состава',

`issue\_form` varchar(128) NOT NULL COMMENT 'Форма выпуска',

`group` varchar(128) NOT NULL COMMENT 'Группа',

`components` varchar(128) NOT NULL COMMENT 'Составляющие',

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Дамп данных таблицы drugs.consist: ~0 rows (приблизительно)

/\*!40000 ALTER TABLE `consist` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `consist` ENABLE KEYS \*/;

-- Дамп структуры для таблица drugs.diagnosis

CREATE TABLE IF NOT EXISTS `diagnosis` (

`id` int(10) unsigned NOT NULL AUTO\_INCREMENT COMMENT 'Код диагноза',

`name` varchar(255) NOT NULL COMMENT 'Название диагноза',

`diagnosis\_form` int(10) unsigned NOT NULL COMMENT 'Ссылка...',

PRIMARY KEY (`id`),

KEY `diagnosis\_form` (`diagnosis\_form`),

CONSTRAINT `FK\_diagnosis\_diagnosis\_form` FOREIGN KEY (`diagnosis\_form`) REFERENCES `diagnosis\_form` (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Дамп данных таблицы drugs.diagnosis: ~0 rows (приблизительно)

/\*!40000 ALTER TABLE `diagnosis` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `diagnosis` ENABLE KEYS \*/;

-- Дамп структуры для таблица drugs.diagnosis\_form

CREATE TABLE IF NOT EXISTS `diagnosis\_form` (

`id` int(10) unsigned NOT NULL AUTO\_INCREMENT COMMENT 'Код формы диагноза',

`name` varchar(255) NOT NULL COMMENT 'Название формы диагноза',

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 ROW\_FORMAT=DYNAMIC;

-- Дамп данных таблицы drugs.diagnosis\_form: ~0 rows (приблизительно)

/\*!40000 ALTER TABLE `diagnosis\_form` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `diagnosis\_form` ENABLE KEYS \*/;

-- Дамп структуры для таблица drugs.manufacturer

CREATE TABLE IF NOT EXISTS `manufacturer` (

`id` int(10) unsigned NOT NULL AUTO\_INCREMENT COMMENT 'Код производителя',

`company` varchar(45) NOT NULL COMMENT 'Фирма производитель',

`country` varchar(45) NOT NULL COMMENT 'Страна производитель',

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Дамп данных таблицы drugs.manufacturer: ~0 rows (приблизительно)

/\*!40000 ALTER TABLE `manufacturer` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `manufacturer` ENABLE KEYS \*/;

-- Дамп структуры для таблица drugs.medications

CREATE TABLE IF NOT EXISTS `medications` (

`id` int(10) unsigned NOT NULL AUTO\_INCREMENT COMMENT 'Код препарата',

`name` varchar(500) NOT NULL COMMENT 'Название препарата',

`applying` varchar(500) NOT NULL COMMENT 'Применение',

`dosing\_regimen` varchar(500) NOT NULL COMMENT 'Режим дозирования',

`manufact\_id` int(10) unsigned NOT NULL COMMENT 'Ссылка на manufacturer.id',

`contr\_id` int(10) unsigned NOT NULL COMMENT 'Ссылка на contraindications.id',

`consist\_id` int(10) unsigned NOT NULL COMMENT 'Ссылка на consist.id',

PRIMARY KEY (`id`),

KEY `manufact\_id` (`manufact\_id`),

KEY `contr\_id` (`contr\_id`),

KEY `consist\_id` (`consist\_id`),

CONSTRAINT `FK\_medications\_consist` FOREIGN KEY (`consist\_id`) REFERENCES `consist` (`id`),

CONSTRAINT `FK\_medications\_manufacturer` FOREIGN KEY (`manufact\_id`) REFERENCES `manufacturer` (`id`),

CONSTRAINT `FK\_medications\_сontraindications` FOREIGN KEY (`contr\_id`) REFERENCES `сontraindications` (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Дамп данных таблицы drugs.medications: ~0 rows (приблизительно)

/\*!40000 ALTER TABLE `medications` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `medications` ENABLE KEYS \*/;

-- Дамп структуры для таблица drugs.сontraindications

CREATE TABLE IF NOT EXISTS `сontraindications` (

`id` int(10) unsigned NOT NULL AUTO\_INCREMENT COMMENT 'Код противопоказания',

`interactions` varchar(500) NOT NULL COMMENT 'Взаимодействие с препаратами',

`side\_effects` varchar(500) NOT NULL COMMENT 'Побочные действия',

`4children` varchar(500) NOT NULL COMMENT 'Для детей',

`4pregnant` varchar(500) NOT NULL COMMENT 'Для беременных',

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Дамп данных таблицы drugs.сontraindications: ~0 rows (приблизительно)

/\*!40000 ALTER TABLE `сontraindications` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `сontraindications` ENABLE KEYS \*/;

/\*!40101 SET SQL\_MODE=IFNULL(@OLD\_SQL\_MODE, '') \*/;

/\*!40014 SET FOREIGN\_KEY\_CHECKS=IF(@OLD\_FOREIGN\_KEY\_CHECKS IS NULL, 1, @OLD\_FOREIGN\_KEY\_CHECKS) \*/;

/\*!40101 SET CHARACTER\_SET\_CLIENT=@OLD\_CHARACTER\_SET\_CLIENT \*/;

## Скріншоти роботи програми

|  |
| --- |
|  |
| Мал. №1. Головне меню |
|  |
| Мал. №2. Список ліків |

|  |
| --- |
|  |
| Мал. №3. Список диагнозів |
|  |
| Мал. №4. Склад ліків |

|  |
| --- |
|  |
| Мал. №5. Протипоказання |
|  |
| Мал. №6. Виробники |

|  |
| --- |
|  |
| Мал. №7. Приклад звіту по запиту ліків «Ібупрофен» |

# Висновок

Досліджування показало, що розвиток передачі даних за допомогою баз даних йде повним ходом, і з кожним роком ці бази даних стають все більшi й більшi. Але працювати з БД MySQL без, так званої, «оболонки» дуже і дуже не зручно, тому використання спеціалізованих програмних засобів, наприклад, як у цьому дипломному проекту, або універсальних MySQL менеджерів, наприклад, HeidiSQL Portable конче необхідне.

Використання мови програмування С++ для роботи з базами даних дуже трудомісткий процес, тому що дана мова сама по собі дуже складна і реалізація такого завдання була б набагато легше вирішена за допомогою, наприклад, об'єктно-орієнтованої мови С# з її Windows Forms, яка в свою чергу вже містить стандартні бібліотеки для обробки і відправки запитів серверу MySQL.

Я вважаю, що цей дипломний проект зіграв важливу роль, як досвід в програмуванні на мові С++, тим паче з такою досить складною зв'язкою — база даних MySQL.