

**ОТЧЕТ**

**ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ №1**

|  |  |  |
| --- | --- | --- |
| Савин Даниил Александрович | | |
| *(Ф.И.О. обучающегося)* | | |
| 02.03.02 Фундаментальная информатика и информационные технологии, 51 Разработка программного обеспечения | | |
| *(направление подготовки (специальность), направленность (профиль))* | | |
|  | | |
| Место прохождения практики | | Кировский ЦГМС, |
|  | *(наименование организации)* | |
| отдел техники и связи | | |
| *(наименование структурного подразделения организации)* | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Итоговая оценка: |  | | | | |
| Руководитель  практики от университета |  |  |  |  | И.Н. Фищева |
|  | *(дата)* |  | *(подпись)* |  | *(Ф.И.О.)* |

Киров, 2021 г.

# СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 4](#_Toc9940173)

[ХАРАКТЕРИСТИКА МЕСТА ПРОХОЖДЕНИЯ ПРАКТИКИ 5](#_Toc9940174)

[ОТЧЕТ О ВЫПОЛНЕНИИ ИНДИВИДУАЛЬНОГО ЗАДАНИЯ 6](#_Toc9940175)

[ЗАКЛЮЧЕНИЕ 13](#_Toc9940182)

[БИБЛИОГРАФИЧЕСКИЙ СПИСОК 14](#_Toc9940183)

[ПРИЛОЖЕНИЕ 15](#_Toc9940184)

# ВВЕДЕНИЕ

Производственная практика проходила в Кировском ЦГМС, отдел техники и связи с 21.06.2021 г. по 4.07.2021 г., количество недель: 2.

Целью производственной практики является получение профессиональных умений и компетенций в сфере профессиональной деятельности.

В ходе прохождения практики необходимо:

* сформировать полное представление о своей профессии;
* выполнить обязанности на первичных должностях в области применения современных математических методов и информационных технологий;
* получить опыт применения математического аппарата и современных информационных технологий для анализа и решения конкретных производственных задач
* закрепить практические навыки по применению пакетов прикладных программ, разработке прикладного и системного программного обеспечения;
* получить навыки самостоятельного или коллективного решения конкретных профессиональных задач;
* приобрести опыт самостоятельной профессиональной деятельности;
* выполнить индивидуальное задание.

# ХАРАКТЕРИСТИКА МЕСТА ПРОХОЖДЕНИЯ ПРАКТИКИ

Кировской центр по гидрометеорологии и мониторингу окружающей среды (Кировский ЦГМС), является организацией Федеральной службы по гидрометеорологии и мониторингу окружающей среды (далее — Росгидромет), созданной Указом Президента Российской Федерации от 22.09.1988 г. № 1142, входит в состав Верхне-Волжского управления по гидрометеорологии и мониторингу окружающей среды.

Основной вид деятельности организации (отдела)  сводится к повышению качества информации за счет интенсификации использования техники, внедрения новых систем связи, программного обеспечения, повышения квалификации сотрудников.

Описание рабочего места (используемое аппаратное обеспечение):

* Персональный компьютер:

Windows 10,

Intel(R) Pentium(R) CPU G4600 @ 3.60GHz 3.60 GHz,

Оперативная память — 8,00 ГБ.

Программное обеспечение, используемое в работе:

* Visual Studio 2019

С рабочих мест обеспечен доступ к Internet, ЭБС ФГБОУ ВО «ВятГУ».

# ОТЧЕТ О ВЫПОЛНЕНИИ ИНДИВИДУАЛЬНОГО ЗАДАНИЯ

# Постановка задания

Разработка методов, алгоритмов и программного обеспечения, направленных на разработку БД прогноза заморозков.

# Цель, задачи, которые необходимо решить для достижения поставленной цели

1. Пройти инструктаж по ознакомлению с правилами внутреннего трудового распорядка, охраны труда, техники безопасности, противопожарной безопасности, санитарно-эпидемиологическими правилами и гигиеническими нормативами, а также вводный инструктаж и инструктаж на рабочем месте.
2. Сбор информации для разработки БД с поддержкой целостности для прогноза заморозков.
3. Разработка методов, алгоритмов и программного обеспечения, направленных на разработку БД прогноза заморозков.

# Обзор имеющихся подходов к решению

SQLite — это библиотека на языке C, которая реализует небольшой, быстрый, автономный, высоконадежный, полнофункциональный механизм базы данных SQL. SQLite — это наиболее часто используемый механизм баз данных в мире. SQLite встроен во все мобильные телефоны и большинство компьютеров и входит в состав множества других приложений, которые люди используют каждый день.

# Обоснование выбора методов

Выбрал SQLite, потому что эта библиотека проста в использовании, а также обеспечивает быстрый, надежный требуемый механизм работы с базами данных.

# Описание решения

Для начала требуется создать базу данных с нужными таблицами и связями:

Создадим базу данных с двумя таблица: таблица «», в которой хранятся номер станции и ее название:

Вторая таблица «» отвечает за хранение данных с сервера — номер станции, хэш измерения, значение измерения, название измерения и дата:

В БД используется связь — один ко многим. Если удаляется значение в таблице «», то и в таблице «» удалится значение с таким же номером станции. Если обновится значение «» в таблице «», то оно обновится и в таблице «».

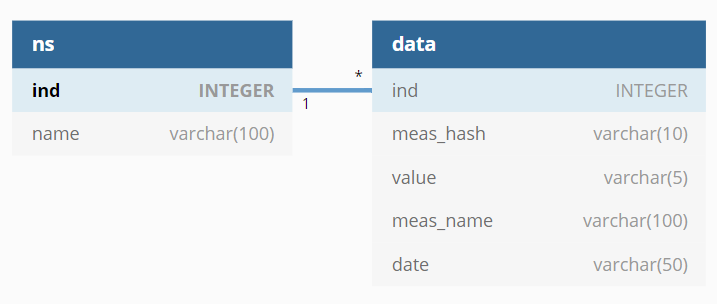


Рисунок 1 – Схема БД.

Для реализации решения сначала нужно скачать библиотеку и добавить в проект для компиляции программного кода.

В библиотеке хранятся важные методы по открытию БД: *sqlite3\_open()*, составлению запросов к БД: *sqlite3\_exec()* и закрытию БД: *sqlite3\_close().*

Функция открытия БД:

*bool open\_db(char name[])*

*{*

*rc = sqlite3\_open(name, &db);*

*if (rc)*

*{*

*fprintf(stderr, "Can't open database: %s\n", sqlite3\_errmsg(db));*

*return false;*

*}*

*else*

*{*

*fprintf(stderr, "\nOpening the database successfully.\n");*

*cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl << endl << endl;*

*}*

*return true;*

*}*

В данной функции происходит открытие БД, если ее нет, то она создастся и откроется, в обратном случае выведет ошибку и ее описание.

Процедура закрытия БД:

*void close\_db()*

*{*

*rc = sqlite3\_close(db);*

*if (rc != SQLITE\_OK)*

*{*

*fprintf(stderr, "SQL error: %s\n", zErrMsg);*

*sqlite3\_free(zErrMsg);*

*}*

*else*

*{*

*cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;*

*cout << "Closing the database successfully." << endl;*

*}*

*}*

В данной процедуре выполняется закрытие БД, в противном случае выведет ошибку и ее описание.

Также есть методы по выводу значений всех таблиц БД, указанной таблицы и обновления данных в таблицах БД:

Процедура вывода значений указанной таблицы:

*void display\_valOfTable(string table) //выводит значения указанной таблицы*

*{*

*string sql;*

*string table\_name = table;*

*if (table\_name != "")*

*{*

*sql = "SELECT \* FROM " + table\_name + ";";*

*}*

*else*

*{*

*while (table\_name == "")*

*{*

*cout << "input table name: ";*

*cin >> table\_name;*

*}*

*sql = "SELECT \* FROM " + table\_name + ";";*

*}*

*rc = sqlite3\_exec(db, sql.c\_str(), callback, (void\*)data1, &zErrMsg);*

*if (rc != SQLITE\_OK) {*

*fprintf(stderr, "SQL error: %s\n", zErrMsg);*

*sqlite3\_free(zErrMsg);*

*}*

*}*

В этой процедуре используется метод для составления запросов - *sqlite3\_exec()*, с помощью которого можно сформировать запрос на показ значений таблицы.

Процедура вывода значений всех таблиц БД:

*void display\_values\_of\_all\_tables() //выводит все таблицы*

*{*

*string sql;*

*sql = "SELECT \* FROM ns;";*

*cout << "table ns: " << endl;*

*rc = sqlite3\_exec(db, sql.c\_str(), callback, (void\*)data1, &zErrMsg);*

*cout << "table data: " << endl;*

*sql = "SELECT \* FROM data;";*

*rc = sqlite3\_exec(db, sql.c\_str(), callback, (void\*)data1, &zErrMsg);*

*if (rc != SQLITE\_OK) {*

*fprintf(stderr, "SQL error: %s\n", zErrMsg);*

*sqlite3\_free(zErrMsg);*

*}*

*}*

В этой процедуре также используется метод *sqlite3\_exec().*

И процедура, отвечающая за обновление значений в таблицах:

*void update\_db() //обновление указанной таблицы*

*{*

*string sql; string req = ""; string v = ""; string wh = ""; string table = ""; string whv = "";*

*display\_values\_of\_all\_tables();*

*while (table == "")*

*{*

*cout << "Введите название таблицы, в которой хотите изменить значение: " << endl;*

*cin >> table;*

*}*

*while (req == "")*

*{*

*cout << "Введите название переменной, значение которой хотите изменить: " << endl;*

*cin >> req;*

*}*

*while (v == "")*

*{*

*cout << "Введите значение переменной, которое хотите записать: " << endl;*

*cin >> v;*

*}*

*while (wh == "")*

*{*

*cout << "Введите название переменной, по которому можно найти изменяемую переменную: " << endl;*

*cin >> wh;*

*}*

*while (whv == "")*

*{*

*cout << "Введите значение этой переменной: " << endl;*

*cin >> whv;*

*}*

*{*

*sql = "update "+ table+" set " + req +"="+ v+" where "+wh+" = '"+whv+"'; ";*

*rc = sqlite3\_exec(db, sql.c\_str(), callback, (void\*)data1, &zErrMsg);*

*if (rc != SQLITE\_OK) {*

*fprintf(stderr, "SQL error: %s\n", zErrMsg);*

*sqlite3\_free(zErrMsg);*

*}*

*else {*

*fprintf(stdout, "\nUpdate successfully\n");*

*}*

*display\_values\_of\_all\_tables();*

*}*

*}*

**Пример работы приложения**

После запуска приложения нас встречает меню:

Изображение выглядит как текст

Автоматически созданное описание

Например, у станции изменился номер, нужно его обновить в таблицах, выбираем пункт 6.

Для удобства выводятся таблицы БД, чтобы знать точное название переменных:

Изображение выглядит как текст

Автоматически созданное описание

Нужно ввести название таблицы, вводим ns:



Далее нужно ввести название переменной, значение которой нужно изменить, вводим ind:



Далее нужно ввести новое значение переменной:



Далее программа просит указать название связанной переменной:



И значение этой переменной:



Изменение прошло успешно сразу в двух таблицах благодаря связи:

Изображение выглядит как текст

Автоматически созданное описание

Нажимаем любую клавишу, и снова появляется меню:

Изображение выглядит как текст

Автоматически созданное описание

# Итоги, заключение

Результатом разработки стало приложение, в котором налажен доступ к БД.

# ЗАКЛЮЧЕНИЕ

Производственная практика способствовала формированию представления о своей профессии, приобретению профессиональных навыков и умению в организации работы в области применения современных методов фундаментальной информатики и информационных технологий, приобретению опыта самостоятельной профессиональной деятельности путём самостоятельного решения поставленных задач.

В ходе практики ознакомился с правилами внутреннего трудового распорядка, охраны труда, техники безопасности, противопожарной безопасности, а также вводный инструктаж и инструктаж на рабочем месте. Освежил в памяти работу с базой данных, а также изучил взаимодействие с базой данных на языке C++.

Наиболее интересным было работать с БД на языке программирования С++.

Наиболее сложным было объединить проекты в один.

# БИБЛИОГРАФИЧЕСКИЙ СПИСОК

# Библиотека официальной технической документации SQLite под ОС Microsoft Windows [Электронный ресурс]. URL: https://www.sqlite.org/docs.html. **ПРИЛОЖЕНИЕ**

#include <iostream>

#include <stdio.h>

#include <stdlib.h>

#include "lib/sqlite3.h"

#include <string>

#include "windows.h"

#include <iostream>

#include <fstream>

using namespace std;

sqlite3\* db;

char\* zErrMsg = 0;

int rc;

string text="";

const char\* data1 = "Callback function called";

struct dataa

{

string ind;

string name;

string meas\_hash;

string value;

string meas\_name;

string date;

};

struct ns

{

string ind;

string name;

};

static int callback(void\* data, int argc, char\*\* argv, char\*\* azColName)

{

int i;

setlocale(0, ".65001");

string zxc;

for (i = 0; i < argc; i++) {

cout << azColName[i] << " = " << (argv[i] ? argv[i] : "NULL") << endl;

text = text + azColName[i] + " = " + (argv[i] ? argv[i] : "NULL") + "\n";

}

printf("\n");

setlocale(0, ".1251");

return 0;

}

bool open\_db(char name[])

{

rc = sqlite3\_open(name, &db);

if (rc)

{

fprintf(stderr, "Can't open database: %s\n", sqlite3\_errmsg(db));

return false;

}

else

{

fprintf(stderr, "\nOpening the database successfully.\n");

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl << endl << endl;

}

return true;

}

void insert\_db()

{

const int size = 1000;

dataa input[size];

ns inp[size];

string sql;

for (int i = 0; i < size && inp[i].ind!=""; i++)

{

sql = "INSERT INTO ns(ind, name)"\

"VALUES('" + inp[i].ind + "', '" + inp[i].name + "')"\

"INSERT INTO data (ind, meas\_hash, value, meas\_name, date)"\

"VALUES ('" + input[i].ind + "', '" + input[i].meas\_hash + "', '" + input[i].value + "', '" + input[i].meas\_name + "', '" + input[i].date + "'); ";

rc = sqlite3\_exec(db, sql.c\_str(), callback, (void\*)data1, &zErrMsg);

if (rc != SQLITE\_OK) {

fprintf(stderr, "SQL error: %s\n", zErrMsg);

sqlite3\_free(zErrMsg);

}

else {

fprintf(stdout, "Records created successfully\n");

}

}

}

void display\_valOfTable(string table) //выводит значения указанной таблицы

{

string sql;

string table\_name = table;

if (table\_name != "")

{

sql = "SELECT \* FROM " + table\_name + ";";

}

else

{

while (table\_name == "")

{

cout << "input table name: ";

cin >> table\_name;

}

sql = "SELECT \* FROM " + table\_name + ";";

}

rc = sqlite3\_exec(db, sql.c\_str(), callback, (void\*)data1, &zErrMsg);

if (rc != SQLITE\_OK) {

fprintf(stderr, "SQL error: %s\n", zErrMsg);

sqlite3\_free(zErrMsg);

}

}

void display\_tables() //выводит структуру всех таблиц

{

string sql;

sql = "select \* from sqlite\_master where type = 'table'";

rc = sqlite3\_exec(db, sql.c\_str(), callback, (void\*)data1, &zErrMsg);

if (rc != SQLITE\_OK) {

fprintf(stderr, "SQL error: %s\n", zErrMsg);

sqlite3\_free(zErrMsg);

}

}

void display\_values\_of\_all\_tables() //выводит все таблицы

{

string sql;

sql = "SELECT \* FROM ns;";

cout << "table ns: " << endl;

rc = sqlite3\_exec(db, sql.c\_str(), callback, (void\*)data1, &zErrMsg);

cout << "table data: " << endl;

sql = "SELECT \* FROM data;";

rc = sqlite3\_exec(db, sql.c\_str(), callback, (void\*)data1, &zErrMsg);

if (rc != SQLITE\_OK) {

fprintf(stderr, "SQL error: %s\n", zErrMsg);

sqlite3\_free(zErrMsg);

}

}

void update\_db() //обновление указанной таблицы

{

string sql; string req = ""; string v = ""; string wh = ""; string table = ""; string whv = "";

display\_values\_of\_all\_tables();

while (table == "")

{

cout << "Введите название таблицы, в которой хотите изменить значение: " << endl;

cin >> table;

}

while (req == "")

{

cout << "Введите название переменной, значение которой хотите изменить: " << endl;

cin >> req;

}

while (v == "")

{

cout << "Введите значение переменной, которое хотите записать: " << endl;

cin >> v;

}

while (wh == "")

{

cout << "Введите название переменной, по которому можно найти изменяемую переменную: " << endl;

cin >> wh;

}

while (whv == "")

{

cout << "Введите значение этой переменной: " << endl;

cin >> whv;

}

sql = "update "+ table+" set " + req +"="+ v+" where "+wh+" = '"+whv+"'; ";

rc = sqlite3\_exec(db, sql.c\_str(), callback, (void\*)data1, &zErrMsg);

if (rc != SQLITE\_OK) {

fprintf(stderr, "SQL error: %s\n", zErrMsg);

sqlite3\_free(zErrMsg);

}

else {

fprintf(stdout, "\nUpdate successfully\n");

}

display\_values\_of\_all\_tables();

}

void pragma()

{

rc = sqlite3\_exec(db, "pragma foreign\_keys=ON", callback, (void\*)data1, &zErrMsg);

if (rc != SQLITE\_OK)

{

fprintf(stderr, "SQL error: %s\n", zErrMsg);

sqlite3\_free(zErrMsg);

}

}

void close\_db()

{

rc = sqlite3\_close(db);

if (rc != SQLITE\_OK)

{

fprintf(stderr, "SQL error: %s\n", zErrMsg);

sqlite3\_free(zErrMsg);

}

else

{

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << "Closing the database successfully." << endl;

}

}

void saveToFile(string name)

{

std::ofstream out; // поток для записи

out.open(name); // окрываем файл для записи

if (out.is\_open())

{

out << text;

}

std::cout << "End of program" << std::endl;

}

int main()

{

setlocale(LC\_ALL, "rus");

string sql;

char name[] = "wd.db"; //название бд

int intro = -1;

string table\_name = "";

string menu = "\n\t\t\t\tМеню: "\

"\n\t|\t1 - Вывод на экран названия БД. \t|"\

"\n\t|\t2 - Вывод на экран БД. \t|"\

"\n\t|\t3 - Вывод на экран структуры таблиц из БД. \t|"\

"\n\t|\t4 - Вывод на экран определенной таблицы из БД. \t|"\

"\n\t|\t5 - Изменить название БД. \t|"\

"\n\t|\t6 - Обновление данных БД. \t|"\

"\n\t|\t7 - Добавление данных в БД. \t|"\

"\n\t|\t8 - Сохранение значений таблицы в файл. \t|"\

"\n\t|\t0 - Выход. \t|"\

;

while (intro!=0)

{

system("cls");

cout << menu << endl << ">> ";

cin >> intro;

switch (intro)

{

case 1:

system("cls");

cout << "\n\t " << name << endl;

system("pause");

break;

case 2:

system("cls");

if (open\_db(name))

{

pragma();

display\_values\_of\_all\_tables();

close\_db();

}

else

{

cout << "Ошибка открытия." << endl;

}

system("pause");

break;

case 3:

system("cls");

if (open\_db(name))

{

pragma();

display\_tables();

close\_db();

}

else

{

cout << "Ошибка открытия." << endl;

}

system("pause");

break;

case 4:

system("cls");

if (open\_db(name))

{

pragma();

cout << "Введите название таблицы: \n>> ";

cin >> table\_name;

display\_valOfTable(table\_name);

close\_db();

}

else

{

cout << "Ошибка открытия." << endl;

}

system("pause");

break;

case 5:

system("cls");

cout << "Введите новое название БД (не переименовывает БД):\n>> ";

cin >> name;

system("pause");

break;

case 6:

system("cls");

if (open\_db(name))

{

pragma();

update\_db();

close\_db();

}

else

{

cout << "Ошибка открытия." << endl;

}

system("pause");

break;

case 7:

system("cls");

if (open\_db(name))

{

pragma();

insert\_db();

close\_db();

}

else

{

cout << "Ошибка открытия." << endl;

}

system("pause");

break;

case 8:

system("cls");

if (open\_db(name))

{

pragma();

display\_values\_of\_all\_tables();

close\_db();

}

else

{

cout << "Ошибка открытия." << endl;

}

saveToFile("print.pdf");

//system("print print.txt");

system("pause");

break;

case 0:

break;

}

}

return 0;

}