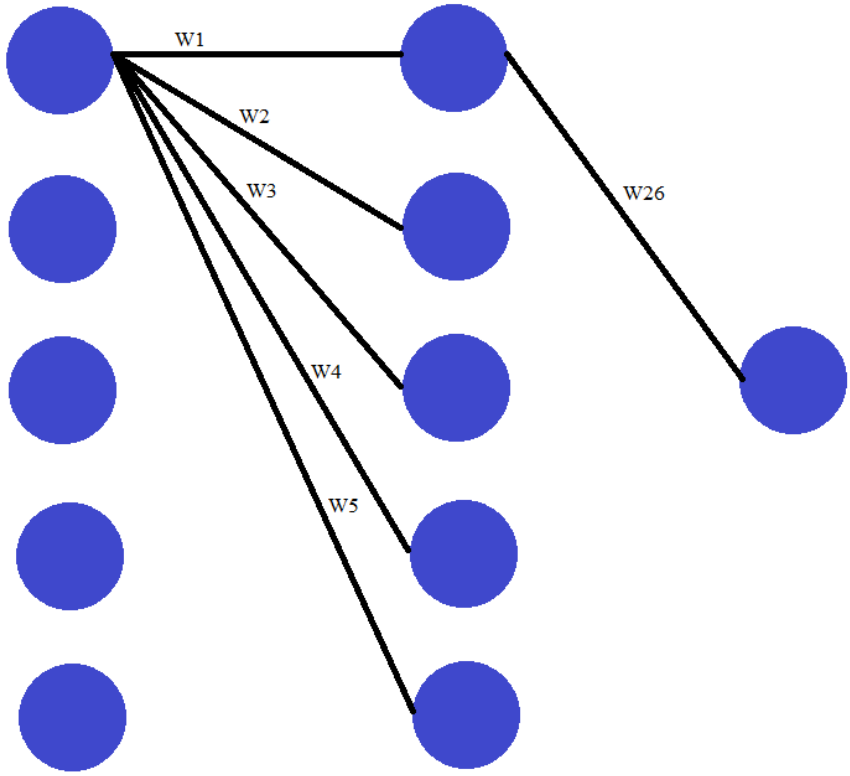
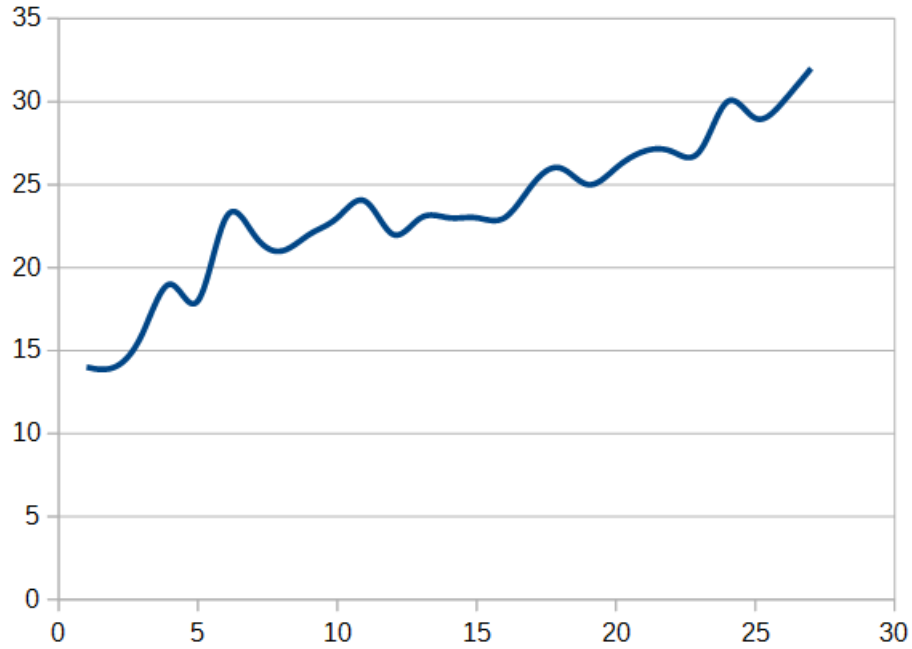
```cpp
#include "stdafx.h"
#include <iostream>
#include <vector>
#include <time.h>
#include <cmath>
#include <fstream>

using namespace std;

const long double speed = 0.99;
const long double Em = 0.0001;

const int n = 5;
const int m = 5;

const int len = 32;

const long double e = 2.71828;

const int iterations = 100;


int _tmain(int argc, _TCHAR* argv[])
{
	setlocale(LC_ALL, "Russian");
	srand(time(0));
	ofstream fout;
	fout.open("D:\\Test.txt");
	//dataset inicialisation
	vector<vector<double>> learn;
	vector<bool> answers;
	srand(time(0));
	for (int i = 0; i < len; i++) {
		vector<double> temp;
		int val = 0;
		for (int j = 0; j < 6; j++) {
			temp.push_back(rand() % 101 / 100);
			if (rand() % 2 == 0) temp[j] = -temp[j];
			else val++;
		}
		learn.push_back(temp);
		if (val % 2 == 0) answers.push_back(true);
		else answers.push_back(false);
	}

	vector<vector<double>> population;
	vector<int> fitness;
	for (int i = 0; i < 100; i++) {
		vector<double> gene;
		vector<vector<long double>> w1;
		for (int i = 0; i < n; i++) {
			vector<long double> temp;
			for (int j = 0; j < m; j++) {
				temp.push_back(rand() / (RAND_MAX + 1.));
				gene.push_back(temp[j]);
			}
			w1.push_back(temp);
		}
		vector<long double> w2;
		vector<long double>  T1;
		for (int i = 0; i < m; i++) {
			w2.push_back(rand() / (RAND_MAX + 1.));
			T1.push_back(rand() / (RAND_MAX + 1.));
			gene.push_back(w2[i]);
		}
```

```
		for (int i = 0; i < m; i++) {
			gene.push_back(T1[i]);
		}
		long double  T2 = rand() / (RAND_MAX + 1.);
		gene.push_back(T2);
		population.push_back(gene);

		int v = 0;
		vector<double> y;
			// Получение выхода нейронов и сети в целом
			for (int i = 0; i < len; i++) {
				vector<vector<long double>> s1;
				vector<long double> s2, res1;
				for (int u = 0, j = n; u < n; u++, j--) {
					vector<long double> temp;
					for (int k = 0; k < m; k++) {
						temp.push_back(w1[u][k] * learn[i][u]);
					}
					s1.push_back(temp);
				}
				for (int u = 0; u < m; u++) {
					long double temp = 0;
					for (int k = 0; k < n; k++) {
						temp += s1[k][u];
					}
					temp -= T1[u];
					temp = 1 / (1 + pow(e, -temp)); // выход нейронов
скрытого слоя
					res1.push_back(temp);
				}
				vector<long double> Fs;
				for (int u = 0; u < m; u++) {
					Fs.push_back(res1[u] * (1 - res1[u]));
				}
				for (int k = 0; k < m; k++) {
					s2.push_back(w2[k] * res1[k]);
				}
				long double temp = 0;
				for (int k = 0; k < m; k++) {
					temp += s2[k];
				}
				temp -= T2; // выход сети
				y.push_back(temp);
			}
			int error = 0;
			for (int i = 0; i < answers.size(); i++) {
				if (y[i] < 0 && !answers[i] || y[i] >= 0 && answers[i])
error++;
			}
			fitness.push_back(len - error);

	}
	int number = 0;
	while (true) {
		number++;
		vector<vector<double>> parents;
		int mid = 0;
		for (int i = 0; i < iterations; i++) {
			mid += fitness[i];
		}
		mid /= iterations;
		for (int i = 0; i < 15; i++) {
			if (fitness[i] >= mid) parents.push_back(population[i]);
		}
		population.clear();
```

```
		fitness.clear();
		cout << "Population " << number + 1 << endl;
		for (int i = 0; i < iterations; i++) {
			int parent1 = rand() % parents.size();
			int parent2 = rand() % parents.size();
			while (true) {
				if (parent1 == parent2) parent2 = rand() % parents.size();
				else break;
			}
			int cross = rand() % parents[parent1].size();
			vector<double> child;
			for (int i = 0; i < parents[parent1].size(); i++) {
				child.push_back(parents[parent1][i] * 2 + parents[parent2][i]
/ 4);
			}
			population.push_back(child);
			int point = rand() % child.size();
			child[point] = rand() / (RAND_MAX + 1.);
			int f = 0;
				vector<vector<long double>> w1;
				for (int i = 0; i < n; i++) {
					vector<long double> temp;
					for (int j = 0; j < m; j++) {
						temp.push_back(child[f]);
						f++;
					}
					w1.push_back(temp);
				}
				vector<long double> w2;
				vector<long double>  T1;
				for (int i = 0; i < m; i++) {
					w2.push_back(child[f]);
					f++;
				}
				for (int i = 0; i < m; i++) {
					T1.push_back(child[f]);
					f++;
				}
				long double  T2 = child[f];

			int v = 0;
			vector<double> y;
			// Получение выхода нейронов и сети в целом
			for (int i = 0; i < len; i++) {
				vector<vector<long double>> s1;
				vector<long double> s2, res1;
				for (int u = 0, j = n; u < n; u++, j--) {
					vector<long double> temp;
					for (int k = 0; k < m; k++) {
						temp.push_back(w1[u][k] * learn[i][u]);
					}
					s1.push_back(temp);
				}
				for (int u = 0; u < m; u++) {
					long double temp = 0;
					for (int k = 0; k < n; k++) {
						temp += s1[k][u];
					}
					temp -= T1[u];
					temp = 1 / (1 + pow(e, -temp)); // выход нейронов
скрытого слоя
					res1.push_back(temp);
				}
				vector<long double> Fs;
				for (int u = 0; u < m; u++) {
```

```
                                        Fs.push_back(res1[u] * (1 - res1[u]));
                                }
                                for (int k = 0; k < m; k++) {
                                        s2.push_back(w2[k] * res1[k]);
                                }
                                long double temp = 0;
                                for (int k = 0; k < m; k++) {
                                        temp += s2[k];
                                }
                                temp -= T2; // выход сети
                                bool res;
                                y.push_back(temp);
                        }
                        int error = 0;
                        for (int i = 0; i < answers.size(); i++) {
                                if (y[i] < 0 && !answers[i] || y[i] >= 0 && answers[i])
error++;
                        }
                        fitness.push_back(len - error);
                        cout << len - error << "    ";

                }
                bool compl = false;
                for (int i = 0; i < fitness.size(); i++) {
                        if (fitness[i] == iterations) compl = true;
                }
                cout << endl << endl;
                if (compl == true) break;
                if (number == 10000) break;
        }
        fout.close();
        return 0;
}
```





X – number of iteration

Y – best fitness in generation

Values:

W1	-0.0807887385975517
W2	-0.360210900362679
W3	-0.221398853334318
W4	0.969727301490366
W5	0.934620120066507
W6	0.00298415217687569
W7	-0.676656131482057
W8	-0.0981439766931087
W9	-0.447110412850562
W10	0.131837159456609
W11	-0.595922192370483
W12	0.930679468405749
W13	0.564169297723178
W14	0.987798357842396
W15	0.0474480120686106
W16	-0.687640359014105
W17	-0.828826693738264
W18	0.761352861188982
W19	0.0500075295800378
W20	-0.699063556128677
W21	0.0303105195194066
W22	0.421785137812507
W23	-0.908295810645584
W24	0.355323450805304
W25	0.868160644019097
W26	0.742539797789669
W27	0.839499257895862
W28	0.263068553182794

W29	0.595387210415391
W30	-0.650466014002667