

```
#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 = 0.000001;

const int n = 6;
const int m = 6;

const int len = 64;

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
            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 < iterations; 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++) {
            if (i < cross) child.push_back(parents[parent1][i]);
            else child.push_back(parents[parent2][i]);
        }
        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
                res1.push_back(temp);
            }
        }
    }
}

```

скрытого слоя

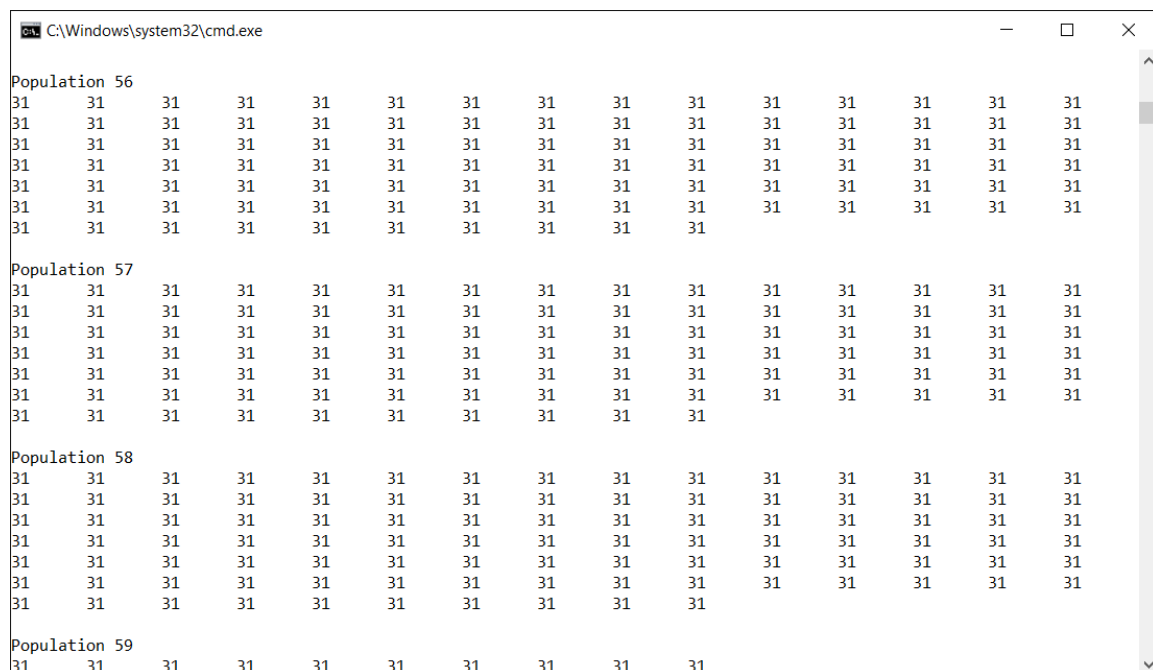
```

    }
    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(100 - 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;
}

```

This neural network doesn't work correctly.



It can recognize only 28-35 figures of 64.