// Расчет длины и направления вектора (угол и четверть расположения)

int Vector\_id::set\_vector(short num1, short num2) {

short X\_L;

short Y\_L;

if (m\_pPtsX[num1] < m\_pPtsX[num2]) {

X\_L = m\_pPtsX[num2] - m\_pPtsX[num1];

if (m\_pPtsY[num1] < m\_pPtsY[num2]) {

Y\_L = m\_pPtsY[num2] - m\_pPtsY[num1];

corner\_n[num1] = 1;

}

else {

Y\_L = m\_pPtsY[num1] - m\_pPtsY[num2];

corner\_n[num1] = 4;

}

}

else {

X\_L = m\_pPtsX[num1] - m\_pPtsX[num2];

if (m\_pPtsY[num1] < m\_pPtsY[num2]) {

Y\_L = m\_pPtsY[num2] - m\_pPtsY[num1];

corner\_n[num1] = 2;

}

else {

Y\_L = m\_pPtsY[num1] - m\_pPtsY[num2];

corner\_n[num1] = 3;

}

}

Leight[num1] = sqrt((X\_L\*X\_L + Y\_L\*Y\_L)); //длина вектора

corner[num1] = acos(float(X\_L / Leight[num1])) \* 100; // угол

return 1;

};

// Проверка возможной погрешности при распозновании (1- возможна 0-невозможна)

int Vector\_id::redefinition(short num) {

// -------по горизонтали (оХ)

if ((corner[num] < 15) && (corner[num + 1] < 15)) {

if (((corner\_n[num] == 3) && corner\_n[num + 1] == 2) || (corner\_n[num] == 2 && corner\_n[num + 1] == 3)) {

set\_vector(num, num + 1); //перерасчет вектора

return 1;

}

if ((corner\_n[num] == 1 && corner\_n[num + 1] == 4) || (corner\_n[num] == 4 && corner\_n[num + 1] == 1)) {

set\_vector(num, num + 1); //перерасчет вектора

return 1;

}

}

else {

//-------- по вертикали (оУ)

if (corner[num] > 140 && corner[num + 1] > 140) {

if ((corner\_n[num] == 4 && corner\_n[num + 1] == 3) || (corner\_n[num] == 3 && corner\_n[num + 1] == 4)) {

set\_vector(num, num + 1);//перерасчет вектора

return 1;

}

if ((corner\_n[num] == 2 && corner\_n[num + 1] == 1) || (corner\_n[num] == 1 && corner\_n[num + 1] == 2)) {

set\_vector(num, num + 1);//перерасчет вектора

return 1;

}

}

}

return 0;

};

// Нахождение диагонали подписи

int Vector\_id::SetL\_S(short& Leight\_S) {

int Xmin, Xmax, Ymin, Ymax;

GetBox(Xmin, Xmax, Ymin, Ymax);

short X\_r = Xmax - Xmin;

short Y\_r = Ymax - Ymin;

Leight\_S = sqrt(X\_r\*X\_r + Y\_r\*Y\_r);

return 1;

}

// Нахождение размеров подписи (максимальные и минимальные позиции на ох/оу)

int Vector\_id::GetBox(int& iMinX, int& iMaxX, int& iMinY, int& iMaxY)

{

if (!m\_shPoints)

return 0;

iMinX = iMaxX = m\_pPtsX[0];

iMinY = iMaxY = m\_pPtsY[0];

for (int iPoint = 0; iPoint < m\_shPoints; iPoint++)

{

int iX = m\_pPtsX[iPoint];

int iY = m\_pPtsY[iPoint];

if (iY == -1)

continue;

if (iX > iMaxX)

iMaxX = iX;

else if (iX < iMinX)

iMinX = iX;

if (iY > iMaxY)

iMaxY = iY;

else if (iY < iMinY)

iMinY = iY;

}

return 1;

}

// Нахождения коэффициента совпадения

int find\_k(Vector\_id& vec1, Vector\_id& vec2) {

// Размер подписи

short leight1(0);

short leight2(0);

// Получение размера

vec1.SetL\_S(leight1);

vec2.SetL\_S(leight2);

short lin\_re\_num(0); //Количество возможных погрешностей распознования

short m\_shP(0);// определяем количество сверяемых векторов

// добавляем шетчик изменения позиционирования сверяемых пар векторов

short m\_sh1(0);

short m\_sh2(0);

//Определяем подпись содержашую минимум векторов и различие в них

if (vec1.m\_shPoints > vec2.m\_shPoints) {

vec2.origin = true;

lin\_re\_num = vec1.m\_shPoints - vec2.m\_shPoints;

m\_shP = vec2.m\_shPoints;

}

else {

vec1.origin = true;

lin\_re\_num = vec2.m\_shPoints - vec1.m\_shPoints;

m\_shP = vec1.m\_shPoints - 1;

}

float\* k\_mass; // Масив идентичности

k\_mass = new float[m\_shP];

//резервирование памяти под размеры (векторов)

float v1\_L(0);

float v2\_L(0);

bool N;//определяет соответствие направлений

//------------------------------Проверка совпадения векторов

for (short num(0); num < m\_shP; num++) {

//------------------------------------если направления углов совпадают

if (vec1.corner\_n[num + m\_sh1] == vec2.corner\_n[num + m\_sh2]) {

// если возможна погрешность

if (lin\_re\_num > 0) {

// определение возможного дробления вектора (2 движения в одном направлении нешитаемые одним движением)

if (vec2.origin == true) {

//преобразование 1го вектора

if (vec1.corner\_n[num + m\_sh1] == vec1.corner\_n[num + m\_sh1 + 1] == vec2.corner\_n[num + m\_sh2] != vec2.corner\_n[num + m\_sh2 + 1])

vec1.set\_vector(num + m\_sh1, num + m\_sh1 + 1);//Создаем новый вектор

m\_sh1++;

lin\_re\_num--;

}

else {

//преобразование 2го вектора

if (vec2.corner\_n[num + m\_sh2] == vec2.corner\_n[num + m\_sh2 + 1] == vec1.corner\_n[num + m\_sh1] != vec1.corner\_n[num + m\_sh1 + 1])

vec2.set\_vector(num + m\_sh2, num + m\_sh2 + 1);//Создаем новый вектор

m\_sh2++;

lin\_re\_num--;

}

}

N = true;

// -----------------------Если направления углов несовпадают

} else {

// если возможна погрешность

if (lin\_re\_num > 0) {

if (vec1.origin == true) {

// Если возможно что направления несовпадают изза погрешности распознования 2 вектора

if (vec2.redefinition(num + m\_sh2) == 1) {

lin\_re\_num--; // Количество допустимых погрешностей

m\_sh2++;

N = true;

}

else {

k\_mass[num] = 0;

N = false;

}

} else {

// Если возможно что направления несовпадают изза погрешности распознования 1 вектора

if (vec1.redefinition(num + m\_sh1) == 1) {

lin\_re\_num--; // Количество допустимых погрешностей

m\_sh1++;

N = true;

}

else {

k\_mass[num] = 0;

N = false;

}

}

}

//-----------------------------------------------

}

//--------------если вектора могут совпадать

if (N) {

v1\_L = float(vec1.Leight[num + m\_sh1]) / leight1;

v2\_L = float(vec2.Leight[num + m\_sh2]) / leight2;

// отношение размеров

if (v1\_L <= v2\_L) {

k\_mass[num] = (v1\_L / v2\_L);

}

else {

k\_mass[num] = (v2\_L / v1\_L);

}

// Сравнение углов

if (vec1.corner[num] <= vec2.corner[num]) {

k\_mass[num] += float(vec1.corner[num]) / vec2.corner[num];

}

else {

k\_mass[num] += float(vec2.corner[num]) / vec1.corner[num];

}

k\_mass[num] = (k\_mass[num] / 2); //Получение результирующего коэффициента совпадения для данного участка.

}

}

// Итоговый коэффициент совпадения

float K\_res(0);

for (short num\_end(0); num\_end < m\_shP; num\_end++) {

K\_res = K\_res + k\_mass[num\_end];

}

K\_res = K\_res / m\_shP;

delete[] k\_mass;

return K\_res;

};