Содержание

Введение..................................................................................................5

1. Анализ технического задания..........................................................7
2. Разработка моделей данных.............................................................9
   1. Концептуальная модель
   2. Логическая модель
   3. Физическая модель
3. Разработка и реализация ИС..........................................................19

3.1

3.2

Заключение...........................................................................................23

Список литературы..............................................................................24

Приложение..........................................................................................25

# Введение

Современные аптеки играют ключевую роль в системе здравоохранения, являясь посредниками между производителями и потребителями лекарств и различных медицинских приспособлений.. Эффективное управление аптечными запасами и услугами требует применения современных информационных технологий. Для этого необходима надежная и функциональная база данных, которая позволит оптимизировать процессы учета, хранения и распространения медикаментов.

В данной курсовой работе рассматривается разработка базы данных для аптеки, которая будет обеспечивать автоматизацию учета товаров, поставщиков, сотрудников а также взаимодействие с клиентами. Основное внимание уделяется структуре базы данных, её проектированию, что позволит не только упростить повседневные операции, но и повысить качество обслуживания клиентов.

Целью курсовой работы является создание информационной системы для добавления, удаления, изменения и поиска записей в базе данных.

Задачи, работающие для этой цели:

1. Проектирование концептуальной, логической и физической моделей базы данных.

2. Реализация физической модели в одной из современных СУБД.

3. Подключение БД к программе с помощью средств среды разработки.

4. Написание обработчиков событий и функций для добавления, удаления, изменения и поиска записей в БД.

5. Тестирование и редактирование

# Анализ технического задания

Автоматизированная информационная система (АИС) представляет собой комплекс взаимосвязанных программных и аппаратных средств, ориентированных на сбор, обработку, хранение и передачу информации внутри организации.

В автоматизированных информационных системах за хранение любой информации отвечают:

1. На физическом уровне:

а. внешние накопители;

б. встроенные устройства памяти (RAM);

в. массивы дисков;

2. На программном уровне:

а. СУБД;

б. файловая система ОС;

в. системы хранения мультимедиа, документов и т. д.

Информационная система (ИС) в данном контексте представляет собой программный комплекс, который обеспечивает сбор, обработку, хранение и предоставление доступа к данным о персонале, клиентах, поставщиках и лекарствах. Наше приложение будет являться частью этой ИС, предоставляя возможности по удобному взаимодействию с данными аптеки***.*** Оно позволит эффективно управлять организацией, автоматизировать рутинные задачи и предоставит инструменты для анализа данных, содействуя более эффективному принятию решений по организации аптечного дела.

В данной курсовой стоит задача создания АИС для аптеки. Для выполнения данной задачи требуется СУБД и среда разработки для создания графического интерфейса и функционирования методов по работе с БД

***Добавить информацию об аналогах.***

Для правильного функционирования следует учесть следующие критерии:

-Наличие данных о сотрудниках, лекарствах, поставщиках и продажах.

-Возможность добавления, редактирования, поиска и удаления записей

-Наличие графического интерфейса, отображающего элементы базы данных

Согласно выдвинутым требованиям к приложению база данных должна содержат в себе несколько таблиц, связанных между собой. Для разработки базы данных была выбрана среда SQLite, которая является СУБД реляционного вида.

К преимуществам SQLite относится:

-Простота в использовании - SQLite не требует установки отдельного сервера, так что его легко интегрировать в проект.

-Быстрая реализация - Благодаря своей простоте и легкости использования, SQLite позволяет быстро начать работу над проектом, минимизируя время разработки.

-Кроссплатформенность - SQLite поддерживает множество платформ, что обеспечивает кроссплатформенность приложений.

-Широкий круг поддерживаемых данных - SQLite поддерживает разнообразные типы данных, включая текстовые, числовые, временные и бинарные данные.

# Разработка моделей данных

Разработка моделей данных представляет собой процесс проектирования структуры данных, которая будет использоваться в приложении для хранения и организации информации в базе данных (БД). Эта модель определяет, как данные будут представлены, связаны между собой и как они будут доступны для приложения.

Роль разработки моделей данных в проектировании приложения администрирования БД невозможно переоценить. Вот несколько ключевых аспектов:

Определение структуры БД: Разработка модели данных включает в себя определение сущностей (таких как таблицы) и их атрибутов. Определение ключевых полей и связей между таблицами играет критическую роль в создании эффективной и структурированной БД.

Связи и нормализация: Разработка моделей данных помогает определить связи между таблицами, что важно для поддержания целостности данных. Применение нормализации данных (разделение таблиц для предотвращения избыточности) также является частью этого процесса.

Оптимизация запросов: Эффективные модели данных могут значительно повлиять на производительность запросов к базе данных. Правильное проектирование позволяет создать оптимизированные структуры данных, ускоряющие выполнение запросов.

## Концептуальная модель

Создание концептуальной модели данных - это первый и абстрактный этап проектирования базы данных, который ориентирован на выделение основных сущностей и их взаимосвязей без углубленного внимания к техническим деталям. Необходимо начать с понимания требований и целей, которые база данных должна поддерживать. Важно понять, какие данные необходимы для проходящих в ней процессов и как они связаны между собой. Необходимо определить для каждой сущности её характеристики, или атрибуты.

Атрибуты - это конкретные данные, которые мы хотим сохранить для каждой сущности. Связи отражают, как сущности взаимодействуют друг с другом. Например, заказ связан с клиентом и лекарствами. Определение типа связей (один-к-одному, один-ко-многим, многие-ко-многим) важно для правильного построения модели.

Будем стремится к минимализации избыточности данных, избегая повторения между сущностями для сохранения целостности и согласованности данных

Визуализация концептуальной модели происходит чаще всего с использованием диаграммы сущность-связь (ER-диаграммы). Это графическое представление сущностей, их атрибутов и связей между ними.

Так как аптека в первую очередь работает с различными лекарствами, то следовательно в нашей базе данных должна быть сущность лекарства с атрибутами: цена, единица измерения, доступное количество, название, группа, продажа по рецепту и поставщик, который в свою очередь, то же является сущностью с атрибутами: название, ФИО агента, номер телефона. По мимо этого выделим отдельную сущность покупка, с атрибутом количество, которая будет содержать купленные лекарства, данная сущность нужна для соединения между собой покупателя и лекарств в чеке. Так же в нашей ИС будут содержатся сведения о сотрудниках аптеки, которые там работают с атрибутами: ФИО, должность, паспорт, дата рождения, дата найма, телефон. график работы и зарплата. Сведения о покупателях с атрибутами ФИО, номер телефона. а так же сущность Чек с атрибутом дата продажи. И наконец, сущность адресс, которая будет содержать в себе атрибуты город, улица, дом, квартира. Назначение данной сущности собрать все адреса сущностей сотрудник, покупатель и поставщик.

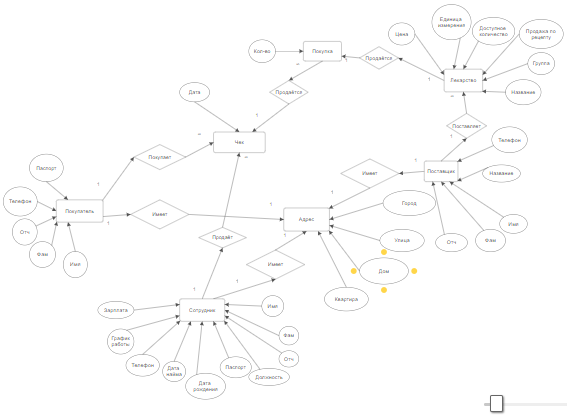


Рисунок 1: Концептуальная модель данных

Таким способом, в нашей базе данных будут следующие отношения между сущностями: Поставщик поставляет лекарства, которые продавец продаёт покупателю, а покупатель, соответственно, их покупает, всё это отраженно в сущности Чек с соотвествующими сложными атрибутами: покупатель, продавец и лекарства.

2.2 Логическая модель

Логическая модель базы данных представляет собой более конкретное и техническое описание того, как данные будут организованы и храниться в базе данных. Каждая сущность в концептуальной модели становится таблицей в логической модели. Каждая запись в таблице представляет собой конкретный экземпляр сущности:

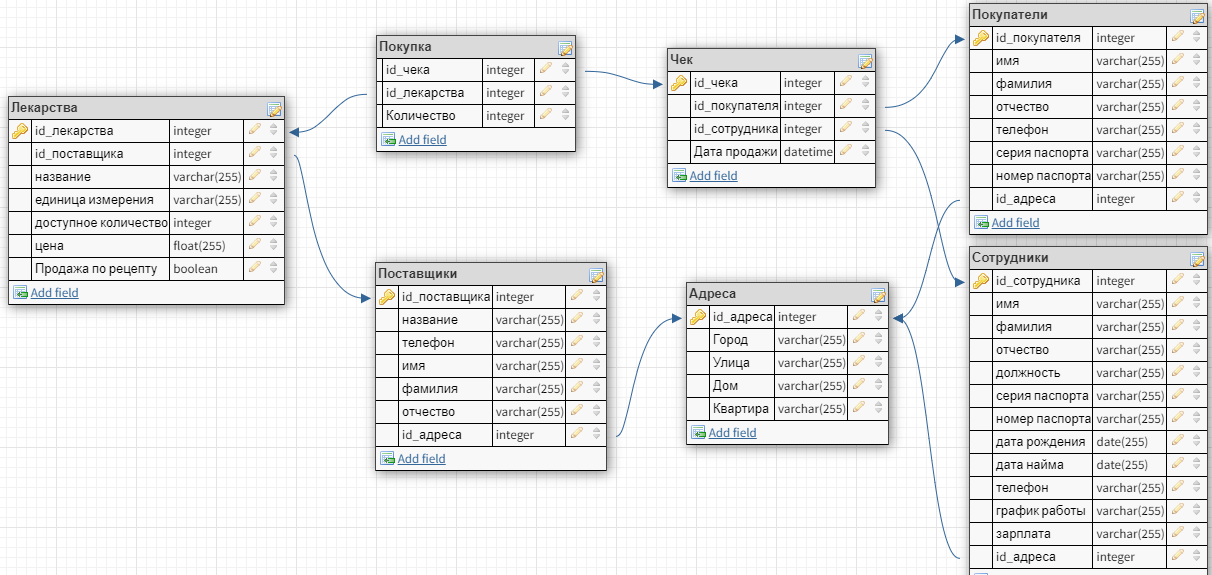


Рисунок 2: Логическая модель данных

Логическая модель базы данных является промежуточным звеном между концептуальной моделью и физической реализацией базы данных. Она предоставляет детальное понимание структуры данных, не зависящее от конкретной СУБД, что позволяет легче адаптировать модель под разные технологические решения.

3.3 Физическая модель

Физическая модель базы данных представляет собой конкретную реализацию структуры данных на уровне конкретной СУБД. Эта модель включает в себя информацию о том, как данные будут храниться, какие индексы будут использованы, и другие технические аспекты:

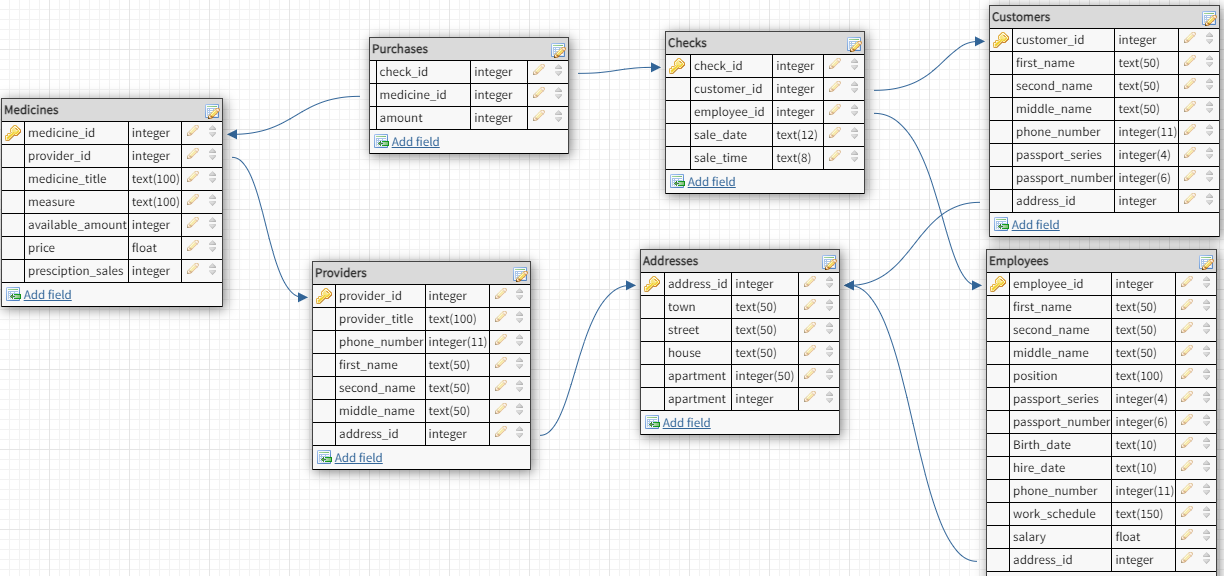


Рисунок 3: физическая модель данных

# Разработка ИС

Для эффективной разработки программы, взаимодействующей с базой данных, будем применять объектно-ориентированный подход с использованием языка программирования C# и графического модуля windows forms.

В контексте создания графического интерфейса приложения, C# предлагает множество преимуществ. ***Вставить сюда название библиотеки или средства для создания оконных приложений.***

Основной фокус при разработке должен быть направлен на создание удобного и интуитивно понятного интерфейса. C# обеспечивает простоту и гибкость в этом плане, позволяя легко реализовывать все необходимые элементы управления для оптимизации пользовательского опыта.

Надежная обработка данных также лежит в основе разработки. C# обеспечивает эффективную работу с данными, гарантируя точность и своевременность вывода результатов пользователю. Благодаря своей отзывчивости на действия пользователя, приложение на C# способно обеспечивать мгновенный отклик и формировать выходные данные в соответствии с запросами пользователей.

# Заключение

В результате выполнения курсовой работы была разработана программа в среде VisualStudio на языке программирования С++, которая позволяет точно находить простые множители числа. Помимо этого реализованна возможность сохранения и загрузки результатов работы программы.

В процессе программирования активно использовалась сторонняя литература, в основном, представленная в виде веб-ресурсов и электронных книг, что содействовало успешному решению возникающих задач.

Разработанная программа отвечает всем поставленным требованием, обеспечивая удобную работу без критических ошибок.

# Список литературы

1) C/C++. Программирование на языке высокого уровня / Т. А. Павловская. — СПб.: Питер, 2003. —461 с: ил.

2) Мейерс, Скотт. М45 Эффективный и современный С++: 42 рекомендации по использованию С++ 11 и С++14.: Пер. с англ. - М. : ООО "ИЛ. Вильяме", 2016. - 304 с.: ил. – Пap.

3) Эффективное использование STL [2002] Скотт Мейерс.

4) Роберт Седжвик: Алгоритмы на c++. Анализ структуры данных. Сортировка. Поиск. Алгоритмы на графах. СПб.: Питер, 2016. -687 с: ил.

5) Яцек Галовиц. C++17 STL. Стандартная библиотека шаблонов. ООО «ИЛ.Вильяме», 2018. -431 с.: ил. – Пар.

# Приложение

Ссылка на весь проект:

<https://github.com/Pacmanoidum/1-kursach/releases/tag/kursach>

Полный код программы:

string Long\_Sravn(string num1, string num2)

{

int len1 = num1.length();

int len2 = num2.length();

// сравним длинну чисел

if (len1 < len2) {

return "<";

}

else if (len1 > len2) {

return ">";

}

// сравним каждый разряд по отдельности

for (int i = 0; i < len1; i++) {

if (num1[i] < num2[i]) {

return "<";

}

else if (num1[i] > num2[i]) {

return ">";

}

}

return "=";

}

string Long\_Sum(string a, string b)

{

string c;

while (a.length() != b.length())

{

// Добавляем нули для равной длины чисел

if (a.length() < b.length())

a = "0" + a;

else b = "0" + b;

}

// Добавляем нули для возможного перехода на новый разряд

a = "0" + a; b = "0" + b;

// Складываем цифры с права налево, r - остаток от прошлого сложения, постепенно заполняем строку c

int sum, r = 0, i;

for (int i = b.length() - 1; i > -1; i--)

{

sum = a[i] - '0' + b[i] - '0' + r;

r = sum / 10;

c = char(sum % 10 + '0') + c;

}

//Удаляем лишний ноль при необходимости

c.erase(0, c.find\_first\_not\_of('0'));

return c;

}

string Long\_Mult(string a, string b)

{

int m = a.size();

int n = b.size();

string result(m + n, '0'); // создаем строку для записи результата

for (int i = m - 1; i >= 0; i--) {

int carry = 0; // число для "переноса"

for (int j = n - 1; j >= 0; j--) {

int temp = (a[i] - '0') \* (b[j] - '0') + carry + (result[i + j + 1] - '0');

carry = temp / 10; // вычисляем "перенос"

result[i + j + 1] = (temp % 10) + '0'; // запоминаем остаток от деления на 10

}

result[i] += carry; // при необходимости прибавляем "перенос" к числу в столбце

}

//проверяем результат

size\_t startPos = result.find\_first\_not\_of("0");

if (startPos != string::npos) {

return result.substr(startPos);

}

return "0";

}

string Long\_Minus(string a, string b)

{

int carry = 0;

int len1 = a.length();

int len2 = b.length();

//

\_Sravn(a, b) == "<")

{ Меняем числа местами для нормального считания модуля разности

if (Long

swap(a, b);

swap(len1, len2);

}

string result;

// Вычитаем числа справа налево

for (int i = 0; i < len2; i++) {

int digit1 = a[len1 - 1 - i] - '0';

int digit2 = b[len2 - 1 - i] - '0';

int diff = (digit1 - carry) - digit2;

if (diff < 0) {

diff += 10;

carry = 1;

}

else {

carry = 0;

}

result.push\_back(diff + '0');

}

// Вычитаем всё, что осталось

for (int i = len2; i < len1; i++) {

int digit = a[len1 - 1 - i] - '0';

int diff = (digit - carry);

if (diff < 0) {

diff += 10;

carry = 1;

}

else {

carry = 0;

}

result.push\_back(diff + '0');

}

reverse(result.begin(), result.end());

result.erase(0, result.find\_first\_not\_of('0'));

if (result.empty()) return "0";

else return result;

}

string Long\_Divide(const string dividend, const string divisor)

{

string quotient;

int dividendLength = dividend.length();

int divisorLength = divisor.length();

string currentDividend = dividend.substr(0, divisorLength); //Получаем строку для деления

int currentIndex = divisorLength;

while (currentIndex <= dividendLength) {

int currentQuotient = 0;

while (Long\_Sravn(currentDividend, divisor) == ">" || Long\_Sravn(currentDividend, divisor) == "=") {

if (currentDividend == "0")

{

currentQuotient = 0;

break;

}

currentDividend = Long\_Minus(currentDividend, divisor); //выполняем деление вычитанием числа n раз

currentQuotient++;

}

quotient += to\_string(currentQuotient);

if (currentIndex == dividendLength)

break;

if (currentDividend == "0") currentDividend.clear();

currentDividend += dividend[currentIndex]; // спиываем следующую цифру сверху

currentIndex++;

}

quotient.erase(0, quotient.find\_first\_not\_of('0'));

return quotient;

}

string Long\_Ostatok(const string dividend, const string divisor)

{

string quotient;

string remainder;

int dividendLength = dividend.length();

int divisorLength = divisor.length();

string currentDividend = dividend.substr(0, divisorLength);

int currentIndex = divisorLength;

while (currentIndex <= dividendLength) {

int currentQuotient = 0;

while (Long\_Sravn(currentDividend, divisor) == ">" || Long\_Sravn(currentDividend, divisor) == "=") {

if (currentDividend == "0")

{

currentQuotient = 0;

break;

}

currentDividend = Long\_Minus(currentDividend, divisor);

currentQuotient++;

}

quotient += to\_string(currentQuotient);

if (currentIndex == dividendLength)

break;

if (currentDividend == "0") currentDividend.clear();

currentDividend += dividend[currentIndex];

currentIndex++;

}

remainder = currentDividend;

quotient.erase(0, quotient.find\_first\_not\_of('0'));

return remainder;

}

bool Long\_Prost(string n)

{

if (n == "1") return 0;

if (n == "2") return 1;

if (n == "3") return 1;

if (n == "5") return 1;

if (n == "5") return 1;

if (Long\_Ostatok(n, "2") == "0") return 0;

if (Long\_Ostatok(n, "3") == "0") return 0;

if (Long\_Ostatok(n, "5") == "0") return 0;

if (Long\_Ostatok(n, "7") == "0") return 0;

static string delta[48] = { "2", "4", "2", "4", "6", "2", "6", "4", "2",

"4", "6", "6", "2", "6", "4", "2", "6", "4", "6", "8", "4", "2", "4", "2", "4", "8",

"6", "4", "6", "2", "4", "6", "2", "6", "6", "4", "2", "4", "6", "2", "6", "4", "2", "4", "2", "10", "2", "10" };

int idx = 0;

for (string i = "11"; Long\_Sravn(Long\_Mult(i, i), n) != ">";)

{

if (Long\_Ostatok(n, i) == "0") return 0;

i = Long\_Sum(i, delta[idx]);

idx = (idx + 1) % 48;

}

return 1;

}

string Long\_NOD(string a, string b)

{

while (b != "0")

{

string temp = b;

b = Long\_Ostatok(a, b);

a = temp;

}

return a;

}

void Long\_Sort(vector<string>& v)

{

int n = v.size();

for (int i = 0; i < n - 1; i++) {

for (int j = 0; j < n - i - 1; j++) {

if (Long\_Sravn(v[j], v[j+1]) == ">") {

swap(v[j], v[j + 1]);

}

}

}

}

string Long\_Polard(string N, string X)

{

if (N == "0") return "0";

if (N == "1") return "1";

if (N == "2") return "2";

if (N == "3") return "3";

if (N == "4") return "2";

if (N == "5") return "5";

if (N == "7") return "7";

string x = Long\_Minus(N, X);

string y = "1"; string i = "0"; string stage = "2";

string nod = (Long\_NOD(N, Long\_Minus(x, y)));

while (nod == "1")

{

if (i == "250")

{

if (Long\_Prost(N) == true)

return N;

}

if (i == stage)

{

y = x;

stage = Long\_Mult(stage, "2");

}

x = Long\_Ostatok(Long\_Sum(Long\_Mult(x, x), "1"), N); //(x \* x + 1) % n

i = Long\_Sum(i, "1");

nod = (Long\_NOD(N, Long\_Minus(x, y)));

}

if (Long\_Prost(nod) != 1)

{

if (Long\_Sravn(nod, N) == "=")

return Long\_Polard(N, Long\_Sum(X, "1"));

else return Long\_Polard(nod, X);

}

else return nod;

}

void CКурсачПраймDlg::OnBnClickedOk()

{

// TODO: добавьте свой код обработчика уведомлений

CDialogEx::OnOK();

}

void CКурсачПраймDlg::OnBnClickedButton1()

{

// TODO: добавьте свой код обработчика уведомлений

auto begin = std::chrono::steady\_clock::now();

vector<string> v;

CString S;

GetDlgItemText(IDC\_EDIT1, S);

string N = CT2A(S);

try

{

if (N.empty()) throw runtime\_error("Строка пуста!");

for (char i : N)

{

if (!isdigit(i))

throw runtime\_error("Число введено некорректно!");

}

}

catch (exception& e)

{

AfxMessageBox(CString(e.what()));

return;

}

string buf;

if (N == "1") v.push\_back("1");

if (N == "0") v.push\_back("0");

else

{

while (N != "1")

{

buf = Long\_Polard(N, "2");

v.push\_back(buf);

N = Long\_Divide(N, buf);

}

}

Long\_Sort(v);

v.erase(unique(v.begin(), v.end()), v.end());

ostringstream oss;

for (auto i : v)

{

oss << i << ", ";

}

string str = oss.str();

str.pop\_back(); str.pop\_back(); str.push\_back('.');

CString cstr(str.c\_str());

SetDlgItemText(IDC\_EDIT2, cstr);

auto end = std::chrono::steady\_clock::now();

auto ms = std::chrono::duration\_cast<std::chrono::milliseconds>(end - begin);

auto s = std::chrono::duration\_cast<std::chrono::seconds>(end - begin);

ostringstream ss;

ss << s.count() << " seconds, " << ms.count() % 1000 << " milliseconds.";

str = ss.str();

CString cstr1(str.c\_str());

SetDlgItemText(IDC\_EDIT3, cstr1);

MessageBox(L"Число декомпозированно");

}

void CКурсачПраймDlg::Load\_Button()

{

// TODO: добавьте свой код обработчика команд

CFileDialog fileDialog(TRUE, 0, L"\*.txt"); //объект класса выбора файла

int result = fileDialog.DoModal(); //запустить диалоговое окно

if (result == IDOK) //если файл выбран

{

ifstream file(fileDialog.GetPathName());

if (file.is\_open())

{

string str;

string line;

while (getline(file, line))

{

str += line;

}

CString cstr(str.c\_str());

SetDlgItemText(IDC\_EDIT1, cstr);

}

else AfxMessageBox(L"No file!");

}

}

void CКурсачПраймDlg::Save\_Button()

{

// TODO: добавьте свой код обработчика команд

CFileDialog fileDialog(TRUE, 0, L"\*.txt"); //объект класса выбора файла

int result = fileDialog.DoModal(); //запустить диалоговое окно

if (result == IDOK) //если файл выбран

{

CString str1; CString str2;

GetDlgItemText(IDC\_EDIT1, str1); GetDlgItemText(IDC\_EDIT2, str2);

string s1 = CT2A(str1); string s2 = CT2A(str2);

ostringstream oss;

oss << s1 << ": " << s2;

string st = oss.str();

ofstream file(fileDialog.GetPathName(), ios::app);

if (file.is\_open())

{

file << st << endl;

file.close();

}

else AfxMessageBox(L"No file!");

}

}