```cpp
#include <iostream>
#include <fstream>
#include <vector>
#include <time.h>
#include <math.h>
#include <algorithm>

using namespace std;

typedef vector<int> Chromosome;
typedef vector<Chromosome> Population;

Population population;
Population population2;

Population makePopulation();
double fitness(Chromosome chromosome);
double s_function(Chromosome chromosome);
bool sortPopulation(const Chromosome &chromosome1, const Chromosome &chromosome2);
Population newPopulation();
double distance(const Chromosome &chromosome1, int k);
double sharing_fitness(Chromosome chromosome);

int main()
{
	srand(time(NULL));
	ofstream file("OutData.txt", ios::out);

	population = makePopulation();
	population2 = population;
	sort(population.begin(), population.end(), sortPopulation);
	population2 = population;

	double aver = 0.0;
	for (int noChanged = 0, k = 0; noChanged < 30 && k < 500; k++)
	{
		population = newPopulation();
		sort(population.begin(), population.end(), sortPopulation);
		population2 = population;

		double max = sharing_fitness(population[0]);
		double average = 0.0;
		for (int i = 0; i < 20; i++)
		{
			average += sharing_fitness(population[i]);
		}
		average = average / 20;
		if (average == aver)
		{
			noChanged++;
		}
		else
		{
			noChanged = 0;
		}
		aver = average;

		file << max << "\t" << average << endl;
	}

	file.close();

	system("pause");
	return 0;
}
```

```
Population makePopulation()
{
	Population result;
	for (int i = 0; i < 20; i++)
	{
		Chromosome chromosome;
		for (int j = 0; j < 22; j++)
		{
			chromosome.push_back(rand() % 2);
		}
		result.push_back(chromosome);
	}
	return result;
}

double fitness(Chromosome chromosome)
{
	int sum = 0;
	for (int i = 1; i < 11; i++)
	{
		sum += chromosome[i] * pow(2, (i - 1));
	}
	double x = sum / 1023.0 * 5.0;
	if (chromosome[0] == 0)
	{
		x = x * (-1.0);
	}

	sum = 0;
	for (int i = 12; i < 22; i++)
	{
		sum += chromosome[i] * pow(2, (i - 12));
	}
	double y = sum / 1023.0 * 5.0;
	if (chromosome[11] == 0)
	{
		y = y * (-1.0);
	}

	double z = x * sin(abs(x)) + y * sin(abs(y));
	return z;
}

double distance(const Chromosome &chromosome1, int k)
{
	int sum = 0;
	for (int i = 1; i < 11; i++)
	{
		sum += chromosome1[i] * pow(2, (i - 1));
	}
	double x1 = sum / 1023.0 * 5.0;
	if (chromosome1[0] == 0)
	{
		x1 = x1 * (-1.0);
	}

	sum = 0;
	for (int i = 12; i < 22; i++)
	{
		sum += chromosome1[i] * pow(2, (i - 12));
	}
	double y1 = sum / 1023.0 * 5.0;
	if (chromosome1[11] == 0)
	{
```

```
                y1 = y1 * (-1.0);
        }

        sum = 0;
        for (int i = 1; i < 11; i++)
        {
                sum += population2[k][i] * pow(2, (i - 1));
        }
        double x2 = sum / 1023.0 * 5.0;
        if (population2[k][0] == 0)
        {
                x2 = x2 * (-1.0);
        }

        sum = 0;
        for (int i = 12; i < 22; i++)
        {
                sum += population2[k][i] * pow(2, (i - 12));
        }
        double y2 = sum / 1023.0 * 5.0;
        if (population2[k][11] == 0)
        {
                y2 = y2 * (-1.0);
        }

        double dist = sqrt((x1 - x2)*(x1 - x2) + (y1 - y2)*(y1 - y2));
        return dist;
}

double s_function(Chromosome chromosome)
{
        double sigma = 1.0;
        double s = 0.0;

        for (int i = 0; i < population.size(); i++)
        {
                double d = distance(chromosome, i);
                if (d < sigma)
                {
                        s += (1 - (d / sigma));
                }
        }

        return s;
}

double sharing_fitness(Chromosome chromosome)
{
        double f = fitness(chromosome) / s_function(chromosome);
        return f;
}

bool sortPopulation(const Chromosome &chromosome1, const Chromosome &chromosome2)
{
        return sharing_fitness(chromosome1) > sharing_fitness(chromosome2);
}

Population newPopulation()
{
        int firstParent, secondParent;
        Population result;

        for (int i = 0; i < 10; i++)
        {
                firstParent = rand() % 10;
```

```
            secondParent = rand() % 10;
            while (firstParent != secondParent)
            {
                  secondParent = rand() % 10;
            }

            Chromosome child1, child2;
            for (int j = 0; j < 22; j++)
            {
                  int mask = rand() % 2;
                  if (mask == 0)
                  {
                        child1.push_back(population[firstParent][j]);
                        child2.push_back(population[secondParent][j]);
                  }
                  else
                  {
                        child1.push_back(population[secondParent][j]);
                        child2.push_back(population[firstParent][j]);
                  }
            }

            result.push_back(child1);
            result.push_back(child2);
      }

      return result;
}
```