My program is written on C++. Program performs 500 iterations or it will stop when the average fitness of the chromosomes do not change for 30 times.

Code:

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

using namespace std;

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

Population makePopulation();
int fitness(const Chromosome &chromosome);
bool sortPopulation(const Chromosome &chromosome1, const Chromosome &chromosome2);
Population newPopulation(Population population, bool flag);

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

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

	int aver = 0;
	for (int noChanged = 0, k = 0; noChanged < 30 && k < 500; k++)
	{
		if (k % 10 == 0)
		{
			population = newPopulation(population, true);
		}
		else
		{
			population = newPopulation(population, false);
		}
		sort(population.begin(), population.end(), sortPopulation);

		int max = fitness(population[0]);
		if (max == 1000)
		{
			break;
		}

		int average = 0;
		for (int i = 0; i < population.size(); i++)
		{
			average += fitness(population[i]);
		}
		average = average / 100;
		if (average == aver)
		{
			noChanged++;
		}
		else
		{
			noChanged = 0;
		}
```

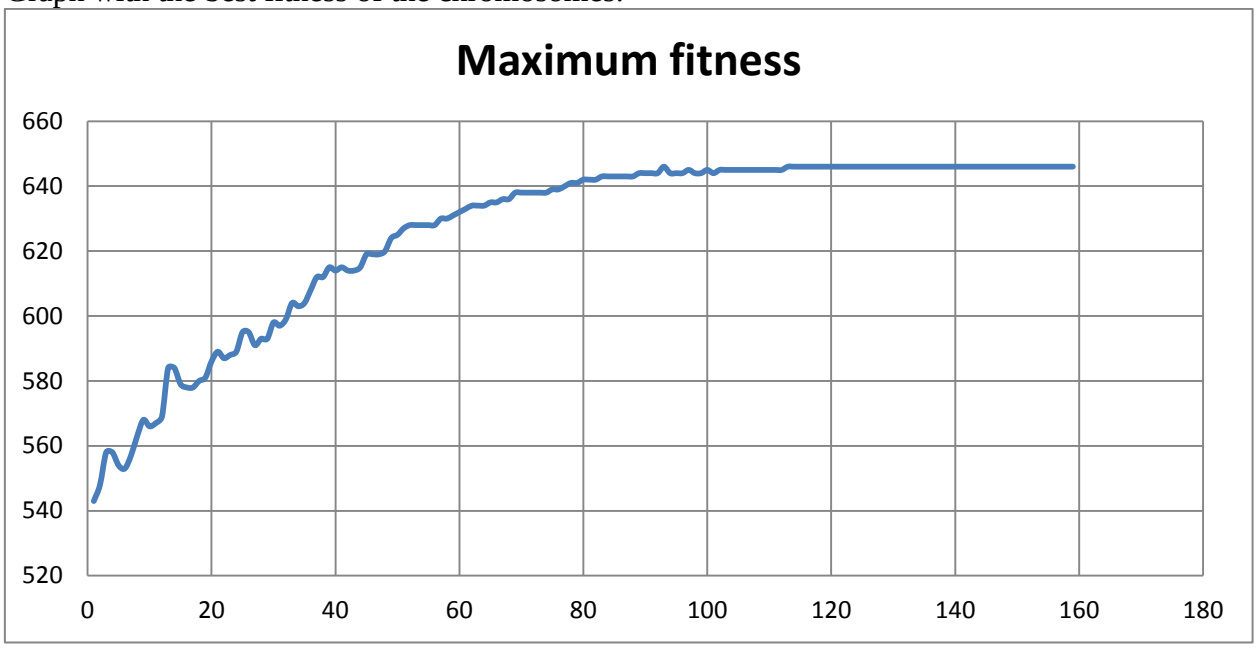
```
			aver = average;

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

		for (int j = 0; j < Table[0].size(); j++)
		{
			for (int i = 0; i < Table.size(); i++)
			{
				fileTable << Table[i][j] << "\t\t";
			}
			fileTable << endl;
		}

		file.close();
		fileTable.close();

		system("pause");
		return 0;
}

Population makePopulation()
{
		Population population;
		for (int i = 0; i < 100; i++)
		{
			Chromosome chromosome;
			for (int j = 0; j < 1000; j++)
			{
				chromosome.push_back(rand() % 2);
			}
			population.push_back(chromosome);
		}
		return population;
}

int fitness(const Chromosome &chromosome)
{
		int summ = 0;
		for (int i = 0; i < chromosome.size(); i++)
		{
			summ += chromosome[i];
		}
		return summ;
}

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

Population newPopulation(Population population, bool flag)
{
		Population result;
		vector<int> fit;
		vector<double> prob;
		vector<int> count(100, 0);

		int summ = 0;
		for (int i = 0; i < population.size(); i++)
		{
			fit.push_back(fitness(population[i]));
			summ += fitness(population[i]);
		}
		for (int i = 0; i < fit.size(); i++)
```

```
	{
		double tmp = (double)fit[i] / summ;
		prob.push_back(tmp);
	}

	for (int k = 0; k < 50; k++)
	{
		int tmp = rand() % 8 * 100000 + rand() % 10 * 10000 + rand() % 10 * 1000 +
rand() % 10 * 100 + rand() % 10 * 10 + rand() % 10;
		double val1 = (double)tmp / 1000000;
		double val2 = (double)tmp / 1000000;
		while (val1 == val2)
		{
			tmp = rand() % 8 * 100000 + rand() % 10 * 10000 + rand() % 10 * 1000
+ rand() % 10 * 100 + rand() % 10 * 10 + rand() % 10;
			val2 = (double)tmp / 1000000;
		}

		double summProb = 0.0;
		int parent1, parent2;
		for (int i = 0; i < prob.size(); i++)
		{
			summProb += prob[i];
			if (summProb >= val1)
			{
				parent1 = i;
				count[i] += 1;
				break;
			}
		}
		summProb = 0.0;
		for (int i = 0; i < prob.size(); i++)
		{
			summProb += prob[i];
			if (summProb >= val2)
			{
				parent2 = i;
				count[i] += 1;
				break;
			}
		}

		Chromosome child1 = population[parent1];
		Chromosome child2 = population[parent2];
		int cut = rand() % 1000;
		for (int i = cut; i < 1000; i++)
		{
			child1[i] = population[parent2][i];
			child2[i] = population[parent1][i];
		}
		result.push_back(child1);
		result.push_back(child2);
	}

	if (flag)
	{
		vector<double> fitTable;
		vector<double> probTable;
		vector<double> countTable;
		for (int i = 0; i < fit.size(); i++)
		{
			fitTable.push_back(fit[i]);
			probTable.push_back(prob[i]);
			countTable.push_back(count[i]);
		}
```

```
            Table.push_back(fitTable);
            Table.push_back(probTable);
            Table.push_back(countTable);
        }

        return result;
}
```

After analyzing the results, we see that the algorithm has created a chromosome with 64,6% of good genes. The maximum and average values are equal, and received for 159 iterations. The result also depends on the generation of a random initial population.

Graph with the best fitness of the chromosomes:



Graph with the average fitness of the chromosomes:

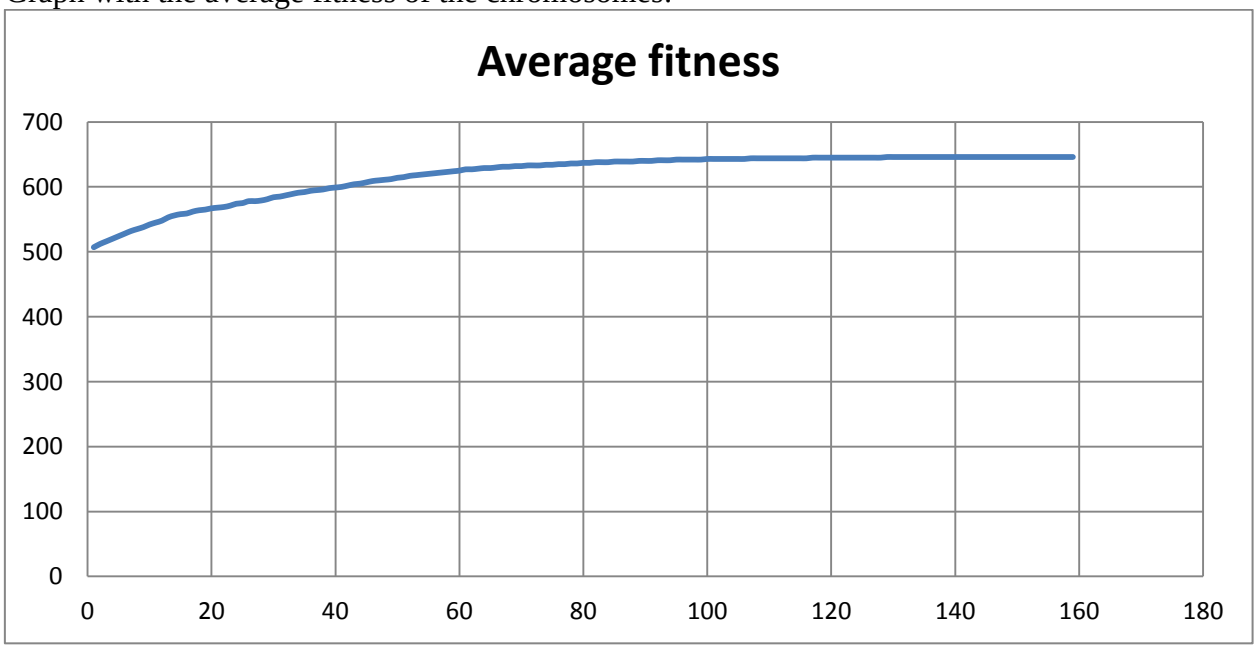


Table with fitness, probability and a how-often-selected

30 iteration			60 iteration			90 iteration			120 iteration			150 iteration		
fit	prob	c	fit	prob	c	fit	prob	c	fit	prob	c	fit	prob	c
598	0.0102331	2	632	0.0100971	1	644	0.0100543	1	646	0.010014	0	646	0.01	1
596	0.0101988	2	631	0.0100812	3	643	0.0100387	1	646	0.010014	0	646	0.01	1
593	0.0101475	0	631	0.0100812	2	642	0.0100231	3	646	0.010014	1	646	0.01	0
593	0.0101475	1	631	0.0100812	1	642	0.0100231	0	646	0.010014	1	646	0.01	0
592	0.0101304	0	631	0.0100812	0	642	0.0100231	1	646	0.010014	1	646	0.01	1
591	0.0101133	0	630	0.0100652	1	642	0.0100231	2	646	0.010014	2	646	0.01	2
591	0.0101133	2	630	0.0100652	1	642	0.0100231	1	646	0.010014	3	646	0.01	2
591	0.0101133	1	630	0.0100652	1	642	0.0100231	1	646	0.010014	0	646	0.01	0
590	0.0100962	0	630	0.0100652	2	642	0.0100231	4	646	0.010014	3	646	0.01	2
590	0.0100962	2	630	0.0100652	1	642	0.0100231	0	646	0.010014	2	646	0.01	1
590	0.0100962	0	630	0.0100652	2	642	0.0100231	2	645	0.00999845	2	646	0.01	1
590	0.0100962	0	629	0.0100492	0	642	0.0100231	0	645	0.00999845	0	646	0.01	2
590	0.0100962	2	629	0.0100492	1	642	0.0100231	1	645	0.00999845	1	646	0.01	1
589	0.0100791	0	629	0.0100492	0	642	0.0100231	3	645	0.00999845	1	646	0.01	0
589	0.0100791	3	629	0.0100492	6	642	0.0100231	1	645	0.00999845	3	646	0.01	2
588	0.0100619	0	629	0.0100492	0	642	0.0100231	1	645	0.00999845	1	646	0.01	2
588	0.0100619	1	629	0.0100492	1	642	0.0100231	0	645	0.00999845	0	646	0.01	4

588	0.0100619	1	628	0.0100332	2	642	0.0100231	1	645	0.00999845	1	646	0.01	1
588	0.0100619	0	628	0.0100332	1	642	0.0100231	1	645	0.00999845	3	646	0.01	1
588	0.0100619	2	628	0.0100332	4	642	0.0100231	1	645	0.00999845	3	646	0.01	3
588	0.0100619	1	628	0.0100332	2	642	0.0100231	1	645	0.00999845	0	646	0.01	4
588	0.0100619	3	628	0.0100332	3	642	0.0100231	2	645	0.00999845	1	646	0.01	1
587	0.0100448	1	628	0.0100332	1	642	0.0100231	0	645	0.00999845	2	646	0.01	2
587	0.0100448	1	628	0.0100332	4	642	0.0100231	2	645	0.00999845	1	646	0.01	0
587	0.0100448	2	628	0.0100332	2	641	0.0100075	1	645	0.00999845	2	646	0.01	0
587	0.0100448	1	627	0.0100173	1	641	0.0100075	2	645	0.00999845	2	646	0.01	1
587	0.0100448	2	627	0.0100173	1	641	0.0100075	1	645	0.00999845	1	646	0.01	1
587	0.0100448	1	627	0.0100173	0	641	0.0100075	1	645	0.00999845	0	646	0.01	1
587	0.0100448	2	627	0.0100173	2	641	0.0100075	2	645	0.00999845	2	646	0.01	2
587	0.0100448	0	627	0.0100173	0	641	0.0100075	0	645	0.00999845	1	646	0.01	2
587	0.0100448	1	627	0.0100173	4	641	0.0100075	0	645	0.00999845	0	646	0.01	2
587	0.0100448	2	627	0.0100173	1	641	0.0100075	0	645	0.00999845	2	646	0.01	0
586	0.0100277	1	627	0.0100173	1	641	0.0100075	3	645	0.00999845	0	646	0.01	2
586	0.0100277	1	627	0.0100173	0	641	0.0100075	2	645	0.00999845	1	646	0.01	0
586	0.0100277	3	627	0.0100173	0	641	0.0100075	2	645	0.00999845	1	646	0.01	1
586	0.0100277	1	627	0.0100173	0	641	0.0100075	0	645	0.00999845	1	646	0.01	0

586	0.0100277	2	627	0.0100173	3	641	0.0100075	0	645	0.00999845	1	646	0.01	0
586	0.0100277	1	627	0.0100173	1	641	0.0100075	1	645	0.00999845	2	646	0.01	0
586	0.0100277	1	627	0.0100173	1	641	0.0100075	3	645	0.00999845	4	646	0.01	0
586	0.0100277	0	626	0.0100013	0	641	0.0100075	2	645	0.00999845	0	646	0.01	0
585	0.0100106	4	626	0.0100013	2	641	0.0100075	0	645	0.00999845	0	646	0.01	1
585	0.0100106	1	626	0.0100013	0	641	0.0100075	1	645	0.00999845	2	646	0.01	4
585	0.0100106	1	626	0.0100013	2	640	0.0099 9188	5	645	0.00999845	1	646	0.01	1
585	0.0100106	3	626	0.0100013	0	640	0.0099 9188	0	645	0.00999845	0	646	0.01	2
585	0.0100106	1	626	0.0100013	1	640	0.0099 9188	0	645	0.00999845	0	646	0.01	0
585	0.0100106	0	626	0.0100013	0	640	0.0099 9188	2	645	0.00999845	1	646	0.01	1
585	0.0100106	2	626	0.0100013	0	640	0.0099 9188	1	645	0.00999845	2	646	0.01	2
585	0.0100106	1	626	0.0100013	2	640	0.0099 9188	0	645	0.00999845	1	646	0.01	0
585	0.0100106	1	626	0.0100013	0	640	0.0099 9188	1	645	0.00999845	1	646	0.01	1
584	0.0099935	4	626	0.0100013	0	640	0.0099 9188	0	645	0.00999845	3	646	0.01	2
584	0.0099935	0	626	0.0100013	3	640	0.0099 9188	3	645	0.00999845	1	646	0.01	2
584	0.0099935	1	626	0.0100013	1	640	0.0099 9188	2	645	0.00999845	0	646	0.01	2
584	0.0099935	3	626	0.0100013	0	640	0.0099 9188	1	645	0.00999845	0	646	0.01	0
584	0.0099935	0	626	0.0100013	1	640	0.0099 9188	0	645	0.00999845	0	646	0.01	1
584	0.0099935	0	626	0.0100013	0	640	0.0099 9188	0	645	0.00999845	2	646	0.01	4

584	0.0099935	1	626	0.0100013	2	640	0.00999188	0	645	0.00999845	0	646	0.01	3
584	0.0099935	1	626	0.0100013	1	640	0.00999188	0	645	0.00999845	1	646	0.01	1
584	0.0099935	3	625	0.0099853	0	640	0.00999188	1	645	0.00999845	0	646	0.01	2
584	0.0099935	1	625	0.0099853	1	640	0.00999188	0	645	0.00999845	1	646	0.01	3
584	0.0099935	1	625	0.0099853	0	640	0.00999188	2	645	0.00999845	2	646	0.01	0
583	0.0099763	1	625	0.0099853	1	640	0.00999188	0	645	0.00999845	2	646	0.01	1
583	0.0099763	2	625	0.0099853	1	640	0.00999188	1	645	0.00999845	2	646	0.01	0
583	0.0099763	0	625	0.0099853	1	640	0.00999188	0	645	0.00999845	2	646	0.01	0
583	0.0099763	0	625	0.0099853	1	640	0.00999188	3	645	0.00999845	0	646	0.01	0
583	0.0099763	2	625	0.0099853	2	640	0.00999188	3	645	0.00999845	3	646	0.01	2
583	0.0099763	1	625	0.0099853	1	640	0.00999188	2	645	0.00999845	4	646	0.01	1
583	0.0099763	3	625	0.0099853	2	640	0.00999188	1	645	0.00999845	1	646	0.01	0
583	0.0099763	1	625	0.0099853	0	640	0.00999188	0	645	0.00999845	0	646	0.01	3
582	0.0099592	3	625	0.0099853	4	640	0.00999188	2	645	0.00999845	0	646	0.01	0
582	0.0099592	3	625	0.0099853	3	640	0.00999188	4	645	0.00999845	0	646	0.01	5
582	0.0099592	0	625	0.0099853	1	640	0.00999188	1	645	0.00999845	3	646	0.01	1
582	0.0099592	3	624	0.0099693	2	640	0.00999188	2	645	0.00999845	1	646	0.01	0
582	0.0099592	3	624	0.0099693	1	640	0.00999188	2	645	0.00999845	1	646	0.01	2
582	0.0099592	0	624	0.0099693	2	640	0.00999188	1	645	0.00999845	1	646	0.01	0

582	0.00995927	1	624	0.00996933	1	640	0.00999188	2	645	0.00999845	1	646	0.01	1
582	0.00995927	1	624	0.00996933	1	640	0.00999188	0	645	0.00999845	2	646	0.01	1
581	0.00994216	0	624	0.00996933	1	640	0.00999188	2	645	0.00999845	2	646	0.01	2
581	0.00994216	0	624	0.00996933	0	640	0.00999188	3	645	0.00999845	2	646	0.01	1
581	0.00994216	1	624	0.00996933	1	640	0.00999188	1	645	0.00999845	1	646	0.01	1
581	0.00994216	0	624	0.00996933	1	640	0.00999188	1	645	0.00999845	2	646	0.01	1
580	0.00992505	0	623	0.00995335	0	640	0.00999188	0	645	0.00999845	0	646	0.01	0
580	0.00992505	0	623	0.00995335	0	640	0.00999188	0	645	0.00999845	0	646	0.01	0
580	0.00992505	0	623	0.00995335	0	640	0.00999188	0	645	0.00999845	0	646	0.01	0
580	0.00992505	0	623	0.00995335	0	640	0.00999188	0	645	0.00999845	0	646	0.01	0
580	0.00992505	0	623	0.00995335	0	640	0.00999188	0	645	0.00999845	0	646	0.01	0
579	0.00990794	0	623	0.00995335	0	639	0.00997627	0	645	0.00999845	0	646	0.01	0
579	0.00990794	0	623	0.00995335	0	639	0.00997627	0	645	0.00999845	0	646	0.01	0
579	0.00990794	0	623	0.00995335	0	639	0.00997627	0	645	0.00999845	0	646	0.01	0
579	0.00990794	0	623	0.00995335	0	639	0.00997627	0	645	0.00999845	0	646	0.01	0
579	0.00990794	0	623	0.00995335	0	639	0.00997627	0	645	0.00999845	0	646	0.01	0
578	0.00989082	0	622	0.00993737	0	639	0.00997627	0	645	0.00999845	0	646	0.01	0
578	0.00989082	0	622	0.00993737	0	639	0.00997627	0	645	0.00999845	0	646	0.01	0
577	0.00987371	0	622	0.00993737	0	639	0.00997627	0	645	0.00999845	0	646	0.01	0

577	0.0098 7371	0	622	0.00993 737	0	639	0.0099 7627	0	645	0.00999 845	0	646	0.01	0
576	0.0098 566	0	622	0.00993 737	0	639	0.0099 7627	0	645	0.00999 845	0	646	0.01	0
576	0.0098 566	0	622	0.00993 737	0	639	0.0099 7627	0	645	0.00999 845	0	646	0.01	0
576	0.0098 566	0	621	0.00992 14	0	639	0.0099 7627	0	645	0.00999 845	0	646	0.01	0
576	0.0098 566	0	621	0.00992 14	0	639	0.0099 7627	0	645	0.00999 845	0	646	0.01	0
576	0.0098 566	0	621	0.00992 14	0	638	0.0099 6066	0	645	0.00999 845	0	646	0.01	0
573	0.0098 0526	0	621	0.00992 14	0	638	0.0099 6066	0	645	0.00999 845	0	646	0.01	0