DENIS RAMSKIY II-11

Code of Program

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Laba_2
{
    class Gen
    {
        private List<int> hromosom;
        private double fitness;

        public Gen(Random rnd)
        {
            hromosom = new List<int>();
            for (int i = 0; i < 22; i++)
                hromosom.Add(rnd.Next()%2);

            List<double> temp = new List<double>(parse(hromosom));
            fitness = temp[0] * Math.Sin(Math.Abs(temp[0])) + temp[1] *
Math.Sin(Math.Abs(temp[1]));
        }

        public Gen(Gen obj)
        {
            this.hromosom = new List<int>();
            for (int i = 0; i < 22; i++)
                this.hromosom.Add(obj.get()[i]);

            this.fitness = obj.get_fitness();
        }

        public Gen(List<int> obj)
        {
            this.hromosom = new List<int>();
            for (int i = 0; i < 22; i++)
                this.hromosom.Add(obj[i]);

            List<double> temp = new List<double>(parse(hromosom));
            fitness = temp[0] * Math.Sin(Math.Abs(temp[0])) + temp[1] *
Math.Sin(Math.Abs(temp[1]));
        }

        public void set_fitness(double a)
        {
            this.fitness = a;
        }

        public List<int> get()
        {
            return hromosom;
        }

        public double get_fitness()
        {
            return fitness;
        }

        public List<double> parse(List<int> gn)
```

```
        {
            List<double> result = new List<double>();

            string str = "";
            for (int i = 1; i < 11; i++)
                str += gn[i].ToString();
            if (gn[0] == 0)
                result.Add(((double)(Convert.ToInt32(str, 2) * -1)/1023)*5);
            else
                result.Add(((double)(Convert.ToInt32(str, 2) ) / 1023) * 5);

            str = "";
            for (int i = 12; i < 22; i++)
                str += gn[i].ToString();
            if (gn[0] == 0)
                result.Add(((double)(Convert.ToInt32(str, 2) * -1)/1023)*5);
            else
                result.Add(((double)(Convert.ToInt32(str, 2) ) / 1023) * 5);

            return result;
        }

        static public Gen operator +(Gen obj1, Gen obj2)
        {
            obj1.get().Clear();
            for (int i = 0; i < 22; i++)
                obj1.get().Add(obj2.get()[i]);
            obj1.set_fitness(obj2.get_fitness());

            return obj1;
        }

    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Laba_2
{
    class Generation
    {
        private List<Gen> generation;
        private double fitness;

        public Generation(Random rnd)
        {
            generation = new List<Gen>();
            for (int I = 0; I < 20; I++)
            {
                Gen temp = new Gen(rnd);
                generation.Add(temp);
                fitness += temp.get_fitness();
            }
            this.sort();
        }

        public void sort()
        {
            for(int i=0;i<20;i++)
                for(int j=0;j<19;j++)
```

```
                    if (generation[j].get_fitness() < generation[j +
1].get_fitness())
                    {
                        Gen temp = new Gen(generation[j]);
                        generation[j] += generation[j + 1];
                        generation[j + 1] += temp;
                    }
        }

        public List<Gen> get_parents(Random rnd)
        {
            List<Gen> TEMP = new List<Gen>();
            TEMP.Add(generation[rnd.Next() % 10]);
            TEMP.Add(generation[rnd.Next() % 10]);
            return TEMP;
        }

        public Gen get_child(Random rnd,List<Gen> par)
        {
            List<int> TEMP=new List<int>();
            for (int i = 0; i < 22; i++)
                if (rnd.Next() % 2 == 0)
                    TEMP.Add(par[0].get()[i]);
                else
                    TEMP.Add(par[1].get()[i]);

            Gen child = new Gen(TEMP);
            return child;
        }

        public void new_generation(Random rnd)
        {
            List<Gen> ngeneration= new List<Gen>();
            double nfitness=0;
            for (int i = 0; i < 20; i++)
            {
                ngeneration.Add(this.get_child(rnd, this.get_parents(rnd)));
                nfitness += ngeneration[i].get_fitness();
            }

            for (int i = 0; i < 20; i++)
                this.generation[i] += ngeneration[i];
            this.fitness = nfitness;
            this.sort();
        }

        public double get_fitness()
        {
            return fitness;
        }

        public List<Gen> get()
        {
            return generation;
        }
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;
```

```
namespace Laba_2
{
    class Program
    {
        static void Main(string[] args)
        {
            Random rnd = new Random(DateTime.Now.Millisecond);
            Generation work = new Generation(rnd);
            double chek = work.get_fitness();
            while (true)
            {
                work.new_generation(rnd);
                for (int i = 0; i < 20; i++)
                {
                    File.AppendAllText("y.txt", 
work.get()[i].get_fitness().ToString() + "\r\n");
                    File.AppendAllText("x1.txt", parse(work.get()[i].get())[0] + 
"\r\n");
                    File.AppendAllText("x2.txt", parse(work.get()[i].get())[1] + 
"\r\n");
                }
            }
        }
        static public List<double> parse(List<int> gn)
        {
            List<double> result = new List<double>();

            string str = "";
            for (int i = 1; i < 11; i++)
                str += gn[i].ToString();
            if (gn[0] == 0)
                result.Add(((double)(Convert.ToInt32(str, 2) * -1) / 1023) * 5);
            else
                result.Add(((double)(Convert.ToInt32(str, 2)) / 1023) * 5);

            str = "";
            for (int i = 12; i < 22; i++)
                str += gn[i].ToString();
            if (gn[0] == 0)
                result.Add(((double)(Convert.ToInt32(str, 2) * -1) / 1023) * 5);
            else
                result.Add(((double)(Convert.ToInt32(str, 2)) / 1023) * 5);

            return result;
        }
    }
}
```

My results:

Generation 2

X1	X2	Y
-4,11535	-3,88563	6,035
-0,21994	-4,69697	4,648426
-4,74585	-2,94233	4,160756
2,1261	1,691105	3,485514
-1,44184	-4,61388	3,161647

1,813294	2,932551	2,368809
0,488759	2,26784	1,968338
0,889541	2,683284	1,878149
0,180841	2,26784	1,771377
-1,4565	-4,05181	1,752457
0,268817	1,676442	1,738491
0,244379	1,036168	0,950708
-1,78886	-3,40665	-0,85408
1,783969	4,55523	-2,7555
-2,08211	-2,78104	-2,79695
-2,23363	-2,32649	-3,45387
4,43304	0,391007	-4,11217
0,180841	4,398827	-4,15182
0,180841	4,55523	-4,46657
4,706745	0,459433	-4,50294

Generation 4

X1	X2	Y
-4,85337	-4,95112	9,615918
-4,85337	-4,99022	9,604075
-4,85337	-4,99022	9,604075
-4,85337	-4,99022	9,604075
-4,85337	-4,99022	9,604075
-4,69697	-4,95112	9,507111
-4,89247	-4,36461	8,916674
-4,85337	-4,36461	8,908538
-4,85337	-4,36461	8,908538
-4,85337	-4,36461	8,908538
-4,69697	-4,36461	8,79973
-4,73607	-4,33529	8,765417
-4,69697	-4,33529	8,727086
-4,22776	-5	8,53555
-4,22776	-5	8,53555
-4,26686	-4,64809	8,488837
-4,26686	-4,63832	8,475948
-4,85337	-4,06158	8,036599
-4,26686	-4,33529	7,881022
-4,22776	-4,36461	7,844247

Generation 6

X1	X2	Y
-4,91202	-4,57967	9,353858
-4,91202	-4,57967	9,353858
-4,84848	-1,29521	3,557316
1,759531	2,209189	3,502385
-4,7263	-1,29521	3,479502

2,385142	2,116325	3,446176
-4,9218	-1,45161	3,372961
-4,85337	-1,48583	3,324753
1,608016	1,608016	3,213804
1,608016	1,603128	3,209192
2,033236	1,251222	3,00755
-4,07136	-1,60802	1,656211
4,071359	1,524927	-1,73979
4,286413	1,29521	-2,65702
2,40958	4,613881	-2,98103
4,872923	2,077224	-2,99377
4,540567	1,251222	-3,28584
4,110459	0,278592	-3,31141
4,604106	1,251222	-3,38927
4,848485	1,29521	-3,55732

Generation 11

X1	X2	Y
-4,90225	-4,94624	9,625769
-4,90225	-4,94624	9,625769
-4,90225	-4,94624	9,625769
-4,90225	-4,94624	9,625769
-4,90225	-4,94624	9,625769
-4,90225	-4,94624	9,625769
-4,90225	-4,94624	9,625769
-4,90225	-4,94624	9,625769
-4,90225	-4,94624	9,625769
-4,90225	-4,94624	9,625769
-4,90225	-4,94624	9,625769
-4,90225	-4,94624	9,625769
-4,90225	-4,94624	9,625769
-4,90225	-4,94624	9,625769
-4,90225	-4,94624	9,625769
-4,90225	-4,94624	9,625769
-4,90225	-4,94624	9,625769
-4,90225	-4,94624	9,625769
-4,90225	-4,94624	9,625769
-4,90225	-4,94624	9,625769