МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИМЕНИ И.С. ТУРГЕНЕВА»

Кафедра информационных систем

**ОТЧЕТ**

по лабораторной работе № 5

на тему: «Простая транзитивная сеть»

по дисциплине: «Проектирование пользовательского интерфейса»

Выполнили: Кожухова О.А., Шорин В.Д.

Институт приборостроения, автоматизации и информационных технологий

Направление: 09.03.04 «Программная инженерия»

Группа: 71-ПГ

Проверили: Олькина Е.В., Лысков О.Э.

Отметка о зачете:

Дата: «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2019 г.

Орел, 2019 г

**Выполнение работы:**

#include <windows.h>

#include <string>

#include <cstdlib>

#include <iostream>

#include <vector>

#include <commctrl.h>

#include <mysql.h>

#include <mysqld\_error.h>

#include <dshow.h>

#define ID\_MENU\_TABLE 1000

#define ID\_MENU\_QUERY 1001

#define ID\_MENU\_EXIT 1002

#define ID\_EDIT\_TABLENAME 100

#define ID\_LB\_FIELDS 101

#define ID\_BTN\_OK 102

#define ID\_LV\_RESTABLE 104

#define ID\_BTN\_NEXT 105

#define ID\_CB\_OPERATIONS 106

#define ID\_EDIT\_VALUE 107

#define ID\_ERROR 108

#define ID\_SUCCESS 109

using namespace std;

enum states {

start, state\_1, state\_2, state\_3, state\_4, state\_5, state\_6, state\_7,

state\_8, state\_9, state\_10, state\_11, state\_12, finish

};

enum actions {

actUnknown, actClearEdit, actConnectDB, actShowError, actShowAllTable, actReturnTo1, actShowFields,

actAddFields, actSelectOnlyFields, actAddWhere, actAddFieldWhere, actAddOperation, actAddValue, actSelectWhere,

actAddLogical, actFinishProgram

};

enum signals {

sigUnknown = 0, sigModeTable = 1, sigModeQuery = 2, sigBtnOk = 3, sigError = 4,

sigSuccess = 5, sigBtnNext = 6, sigAddField = 7, sigAddOperation = 8, sigExitProgram = 9

};

//Таблица переходов

states changeState[14][10] = {

//sigUnknown sigModeTable sigModeQuery sigBtnOk sigError sigSuccess sigBtnNext sigAddField sigAddOperation sigExitProgram

{ state\_1, start, start, start, start, start, start, start, start, start }, //start

{ state\_1, state\_2, state\_6, state\_1, state\_1, state\_1, state\_1, state\_1, state\_1, finish }, //state\_1

{ state\_2, state\_2, state\_2, state\_3, state\_2, state\_2, state\_2, state\_2, state\_2, state\_2 }, //state\_2

{ state\_3, state\_3, state\_3, state\_3, state\_4, state\_5, state\_3, state\_3, state\_3, state\_3 }, //state\_3

{ state\_4, state\_4, state\_4, state\_2, state\_4, state\_4, state\_4, state\_4, state\_4, state\_4 }, //state\_4

{ state\_5, state\_5, state\_5, state\_1, state\_5, state\_5, state\_5, state\_5, state\_5, state\_5 }, //state\_5

{ state\_6, state\_6, state\_6, state\_6, state\_6, state\_6, state\_7, state\_6, state\_6, state\_6 }, //state\_6

{ state\_7, state\_7, state\_7, state\_8, state\_7, state\_7, state\_9, state\_7, state\_7, state\_7 }, //state\_7

{ state\_8, state\_8, state\_8, state\_1, state\_8, state\_8, state\_8, state\_8, state\_8, state\_8 }, //state\_8

{ state\_9, state\_9, state\_9, state\_9, state\_9, state\_9, state\_9, state\_10, state\_9, state\_9 }, //state\_9

{ state\_10, state\_10, state\_10, state\_10, state\_10, state\_10, state\_10, state\_10, state\_11, state\_10 }, //state\_10

{ state\_11, state\_11, state\_11, state\_11, state\_11, state\_11, state\_12, state\_11, state\_11, state\_11 }, //state\_11

{ state\_12, state\_12, state\_12, state\_8, state\_12, state\_12, state\_9, state\_12, state\_12, state\_12 }, //state\_12

{ finish, finish, finish, finish, finish, finish, finish, finish, finish, finish }, //finish

};

//Таблица состояний

actions getAction[14][10] = {

//sigUnknown sigModeTable sigModeQuery sigBtnOk sigError sigSuccess sigBtnNext sigAddField sigAddOperation sigExitProgram

{ actUnknown, actUnknown, actUnknown, actUnknown, actUnknown, actUnknown, actUnknown, actUnknown, actUnknown, actUnknown }, //start

{ actUnknown, actClearEdit, actClearEdit, actUnknown, actUnknown, actUnknown, actUnknown, actUnknown, actUnknown, actFinishProgram }, //state\_1

{ actUnknown, actUnknown, actUnknown, actConnectDB, actUnknown, actUnknown, actUnknown, actUnknown, actUnknown, actUnknown }, //state\_2

{ actUnknown, actUnknown, actUnknown, actUnknown, actShowError, actShowAllTable, actUnknown, actUnknown, actUnknown, actUnknown }, //state\_3

{ actUnknown, actUnknown, actUnknown, actClearEdit, actUnknown, actUnknown, actUnknown, actUnknown, actUnknown, actUnknown }, //state\_4

{ actUnknown, actUnknown, actUnknown, actReturnTo1, actUnknown, actUnknown, actUnknown, actUnknown, actUnknown, actUnknown }, //state\_5

{ actUnknown, actUnknown, actUnknown, actUnknown, actUnknown, actUnknown, actShowFields, actUnknown, actUnknown, actUnknown }, //state\_6

{ actUnknown, actUnknown, actUnknown, actSelectOnlyFields, actUnknown, actUnknown, actAddWhere, actAddFields, actUnknown, actUnknown }, //state\_7

{ actUnknown, actUnknown, actUnknown, actReturnTo1, actUnknown, actUnknown, actUnknown, actUnknown, actUnknown, actUnknown }, //state\_8

{ actUnknown, actUnknown, actUnknown, actUnknown, actUnknown, actUnknown, actUnknown, actAddFieldWhere, actUnknown, actUnknown }, //state\_9

{ actUnknown, actUnknown, actUnknown, actUnknown, actUnknown, actUnknown, actUnknown, actUnknown, actAddOperation, actUnknown }, //state\_10

{ actUnknown, actUnknown, actUnknown, actUnknown, actUnknown, actUnknown, actAddValue, actUnknown, actUnknown, actUnknown }, //state\_11

{ actUnknown, actUnknown, actUnknown, actSelectWhere, actUnknown, actUnknown, actAddLogical, actUnknown, actUnknown, actUnknown }, //state\_12

{ actUnknown, actUnknown, actUnknown, actUnknown, actUnknown, actUnknown, actUnknown, actUnknown, actUnknown, actUnknown }, //finish

};

//Таблица контекста состояний

/\*

0 - Меню, 1 - таблица, 2 - список с полями таблицы, 3 - список со операциями,

4 - текстовое поле для ввода таблицы, 5 - текст.поле для ввода значения, 6 - ОК,

7 - Далее, 8 - многострочное текстовое поле

\*/

string tableContext[14][9] = {

// 0 1 2 3 4 5 6 7 8

{ "+", " ", " ", " ", " ", " ", " ", " ", " " }, //start

{ "+", " ", " ", " ", " ", " ", " ", " ", " " }, //state\_1

{ " ", " ", " ", " ", "+", " ", "+", " ", "+" }, //state\_2

{ " ", " ", " ", " ", "+", " ", "-", " ", "-" }, //state\_3

{ " ", " ", " ", " ", "-", " ", "+", " ", "-" }, //state\_4

{ " ", "+", " ", " ", "-", " ", "+", " ", "-" }, //state\_5

{ " ", " ", " ", " ", "+", " ", "-", "+", "+" }, //state\_6

{ " ", " ", "+", " ", "-", " ", "+", "+", "+" }, //state\_7

{ " ", "+", "-", " ", "-", " ", "+", "-", "-" }, //state\_8

{ " ", " ", "+", "-", "-", "-", "-", "-", "+" }, //state\_9

{ " ", " ", "-", "+", "-", "-", "-", "-", "+" }, //state\_10

{ " ", " ", "-", "-", "-", "+", "-", "+", "+" }, //state\_11

{ " ", " ", "-", "+", "-", " ", "+", "+", "+" }, //state\_12

{ " ", " ", " ", " ", " ", " ", " ", " ", " " }, //finish

};

enum states current\_state = state\_1;

enum signals current\_signal = sigUnknown;

enum actions current\_action;

HINSTANCE hInst;

BOOL RegClass(WNDPROC, LPCTSTR, UINT);

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

HWND CreateListView(HWND, UINT);

void getSignal(int, int);

void changeContext(states);

void doQuery(string curQuery);

string errorMsg;

bool first = true;

int colCount;

HMENU hMainMenu, hModeMenu;

HWND hwndEditTableName;

HWND hwndListFields;

HWND hwndQueryStatic;

HWND hwndBtnOK;

HWND hwndBtnNext;

HWND hwndResTable;

HWND hwndOperations;

HWND hwndEditValue;

string operations[8] = { "=", ">", "<", ">=", "<=", "!=", "AND", "OR"};

MYSQL \*conn, mysql;

MYSQL\_RES \*res;

MYSQL\_ROW row;

MYSQL\_FIELD \*field;

char globalTableName[64];

string globalQuery;

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam){

switch (message){

//region WM\_CREATE

case WM\_CREATE:

{

mysql\_init(&mysql);

conn = mysql\_real\_connect(&mysql, "localhost", "root", "1234", "winapi", 3306, NULL, 0);

hMainMenu = CreateMenu();

hModeMenu = CreatePopupMenu();

AppendMenu(hMainMenu, MF\_STRING | MF\_POPUP, (UINT\_PTR)hModeMenu, "Mode");

AppendMenu(hModeMenu, MF\_STRING, ID\_MENU\_TABLE, "Table");

AppendMenu(hModeMenu, MF\_STRING, ID\_MENU\_QUERY, "Request");

SetMenu(hWnd, hMainMenu);

SetMenu(hWnd, hModeMenu);

hwndEditTableName = CreateWindowEx(0, "EDIT", " ",

WS\_CHILD | ES\_LEFT | WS\_BORDER,

0, 0, 200, 25,

hWnd,(HMENU)ID\_EDIT\_TABLENAME, hInst, NULL);

hwndListFields = CreateWindowEx(0, "listbox", NULL,

WS\_CHILD | LBS\_STANDARD| CBS\_AUTOHSCROLL|WS\_VSCROLL,

210, 0, 150, 200,

hWnd,(HMENU)ID\_LB\_FIELDS, hInst, NULL);

hwndQueryStatic = CreateWindowEx(0, "STATIC", NULL,

WS\_CHILD,

0, 30, 200, 165,

hWnd, nullptr, hInst, NULL);

hwndBtnOK = CreateWindowEx(0, "button", "OK",

WS\_CHILD | BS\_PUSHBUTTON,

450, 10, 50, 25,

hWnd, (HMENU)ID\_BTN\_OK, hInst, NULL);

hwndBtnNext = CreateWindowEx(0, "button", "Next",

WS\_CHILD | BS\_PUSHBUTTON,

450, 45, 50, 25,

hWnd, (HMENU)ID\_BTN\_NEXT, hInst, NULL);

hwndResTable = CreateListView(hWnd, ID\_LV\_RESTABLE);

hwndOperations = CreateWindowEx(WS\_EX\_STATICEDGE, "COMBOBOX", nullptr,

CBS\_DROPDOWNLIST | WS\_CHILD | WS\_VSCROLL,

375, 80, 60, 500,

hWnd, (HMENU)ID\_CB\_OPERATIONS, hInst, NULL);

hwndEditValue = CreateWindowEx(0, "EDIT", "",

WS\_CHILD | ES\_LEFT | WS\_BORDER,

375, 130, 200, 25,

hWnd, (HMENU)ID\_EDIT\_VALUE, hInst, NULL);

EnableWindow(hwndEditValue, FALSE);

break;

}

//endregion

case WM\_COMMAND:

{

int lowordId = LOWORD(wParam);

int hiwordId = HIWORD(wParam);

getSignal(lowordId, hiwordId);

current\_action = getAction[current\_state][current\_signal];

switch (current\_action) {

//region actConnectDB

case actConnectDB:

{

if(conn){

globalQuery = "SELECT \* FROM ";

GetWindowText(hwndEditTableName, globalTableName, 64);

globalQuery += globalTableName;

const char\* q = globalQuery.c\_str();

mysql\_query(conn, q);

if(res = mysql\_store\_result(conn)){

PostMessage(hWnd, WM\_COMMAND, ID\_SUCCESS, hiwordId);

}

else{

errorMsg = mysql\_error(conn);

PostMessage(hWnd, WM\_COMMAND, ID\_ERROR, hiwordId);

}

}

else{

cout << "ERR" << endl;

PostMessage(hWnd, WM\_COMMAND, ID\_ERROR, hiwordId);

}

break;

}

//endregion

//region actClearEdit

case actClearEdit:

{

PostMessage(hwndEditTableName, EM\_SETSEL, 0, -1);

PostMessage(hwndEditTableName, WM\_CLEAR, 0, 0);

break;

}

//endregion

//region actShowError

case actShowError:

{

if(MessageBox(hWnd, errorMsg.c\_str(), "Error", MB\_OK | MB\_ICONERROR) == IDOK)

PostMessage(hWnd, WM\_COMMAND, ID\_BTN\_OK, hiwordId);

break;

}

//endregion

//region actShowAllTable

case actShowAllTable: //Вывод всей таблицы

{

globalQuery = "SELECT \* FROM ";

globalQuery += globalTableName;

SetWindowText(hwndQueryStatic, globalQuery.c\_str());

doQuery(globalQuery);

break;

}

//endregion

//region actReturnTo1

case actReturnTo1:

{

globalQuery = "";

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

ListView\_DeleteColumn(hwndResTable, i);

}

ListView\_DeleteAllItems(hwndResTable);

PostMessage(hwndListFields, LB\_RESETCONTENT, 0, 0);

Static\_SetText(hwndQueryStatic, " ");

break;

}

//endregion

//region actShowFields

case actShowFields: //Вывод полей таблицы

{

string query = "SELECT \* FROM ";

GetWindowText(hwndEditTableName, globalTableName, 64);

query += globalTableName;

const char\* q = query.c\_str();

mysql\_query(conn, q);

res = mysql\_store\_result(conn);

while((field = mysql\_fetch\_field(res))){

SendMessage(hwndListFields, LB\_ADDSTRING, 0, (LPARAM)field->name);

}

globalQuery = "SELECT ";

SetWindowText(hwndQueryStatic, globalQuery.c\_str());

PostMessage(hwndEditValue, EM\_SETSEL, 0, -1);

PostMessage(hwndEditValue, WM\_CLEAR, 0, 0);

break;

}

//endregion

//region actAddFields

case actAddFields: //Добавление поля таблицы в запрос

{

int index = (int)SendMessage(hwndListFields, LB\_GETCURSEL, 0, wParam);

if(index != LB\_ERR){

char fieldName[32];

SendMessage(hwndListFields, LB\_GETTEXT, (WPARAM)index, (LPARAM)fieldName);

if (first){

first = !first;

}

else{

globalQuery += ", ";

}

globalQuery += fieldName;

SetWindowText(hwndQueryStatic, globalQuery.c\_str());

}

break;

}

//endregion

//region actSelectOnlyFields

case actSelectOnlyFields:

{

globalQuery += " FROM ";

globalQuery += globalTableName;

SetWindowText(hwndQueryStatic, globalQuery.c\_str());

doQuery(globalQuery);

break;

}

//endregion

//region actAddWhere

case actAddWhere:

{

globalQuery += " FROM ";

globalQuery += globalTableName;

globalQuery += " WHERE ";

SetWindowText(hwndQueryStatic, globalQuery.c\_str());

first = true;

break;

}

//endregion

//region actAddFieldWhere

case actAddFieldWhere: //Выбор поля для WHERE

{

int fieldsIndex = (int)SendMessage(hwndListFields, LB\_GETCURSEL, 0, wParam);

if (fieldsIndex != LB\_ERR){

char fieldName[10];

SendMessage(hwndListFields, LB\_GETTEXT, (WPARAM)fieldsIndex, (LPARAM)fieldName);

globalQuery += fieldName;

SetWindowText(hwndQueryStatic, globalQuery.c\_str());

}

for (int i = 0; i < 6; i++){

ComboBox\_AddString(hwndOperations, operations[i].c\_str());

}

}

//endregion

//region actAddOperation

case actAddOperation: //Добавление операции сравнения в запрос

{

first = true;

int operationIndex = SendMessage(hwndOperations, CB\_GETCURSEL, 0, 0);

if (operationIndex != CB\_ERR){

globalQuery += operations[operationIndex];

SetWindowText(hwndQueryStatic, globalQuery.c\_str());

ComboBox\_ResetContent(hwndOperations);

}

break;

}

//endregion

//region actAddValue

case actAddValue:

{

char value[64];

GetWindowText(hwndEditValue, value, 64);

globalQuery += "'";

globalQuery += value;

globalQuery += "'";

SetWindowText(hwndQueryStatic, globalQuery.c\_str());

for (int i = 6; i < 8; i++){

ComboBox\_AddString(hwndOperations, operations[i].c\_str());

}

break;

}

//endregion

//region actSelectWhere

case actSelectWhere:

{

ComboBox\_ResetContent(hwndOperations);

doQuery(globalQuery);

break;

}

//endregion

//region actAddLogical

case actAddLogical:

{

int logicalIndex = SendMessage(hwndOperations, CB\_GETCURSEL, 0, 0);

if (logicalIndex != CB\_ERR){

globalQuery += " ";

globalQuery += operations[logicalIndex + 6];

globalQuery += " ";

SetWindowText(hwndQueryStatic, globalQuery.c\_str());

ComboBox\_ResetContent(hwndOperations);

}

PostMessage(hwndEditValue, EM\_SETSEL, 0, -1);

PostMessage(hwndEditValue, WM\_CLEAR, 0, 0);

break;

}

//endregion

//region actFinishProgram

case actFinishProgram:

{

mysql\_close(conn);

PostQuitMessage(0);

break;

}

//endregion

default: break;

}

current\_state = changeState[current\_state][current\_signal];

changeContext(current\_state);

}

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

case WM\_CLOSE:

SendMessage(hWnd, WM\_COMMAND, ID\_MENU\_EXIT, 0);

break;

default:

return DefWindowProc(hWnd, message, wParam, lParam);

}

return 0;

}

void doQuery(string curQuery){

const char\* q = curQuery.c\_str();

if(mysql\_query(conn, q) != 0){

MessageBox(NULL, "Error: can't execute query", "Err", MB\_OK | MB\_ICONERROR);

return;

}

res = mysql\_store\_result(conn);

//Структура для создания столбцов

LVCOLUMN structColumn;

memset(&structColumn, 0, sizeof(structColumn));

structColumn.mask = LVCF\_FMT | LVCF\_WIDTH | LVCF\_TEXT;

structColumn.fmt = LVCFMT\_LEFT;

structColumn.cx = 100; //ширина столбца

int i = 0;

while(field = mysql\_fetch\_field(res)){

structColumn.pszText = field->name;

ListView\_InsertColumn(hwndResTable, i, &structColumn);

i++;

}

int lviRow = 0;

colCount = mysql\_num\_fields(res);

while(row = mysql\_fetch\_row(res)){

LVITEM item;

memset(&item, 0, sizeof(LVITEM));

item.mask = LVIF\_TEXT;

item.iItem = lviRow;

int list = ListView\_InsertItem(hwndResTable, &item);

for(int j = 0; j < colCount; j++){

item.iSubItem = j;

ListView\_InsertItem(hwndResTable, &item);

ListView\_SetItemText(hwndResTable, list, j, row[j]);

}

lviRow++;

}

first = true;

}

void getSignal(int lowordId, int hiwordId){

switch (lowordId) {

case ID\_MENU\_TABLE:

current\_signal = sigModeTable;

break;

case ID\_MENU\_QUERY:

current\_signal = sigModeQuery;

break;

case ID\_BTN\_OK:

current\_signal = sigBtnOk;

break;

case ID\_BTN\_NEXT:

current\_signal = sigBtnNext;

if (current\_state == state\_12)

if (SendMessage(hwndOperations, CB\_GETCURSEL, 0, 0) != CB\_ERR)

current\_signal = sigBtnNext;

else current\_signal = sigUnknown;

break;

case ID\_ERROR:

current\_signal = sigError;

break;

case ID\_SUCCESS:

current\_signal = sigSuccess;

break;

case ID\_LB\_FIELDS:

switch (hiwordId) {

case LBN\_DBLCLK:

current\_signal = sigAddField;

break;

default:

current\_signal = sigUnknown;

}

break;

case ID\_CB\_OPERATIONS:

switch (hiwordId) {

case CBN\_SELCHANGE:

current\_signal = sigAddOperation;

break;

default:

current\_signal = sigUnknown;

}

break;

case ID\_MENU\_EXIT:

current\_signal = sigExitProgram;

break;

default:

current\_signal = sigUnknown;

break;

}

}

void changeContext(states state){

for (int i = 0; i < 9; i++)

switch (i){

case 0:

EnableMenuItem(hMainMenu, ID\_MENU\_TABLE, tableContext[state][i] == "+" ? MF\_ENABLED : MF\_DISABLED);

EnableMenuItem(hMainMenu, ID\_MENU\_QUERY, tableContext[state][i] == "+" ? MF\_ENABLED : MF\_DISABLED);

EnableMenuItem(hMainMenu, ID\_MENU\_EXIT, tableContext[state][i] == "+" ? MF\_ENABLED : MF\_DISABLED);

break;

case 1:

ShowWindow(hwndResTable, tableContext[state][i] != " " ? SW\_SHOW : SW\_HIDE);

EnableWindow(hwndResTable, tableContext[state][i] == "+" ? true : false);

break;

case 2:

ShowWindow(hwndListFields, tableContext[state][i] != " " ? SW\_SHOW : SW\_HIDE);

EnableWindow(hwndListFields, tableContext[state][i] == "+" ? true : false);

break;

case 3:

ShowWindow(hwndOperations, tableContext[state][i] != " " ? SW\_SHOW : SW\_HIDE);

EnableWindow(hwndOperations, tableContext[state][i] == "+" ? true : false);

break;

case 4:

ShowWindow(hwndEditTableName, tableContext[state][i] != " " ? SW\_SHOW : SW\_HIDE);

EnableWindow(hwndEditTableName, tableContext[state][i] == "+" ? true : false);

break;

case 5:

ShowWindow(hwndEditValue, tableContext[state][i] != " " ? SW\_SHOW : SW\_HIDE);

EnableWindow(hwndEditValue, tableContext[state][i] == "+" ? true : false);

break;

case 6:

ShowWindow(hwndBtnOK, tableContext[state][i] != " " ? SW\_SHOW : SW\_HIDE);

EnableWindow(hwndBtnOK, tableContext[state][i] == "+" ? true : false);

break;

case 7:

ShowWindow(hwndBtnNext, tableContext[state][i] != " " ? SW\_SHOW : SW\_HIDE);

EnableWindow(hwndBtnNext, tableContext[state][i] == "+" ? true : false);

break;

case 8:

ShowWindow(hwndQueryStatic, tableContext[state][i] != " " ? SW\_SHOW : SW\_HIDE);

EnableWindow(hwndQueryStatic, tableContext[state][i] == "+" ? true : false);

break;

default: break;

}

InvalidateRect(GetParent(hwndQueryStatic), NULL, TRUE);

}

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow){

MSG msg;

hInst = hInstance;

if (!RegClass(WndProc, "Main", COLOR\_WINDOW)) { return FALSE; };

HWND hWnd = CreateWindow("Main", "Main",

WS\_VISIBLE | WS\_SYSMENU | WS\_MINIMIZEBOX,

CW\_USEDEFAULT, CW\_USEDEFAULT,

600, 600,

nullptr, nullptr, hInstance, nullptr);

if (!hWnd){

MessageBox(NULL, "Window Creation Failed.", "Error", MB\_OK | MB\_ICONERROR);

return FALSE;

}

ShowWindow(hWnd, nCmdShow);

UpdateWindow(hWnd);

while (GetMessage(&msg, nullptr, 0, 0)){

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return msg.wParam;

}

BOOL RegClass(WNDPROC Proc, LPCTSTR szName, UINT brBackground){

WNDCLASS wc;

wc.style = CS\_HREDRAW | CS\_VREDRAW;

wc.cbClsExtra = 0;

wc.cbWndExtra = 0;

wc.lpfnWndProc = Proc;

wc.hInstance = hInst;

wc.hIcon = LoadIcon(NULL, IDI\_APPLICATION);

wc.hCursor = LoadCursor(NULL, IDC\_ARROW);

wc.hbrBackground = (HBRUSH)(brBackground + 1);

wc.lpszMenuName = (LPCTSTR)NULL;

wc.lpszClassName = szName;

return (RegisterClass(&wc) != 0);

}

HWND CreateListView(HWND hWndParent, UINT uId){

INITCOMMONCONTROLSEX icex;

icex.dwICC = ICC\_LISTVIEW\_CLASSES;

InitCommonControlsEx(&icex);

RECT rcClient;

GetClientRect(hWndParent, &rcClient);

HWND hWndLV = CreateWindowEx(NULL, WC\_LISTVIEW, NULL,

WS\_CHILD | WS\_BORDER | LVS\_REPORT | LVS\_REPORT | LVS\_EDITLABELS | LVS\_EX\_FULLROWSELECT,

0, 250,

600, 300,

hWndParent, (HMENU)uId, hInst, NULL);

ListView\_SetExtendedListViewStyleEx(hWndLV, 0, LVS\_EX\_FULLROWSELECT | LVS\_EX\_GRIDLINES);

return (hWndLV);

}

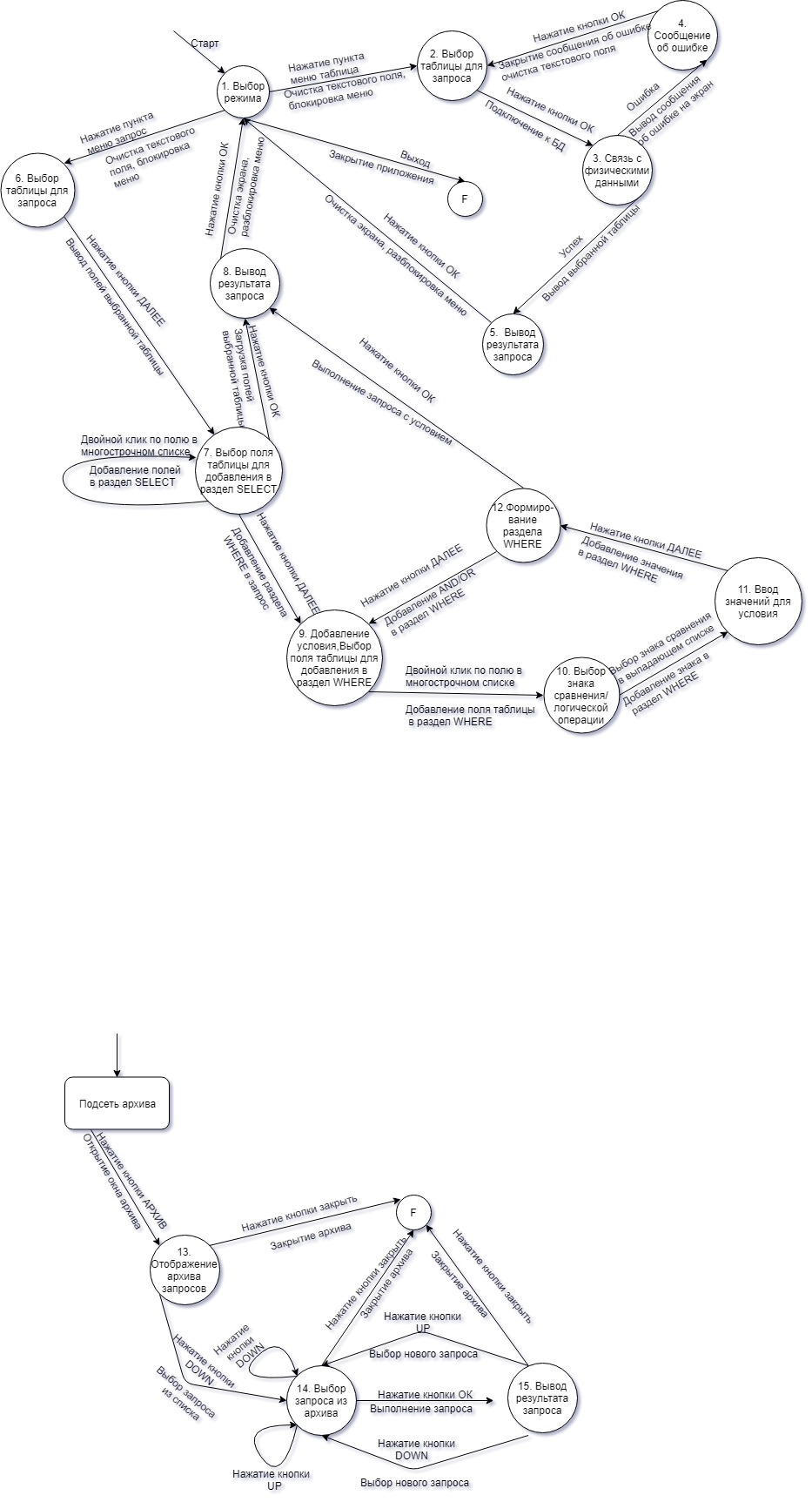


Рисунок 1 - Схема простой транзитивной сети

Таблица контекста состояний:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Cостояние/объект (окно) | Меню | Таблица для отображе-ния данных | Список с полями таблицы | Список с операциями сравнения | Поле для ввода имени таблицы | Поле для ввода значения | Кнопка ОК | Кнопка ДАЛЕЕ | Многост-рочное текстовое поле для текста запроса | Сооб-щение об ошиб-ке |
| Start |  |  |  |  |  |  |  |  |  |  |
| 1 | + |  |  |  |  |  |  |  |  |  |
| 2 | - |  |  |  | + |  | + |  | + |  |
| 3 |  |  |  |  |  |  | - |  | - |  |
| 4 |  |  |  |  | - |  | + |  |  | + |
| 5 |  | + |  |  |  |  |  |  |  |  |
| 6 |  |  |  |  | + |  | - | + | + |  |
| 7 |  |  | + |  | - |  | + |  |  |  |
| 8 |  | + | - |  |  |  |  | - | - |  |
| 9 |  |  | + | - |  | - | - |  | + |  |
| 10 |  |  | - | + |  |  |  |  |  |  |
| 11 |  |  |  | - |  | + |  | + |  |  |
| 12 |  |  |  | + |  |  | + |  |  |  |
| finish |  |  |  |  |  |  |  |  |  |  |

Таблица состояний:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Входные сигналы /состояния | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| Нажатие пункта меню таблица | Очистка текстового поля, блокировка меню |  |  |  |  |  |  |  |  |  |  |  |
| Нажатие пункта меню запрос | Очистка текстового поля, блокировка меню |  |  |  |  |  |  |  |  |  |  |  |
| Нажатие кнопки ОК |  | Подключение к БД |  | Закрытие сообщения об ошибке, очистка текстового поля | Очистка экрана, разблокировка меню |  | Загрузка полей выбранной таблицы | Очистка экрана, разблокировка меню |  |  |  | Выполнение запроса с условием |
| Ошибка |  |  | Вывод сообщения об ошибке |  |  |  |  |  |  |  |  |  |
| Успех |  |  | Вывод выбранной таблицы |  |  |  |  |  |  |  |  |  |
| Нажатие кнопки ДАЛЕЕ |  |  |  |  |  | Вывод полей выбранной таблицы | Добавление раздела WHERE в запрос |  |  |  | Добавление значения в раздел WHERE | Добавление AND/OR в раздел WHERE |
| Двойной клик по полю в многост-рочном списке |  |  |  |  |  |  | Добавление полей в раздел SELECT |  | Добавление поля таблицы в раздел WHERE |  |  |  |
| Выбор знака сравнения в выпадающем списке |  |  |  |  |  |  |  |  |  | Добавление знака в раздел WHERE |  |  |
| Выход | Закрытие приложения |  |  |  |  |  |  |  |  |  |  |  |

Таблица переходов / таблица действий:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Входные сигналы /состояния | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| Нажатие пункта меню таблица | 2 |  |  |  |  |  |  |  |  |  |  |  |
| Нажатие пункта меню запрос | 6 |  |  |  |  |  |  |  |  |  |  |  |
| Нажатие кнопки ОК |  | 3 |  | 2 | 1 |  | 8 | 1 |  |  |  | 8 |
| Ошибка |  |  | 4 |  |  |  |  |  |  |  |  |  |
| Успех |  |  | 5 |  |  |  |  |  |  |  |  |  |
| Нажатие кнопки ДАЛЕЕ |  |  |  |  |  | 7 | 9 |  |  |  | 12 | 9 |
| Двойной клик по полю в многострочном списке |  |  |  |  |  |  | 7 |  | 10 |  |  |  |
| Выбор знака сравнения в выпадающем списке |  |  |  |  |  |  |  |  |  | 11 |  |  |
| Выход | F |  |  |  |  |  |  |  |  |  |  |  |