*Министерство образования и науки Российской федерации*

федеральное государственное бюджетное образование учреждение

высшего образования

«Курганский государственный университет»

РФ КГУ 09.03.03 КП20 910038 07

Кафедра «Программное обеспечение автоматизированных систем»

**КУРСОВАЯ РАБОТА**

**по дисциплине**

**«Алгоритмы и структуры данных»**

**Руководство программиста**

Листов 27

Выполнил студент группы ИТ-20819\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Степанов Д.А. /

Проверил канд. техн. наук, доцент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Семахин А.М. /

Курган, 2020

# АННОТАЦИЯ

Документ содержит общие сведения о программе, функциональное назначение, описание логической структуры, используемых технических средств, входных и выходных данных.

Предназначен для программистов.

Оглавление

[1 ОБЩИЕ СВЕДЕНИЯ 4](#_Toc59708316)

[1.1 Программное обеспечение 4](#_Toc59708317)

[1.2 Языки программирования 4](#_Toc59708318)

[2 ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ 5](#_Toc59708319)

[3 ОПИСАНИЕ ЛОГИЧЕСКОЙ СТРУКТУРЫ 6](#_Toc59708320)

[4 ВХОДНЫЕ И ВЫХОДНЫЕ ДАННЫЕ 7](#_Toc59708321)

[5 ЛИСТИНГ АЛГОРИТМА 8](#_Toc59708322)

# 1 ОБЩИЕ СВЕДЕНИЯ

Данное приложение формализует метод нахождения остовного дерева минимального веса по Алгоритму Прима. Визуальное приложение разработано с использованием объектно-ориентированного метода программирования. В программе обеспечены: ввод исходных данных (количество вершин, вес рёбер), поиск минимального остовного дерева, вывод данных в форме графа и таблицы.

### Программное обеспечение

Данная программа выполняется под управлением операционной системы Windows 10. Для запуска программы необходимо наличие исполняемого файла cursedprima.exe. Для отображения графа необходимо установить программу graphviz 2.44.2.exe.

## 1.2 Языки программирования

Для реализации программного продукта выбрана среда разработки MS Visual Studio 2019 и язык программирования С++.

# ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ

Данное приложение формализует метод нахождения остовного дерева минимального веса по Алгоритму Прима. Визуальное приложение разработано с использованием объектно-ориентированного метода программирования. В программе обеспечены: ввод исходных данных (количество вершин, вес рёбер), поиск минимального остовного дерева, вывод данных в форме графа и таблицы.

# ОПИСАНИЕ ЛОГИЧЕСКОЙ СТРУКТУРЫ

Программа основана на нескольких функциях. В них реализованы следующие возможности программы:

1) Заполнение матрицы смежности вручную.

2) Заполнение матрицы смежности автоматически, используя генератор псевдослучайных целых чисел.

3) Поиск остовного дерева минимальной стоимости, используя Алгоритм Прима.

4) Графическое представление заданного графа.

5) Возможность просмотреть подробный отчет о работе алгоритма.

# 4 ВХОДНЫЕ И ВЫХОДНЫЕ ДАННЫЕ

Исходными данными в программе являются:

1. Ввод количества вершин.
2. Возможность ручного заполнения матрицы смежности.

Выходными данными в программе являются:

1. Остовное дерево минимальное стоимости.
2. Графическое представление графа.
3. Подробный отчет о работе алгоритма.

# 5 ЛИСТИНГ АЛГОРИТМА

*// Вектор.*

*vector<vector<int>> prima\_matriX;*

*vector<int> temp;*

*vector<int> temp\_num;*

*vector<int> temp\_reb;*

*vector<vector<int>> graph\_prima;*

*// Предикат для функции min\_element.*

*struct mypred {*

*bool operator()(const pair<int, int> f\_p, const pair<int, int> s\_p) {*

*return f\_p.second < s\_p.second;*

*}*

*};*

*// Алгоритм Прима.*

*int algorithm\_prima(vector<vector<int>> vec\_name, int number\_of\_vertex, int beg\_vertex, ListBox^ listbox\_name) {*

*int new\_vertex = 0, counter = 1;*

*vector<int> checked\_vertex, unchecked\_vertex;*

*vector<pair<int, int>> ways\_arr;*

*pair<int, int> min\_vertex;*

*String^ text;*

*// Выбирается вершина.*

*checked\_vertex.push\_back(beg\_vertex - 1);*

*text = "Выбранная вершина: " + Convert::ToString(beg\_vertex);*

*listbox\_name->Items->Add(text);*

*// Остальные number\_of\_vertex - 1 вершин - невыбраны.*

*for (int i = 0; i < number\_of\_vertex; i++)*

*if (i != beg\_vertex - 1) unchecked\_vertex.push\_back(i);*

*// Алгоритм Прима.*

*listbox\_name->Items->Add("Начало работы алгоритма.");*

*listbox\_name->Items->Add("------------------------");*

*for (int i = beg\_vertex - 1; i < number\_of\_vertex; ) {*

*// Определение веса между выбранной вершиной и остальными невыбранными.*

*for (int g = 0; g < number\_of\_vertex; g++) {*

*if (checked\_vertex.size() > 1) {*

*auto f = find(checked\_vertex.begin(), checked\_vertex.end(), g);*

*if (f != checked\_vertex.end()) continue;*

*}*

*if (vec\_name[i][g] != 0 && vec\_name[i][g] != -1)*

*ways\_arr.emplace\_back(g, vec\_name[i][g]);*

*}*

*// Определяем вершину с наименьшим весом до неё. Фиксируем выбранное ребро и вес.*

*min\_vertex = \*min\_element(ways\_arr.begin(), ways\_arr.end(), mypred());*

*// Выбранную вершину исключаем из перечня невыбранных.*

*if (unchecked\_vertex.size() != 0) {*

*auto it = find(unchecked\_vertex.begin(), unchecked\_vertex.end(), min\_vertex.first);*

*unchecked\_vertex.erase(it);*

*ways\_arr.erase(min\_element(ways\_arr.begin(), ways\_arr.end(), mypred()));*

*for (auto del\_it = ways\_arr.begin(); del\_it != ways\_arr.end(); ) {*

*if ((\*del\_it).first == min\_vertex.first) del\_it = ways\_arr.erase(del\_it);*

*else del\_it++;*

*}*

*}*

*checked\_vertex.push\_back(min\_vertex.first);*

*text = "Следующая вершина: " + Convert::ToString(min\_vertex.first + 1);*

*listbox\_name->Items->Add(text);*

*text = "Вес ребра между вершиной " + Convert::ToString(i + 1) + " и " + Convert::ToString(min\_vertex.first + 1) + ": " + Convert::ToString(min\_vertex.second);*

*listbox\_name->Items->Add(text);*

*i = min\_vertex.first;*

*new\_vertex += min\_vertex.second;*

*// Если проверены все вершины, то функция завершает свою работу.*

*text = "Значение пути после выполнения " + Convert::ToString(counter) + "-ой";*

*listbox\_name->Items->Add(text);*

*text = " итерации цикла : " + Convert::ToString(new\_vertex);*

*listbox\_name->Items->Add(text);*

*if (checked\_vertex.size() == number\_of\_vertex) break;*

*counter++;*

*listbox\_name->Items->Add("// Переход на следующую итерацию //");*

*listbox\_name->Items->Add("//-------------------------------//");*

*}*

*return new\_vertex;*

*}*

*#pragma endregion*

*// Объявление переменных.*

*public:*

*int n, val;*

*Point^ colnrow;*

*int floppy = 0, s\_floppy = 0, rfloppy = 0, rs\_floppy = 0, tmp\_fl = 0;*

*int tableindex = 1;*

*// Кнопка, запоминающая количество вершин. Открывает доступ к основному функционалу программы.*

*private: System::Void btnaccept\_Click(System::Object^ sender, System::EventArgs^ e) {*

*n = Convert::ToInt32(numvertex->Value);*

*numvertex->Enabled = false;*

*btnaccept->Enabled = false;*

*btnrandom->Enabled = true;*

*btnacceptmatrix->Enabled = true;*

*datamatrix->ColumnCount = n;*

*datamatrix->RowCount = n;*

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

*datamatrix->Columns[i]->HeaderText = "x" + Convert::ToString(i + 1);*

*datamatrix->Columns[i]->SortMode = System::Windows::Forms::DataGridViewColumnSortMode::NotSortable;*

*}*

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

*for (int j = 0; j < n; j++) {*

*if (i == j) {*

*datamatrix->Rows[i]->Cells[j]->ReadOnly = true;*

*datamatrix->Rows[i]->Cells[j]->Value = 0;*

*}*

*}*

*}*

*}*

*// Заполнение матрицы вручную.*

*private: System::Void datamatrix\_CellValueChanged(System::Object^ sender, System::Windows::Forms::DataGridViewCellEventArgs^ e) {*

*colnrow = datamatrix->CurrentCellAddress;*

*if (colnrow->X != 0 || colnrow->Y != 0) {*

*if (colnrow->X != -1 && colnrow->Y != -1) {*

*val = Convert::ToInt32(datamatrix->Rows[colnrow->Y]->Cells[colnrow->X]->Value);*

*if (val == 0)*

*datamatrix->Rows[colnrow->Y]->Cells[colnrow->X]->Value = -1;*

*datamatrix->Rows[colnrow->X]->Cells[colnrow->Y]->Value = datamatrix->Rows[colnrow->Y]->Cells[colnrow->X]->Value;*

*}*

*}*

*}*

*// Заполнение матрицы псевдослучайными числами.*

*private: System::Void btnrandom\_Click(System::Object^ sender, System::EventArgs^ e) {*

*srand(time(NULL));*

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

*for (int j = 0; j < n; j++) {*

*if (i != j) {*

*datamatrix->Rows[i]->Cells[j]->Value = -1 + rand() % 125;*

*if (Convert::ToInt32(datamatrix->Rows[i]->Cells[j]->Value) == 0) datamatrix->Rows[i]->Cells[j]->Value = -1;*

*datamatrix->Rows[j]->Cells[i]->Value = datamatrix->Rows[i]->Cells[j]->Value;*

*}*

*}*

*}*

*}*

*// Кнопка, запрещающая дальнейшее изменение матрицы.*

*private: System::Void btnacceptmatrix\_Click(System::Object^ sender, System::EventArgs^ e) {*

*btnrunprima->Enabled = true;*

*btnacceptmatrix->Enabled = false;*

*btnrandom->Enabled = false;*

*graphpaint->Enabled = true;*

*}*

*// Кнопка, вызывающая функцию algorithm\_prima(). Расчет основного алгоритма.*

*private: System::Void btnrunprima\_Click(System::Object^ sender, System::EventArgs^ e) {*

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

*for (int j = 0; j < n; j++) {*

*temp.push\_back(Convert::ToInt32(datamatrix->Rows[j]->Cells[i]->Value));*

*}*

*prima\_matriX.push\_back(temp);*

*temp.clear();*

*}*

*srand(time(NULL));*

*int rand\_vertex = 1 + rand() % n;*

*double start, finish, final\_time;*

*start = GetTickCount64();*

*int answer = algorithm\_prima(prima\_matriX, n, rand\_vertex, listcalc);*

*finish = GetTickCount64();*

*txtanswer->Text = Convert::ToString(answer);*

*final\_time = (finish - start) / 1000;*

*txttimesec->Text = Convert::ToString(final\_time) + " секунд.";*

*final\_time = finish - start;*

*txttimemilli->Text = Convert::ToString(final\_time) + " миллисекунд.";*

*if (n >= 16) graphpaint->Enabled = false;*

*}*

*// Кнопка очищения всех контейнеров, переменных. Перевод программы в её изначальное состояние.*

*private: System::Void btnclear\_Click(System::Object^ sender, System::EventArgs^ e) {*

*numvertex->Value = 2;*

*btnaccept->Enabled = true;*

*btnrunprima->Enabled = false;*

*btnacceptmatrix->Enabled = false;*

*btnrandom->Enabled = false;*

*txtanswer->Text = "";*

*txttimemilli->Text = "";*

*txttimesec->Text = "";*

*prima\_matriX.clear();*

*temp.clear();*

*datamatrix->ColumnCount = 0;*

*datamatrix->RowCount = 0;*

*numvertex->Enabled = true;*

*listcalc->Items->Clear();*

*graphpaint->Enabled = false;*

*toolgraph->Enabled = false;*

*picbox->Visible = false;*

*DeleteFile(L"painted\_graph.dot");*

*picbox->Image = nullptr;*

*DeleteFile(L"graph.png");*

*}*

*// Кнопка, отвечающая за рисование графа.*

*private: System::Void graphpaint\_Click(System::Object^ sender, System::EventArgs^ e) {*

*String^ path = savefile->FileName;*

*String^ text;*

*StreamWriter^ file = gcnew StreamWriter(path, false);*

*if (savefile->ShowDialog() == System::Windows::Forms::DialogResult::OK) {*

*file->WriteLine("digraph {");*

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

*for (int j = i; j < n; j++) {*

*if (i != j) {*

*text = Convert::ToString(i + 1) + "->" + Convert::ToString(j + 1) + "[label=\"" + Convert::ToString(datamatrix->Rows[j]->Cells[i]->Value) + "\"];";*

*file->WriteLine(text);*

*}*

*}*

*}*

*file->WriteLine("}");*

*file->Close();*

*system("dot painted\_graph.dot -Tpng -ograph.png");*

*picbox->Image = Image::FromFile("graph.png");*

*toolgraph->Enabled = true;*

*picbox->Visible = true;*

*graphpaint->Enabled = false;*

*MessageBox::Show("Если не перезапускать программу, то следующий граф не будет рисоваться. Причина в том, что приложение даже после очищения picturebox'а не удаляет информацию о рисуемой картинке, поэтому она не перезаписывается. Я не знаю, как решить эту проблему.", "Предупреждение", MessageBoxButtons::OK, MessageBoxIcon::Information);*

*}*

*}*

*// Кнопка меню, отвечающая за вывод графа на экран.*

*private: System::Void toolgraph\_Click(System::Object^ sender, System::EventArgs^ e) {*

*picbox->Visible = true;*

*panelgraphcomp->Visible = false;*

*groupBox1->Visible = true;*

*groupBox2->Visible = true;*

*groupBox3->Visible = true;*

*groupcalculates->Visible = true;*

*}*

*private: System::Void toolhidepicbox\_Click(System::Object^ sender, System::EventArgs^ e) {*

*picbox->Visible = false;*

*panelgraphcomp->Visible = false;*

*groupBox1->Visible = true;*

*groupBox2->Visible = true;*

*groupBox3->Visible = true;*

*groupcalculates->Visible = true;*

*}*

*private: System::Void toolbtncomputing\_Click(System::Object^ sender, System::EventArgs^ e) {*

*panelgraphcomp->Visible = true;*

*picbox->Visible = false;*

*groupBox1->Visible = false;*

*groupBox2->Visible = false;*

*groupBox3->Visible = false;*

*groupcalculates->Visible = false;*

*}*

*private: System::Void btnaddvertex\_Click(System::Object^ sender, System::EventArgs^ e) {*

*datatable->RowCount = 6;*

*datatable->Rows[5]->Cells[0]->Value = "Сум";*

*if (tableindex == 5) {*

*btnaddvertex->Enabled = false;*

*btncomputinggraph->Enabled = true;*

*}*

*int num = Convert::ToInt32(numvertexesnumber->Value);*

*// X1i - количество вершин.*

*datatable->Rows[tableindex - 1]->Cells[1]->Value = num;*

*datatable->Rows[tableindex - 1]->Cells[4]->Value = num \* num;*

*// X2i - количество рёбер.*

*int rnum = Convert::ToInt32(numrebsnumber->Value);*

*datatable->Rows[tableindex - 1]->Cells[2]->Value = rnum;*

*datatable->Rows[tableindex - 1]->Cells[5]->Value = rnum \* rnum;*

*// X1i \* X2i.*

*datatable->Rows[tableindex - 1]->Cells[6]->Value = num \* rnum;*

*// Номер опыта.*

*datatable->Rows[tableindex - 1]->Cells[0]->Value = tableindex;*

*temp\_num.push\_back(num);*

*temp\_reb.push\_back(rnum);*

*tableindex++;*

*floppy += num;*

*rfloppy += rnum;*

*s\_floppy += num \* num;*

*rs\_floppy += rnum \* rnum;*

*tmp\_fl += num \* rnum;*

*datatable->Rows[5]->Cells[1]->Value = floppy; // X1i.*

*datatable->Rows[5]->Cells[4]->Value = s\_floppy; // X1i^2.*

*datatable->Rows[5]->Cells[2]->Value = rfloppy; // X2i.*

*datatable->Rows[5]->Cells[5]->Value = rs\_floppy; // X2i^2.*

*datatable->Rows[5]->Cells[6]->Value = tmp\_fl; // X1i \* X2i.*

*}*

*private: System::Void numvertexesnumber\_ValueChanged(System::Object^ sender, System::EventArgs^ e) {*

*int tmp = Convert::ToInt32(numvertexesnumber->Value);*

*int final\_reb = (tmp \* tmp - tmp) / 4;*

*txtmaxrebs->Text = Convert::ToString((tmp \* tmp - tmp) / 2);*

*numrebsnumber->Minimum = final\_reb + 1;*

*numrebsnumber->Value = numrebsnumber->Minimum;*

*}*

*private: System::Void btncomputinggraph\_Click(System::Object^ sender, System::EventArgs^ e) {*

*graphic->Series[0]->Points->Clear();*

*// Всё это - заполнение таблицы:*

*srand(time(NULL));*

*temp.clear();*

*int calc\_rand = 0;*

*int rebs\_count;*

*// Количество рёбер, равные -1:*

*for (int f = 0; f < temp\_num.size(); f++) {*

*for (int i = 0; i < temp\_num[f]; i++) {*

*for (int j = 0; j < temp\_num[f]; j++) {*

*calc\_rand = 1 + rand() % 130;*

*if (i == j) temp.push\_back(0);*

*else temp.push\_back(calc\_rand);*

*}*

*graph\_prima.push\_back(temp);*

*temp.clear();*

*}*

*// Смежные вершины равны друг другу...*

*for (int k = 0; k < temp\_num[f]; k++) {*

*for (int kk = k; kk < temp\_num[f]; kk++) {*

*if (k == kk) graph\_prima[kk][k] = 0;*

*graph\_prima[k][kk] = graph\_prima[kk][k];*

*}*

*}*

*// Заполнение -1.*

*rebs\_count = (temp\_num[f] \* temp\_num[f] - temp\_num[f]) / 2 - temp\_reb[f];*

*int false\_key = 1;*

*bool flag\_key = false;*

*while (rebs\_count != 0 && false\_key <= temp\_num[f]) {*

*for (int g = 1; g < temp\_num[f]; g++) {*

*if (graph\_prima[false\_key][g] == 0) continue;*

*else if (rebs\_count > 0) {*

*graph\_prima[false\_key][g] = -1;*

*graph\_prima[g][false\_key] = -1;*

*rebs\_count--;*

*}*

*if (rebs\_count == 0) {*

*flag\_key = true;*

*break;*

*}*

*}*

*if (flag\_key == true) break;*

*false\_key++;*

*}*

*int start = 0, finish = 0, final\_time;*

*int answer = 0;*

*while ((finish - start) == 0) {*

*start = GetTickCount64();*

*answer = algorithm\_prima(graph\_prima, temp\_num[f], 1, listcalc);*

*finish = GetTickCount64();*

*}*

*listcalc->Items->Clear();*

*graph\_prima.clear();*

*final\_time = finish - start;*

*// Yi*

*datatable->Rows[f]->Cells[3]->Value = final\_time;*

*// Yi^2*

*datatable->Rows[f]->Cells[7]->Value = final\_time \* final\_time;*

*// X1i \* Yi.*

*datatable->Rows[f]->Cells[8]->Value = final\_time \* Convert::ToInt32(datatable->Rows[f]->Cells[1]->Value);*

*// X2i \* Yi.*

*datatable->Rows[f]->Cells[9]->Value = final\_time \* Convert::ToInt32(datatable->Rows[f]->Cells[2]->Value);*

*}*

*// Вычисления корреляции, детерминации и т.д.*

*double x1i = 0, x2i, yi = 0, xx1i = 0, xx2i = 0, x1ix2i = 0, yyi = 0, x1iyi = 0, x2iyi = 0;*

*for (int colrow = 0; colrow < 5; colrow++) {*

*yi += Convert::ToInt32(datatable->Rows[colrow]->Cells[3]->Value);*

*yyi += Convert::ToInt32(datatable->Rows[colrow]->Cells[7]->Value);*

*datatable->Rows[colrow]->Cells[8]->Value = Convert::ToInt32(datatable->Rows[colrow]->Cells[1]->Value) \* Convert::ToInt32(datatable->Rows[colrow]->Cells[3]->Value);*

*x1iyi += Convert::ToInt32(datatable->Rows[colrow]->Cells[8]->Value);*

*datatable->Rows[colrow]->Cells[9]->Value = Convert::ToInt32(datatable->Rows[colrow]->Cells[2]->Value) \* Convert::ToInt32(datatable->Rows[colrow]->Cells[3]->Value);*

*x2iyi += Convert::ToInt32(datatable->Rows[colrow]->Cells[9]->Value);*

*// График.*

*graphic->Series[0]->Points->AddXY(Convert::ToInt32(datatable->Rows[colrow]->Cells[3]->Value), Convert::ToInt32(datatable->Rows[colrow]->Cells[1]->Value));*

*}*

*datatable->Rows[5]->Cells[3]->Value = yi;*

*datatable->Rows[5]->Cells[7]->Value = yyi;*

*datatable->Rows[5]->Cells[8]->Value = x1iyi;*

*datatable->Rows[5]->Cells[9]->Value = x2iyi;*

*x1i = Convert::ToDouble(datatable->Rows[5]->Cells[1]->Value);*

*x2i = Convert::ToDouble(datatable->Rows[5]->Cells[2]->Value);*

*xx1i = Convert::ToDouble(datatable->Rows[5]->Cells[4]->Value);*

*xx2i = Convert::ToDouble(datatable->Rows[5]->Cells[5]->Value);*

*x1ix2i = Convert::ToDouble(datatable->Rows[5]->Cells[6]->Value);*

*// Здесь далее будут высчитываться значения коэффициентов a1, a0, корреляции, детерминации. Будет составляться уравнение связи.*

*double correlation, determination, sigmay, sigmax1, sigmax2;*

*double a1, a2, a0, ryx1, ryx2, rx1x2;*

*// Средние квадратические отклонения.*

*sigmay = sqrt((yyi / 5.0) - ((yi / 5.0) \* (yi / 5.0)));*

*sigmay = round(sigmay \* 1000) / 1000;*

*txtsigmay->Text = Convert::ToString(sigmay);*

*sigmax1 = sqrt((xx1i / 5.0) - ((x1i / 5.0) \* (x1i / 5.0)));*

*sigmax1 = round(sigmax1 \* 1000) / 1000;*

*txtsigmax1->Text = Convert::ToString(sigmax1);*

*sigmax2 = sqrt((xx1i / 5.0) - ((x1i / 5.0) \* (x1i / 5.0)));*

*sigmax2 = round(sigmax2 \* 1000) / 1000;*

*txtsigmax2->Text = Convert::ToString(sigmax2);*

*//r(y, x1);*

*ryx1 = (5 \* x1iyi - x1i \* yi) / sqrt((5 \* xx1i - x1i \* x1i) \* (5 \* yyi - yi \* yi));*

*ryx1 = round(ryx1 \* 1000) / 1000;*

*txtryx1->Text = Convert::ToString(ryx1);*

*//r(y, x2);*

*ryx2 = (5 \* x2iyi - x2i \* yi) / sqrt((5 \* xx2i - x2i \* x2i) \* (5 \* yyi - yi \* yi));*

*ryx2 = round(ryx2 \* 1000) / 1000;*

*txtryx2->Text = Convert::ToString(ryx2);*

*//r(x1, x2);*

*rx1x2 = (5 \* x1ix2i - x1i \* x2i) / sqrt((5 \* xx2i - x2i \* x2i) \* (5 \* xx1i - x1i \* x1i));*

*rx1x2 = round(rx1x2 \* 1000) / 1000;*

*txtrx1x2->Text = Convert::ToString(rx1x2);*

*// Коэффициент корреляции.*

*correlation = sqrt((ryx1 \* ryx1 + ryx2 \* ryx2 - 2 \* ryx1 \* ryx2 \* rx1x2)/(1 - rx1x2 \* rx1x2));*

*correlation = round(correlation \* 1000) / 1000;*

*txtcorrelation->Text = Convert::ToString(correlation);*

*// Коэффициент детерминации.*

*determination = correlation \* correlation;*

*determination = round(determination \* 1000) / 1000;*

*txtdetermination->Text = Convert::ToString(determination);*

*// y = a0 + a1x1 + a2x2.*

*a1 = (sigmay / sigmax1) \* ((ryx1 - ryx2 \* rx1x2) / (1 - rx1x2 \* rx1x2));*

*a1 = round(a1 \* 1000) / 1000;*

*txta1->Text = Convert::ToString(a1);*

*a2 = (sigmay / sigmax2) \* ((ryx2 - ryx1 \* rx1x2) / (1 - rx1x2 \* rx1x2));*

*a2 = round(a2 \* 1000) / 1000;*

*txta2->Text = Convert::ToString(a2);*

*a0 = yi - a1 \* x1i - a2 \* x2i;*

*a0 = round(a0 \* 1000) / 1000;*

*txta0->Text = Convert::ToString(a0);*

*if (a1 > 0.0 && a2 > 0.0) txtrelatex->Text = "y = " + Convert::ToString(a0) + " + " + Convert::ToString(a1) + " \* x1 + " + Convert::ToString(a2) + " \* x2.";*

*else if (a1 < 0.0 && a2 > 0.0) txtrelatex->Text = "y = " + Convert::ToString(a0) + " " + Convert::ToString(a1) + " \* x1 + " + Convert::ToString(a2) + " \* x2.";*

*else if (a1 > 0.0 && a2 < 0.0) txtrelatex->Text = "y = " + Convert::ToString(a0) + " + " + Convert::ToString(a1) + " \* x1 " + Convert::ToString(a2) + " \* x2.";*

*else txtrelatex->Text = "y = " + Convert::ToString(a0) + " " + Convert::ToString(a1) + " \* x1 " + Convert::ToString(a2) + " \* x2.";*

*txtrelateslvl->Text = Convert::ToString(a0 + a1 \* x1i + a2 \* x2i);*

*// Уровень связи:*

*if (abs(correlation) == 0) txtrellvl->Text = "Связь отсутствует.";*

*else if (abs(correlation) > 0 && abs(correlation) <= 0.5) txtrellvl->Text = "Отрезок коэфф. корреляции [0; 0.5). Связь слабая.";*

*else if (abs(correlation) > 0.5 && abs(correlation) <= 0.8) txtrellvl->Text = "Отрезок коэфф. корреляции [0.5; 0.8). Связь умеренная.";*

*else txtrellvl->Text = "Отрезок коэфф. корреляции [0.8; 1]. Связь сильная.";*

*}*

*private: System::Void btncleargraph\_Click(System::Object^ sender, System::EventArgs^ e) {*

*txta1->Text = "";*

*txta2->Text = "";*

*txta0->Text = "";*

*txtrx1x2->Text = "";*

*txtryx2->Text = "";*

*txtryx1->Text = "";*

*txtsigmay->Text = "";*

*txtsigmax1->Text = "";*

*txtsigmax2->Text = "";*

*floppy = 0;*

*s\_floppy = 0;*

*rfloppy = 0;*

*rs\_floppy = 0;*

*tmp\_fl = 0;*

*tableindex = 1;*

*graphic->Series[0]->Points->Clear();*

*txtrelatex->Text = "";*

*txtrelateslvl->Text = "";*

*txtcorrelation->Text = "";*

*txtdetermination->Text = "";*

*temp\_num.clear();*

*temp\_reb.clear();*

*graph\_prima.clear();*

*for (int i = 0; i < datatable->RowCount; i++) {*

*for (int j = 0; j < datatable->ColumnCount; j++) datatable->Rows[i]->Cells[j]->Value = 0;*

*}*

*btnaddvertex->Enabled = true;*

*btncomputinggraph->Enabled = false;*

*}*

*};*

*}*