DENIS RAMSKIY
GROUP II-11

the changed class

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

namespace GenAlg
{
    class Generation
    {
        List<Gen> parens_generation;
        int general_fitness;

        public Generation()
        {
            parens_generation = new List<Gen>();
            general_fitness = 0;
            for (int i = 0; i < 100; i++)
            {
                Gen temp = new Gen();
                parens_generation.Add(temp);
                general_fitness += temp.get_fitness();
            }
            sort();
        }
        public void sort()
        {
            for (int i = 0; i < parens_generation.Count; i++)
                for (int j = 0; j < parens_generation.Count - 1; j++)
                    if (parens_generation[j].get_fitness() < parens_generation[j +
1].get_fitness())
                    {
                        Gen temp = new Gen(parens_generation[j]);
                        parens_generation[j] += parens_generation[j + 1];
                        parens_generation[j + 1] += temp;
                    }
        }
        public Gen get_gen(int i)
        {
            return parens_generation[i];
        }
        public int get_general_fitness()
        {
            return general_fitness;
        }

        public List<Gen> get_parents_tranc()
        {
            List<Gen> parents = new List<Gen>();
            int rnd_tik = 17;
            while (parents.Count < 2)
            {
                for (int i = 0; i < 100; i++)
                {
                    Random rnd = new Random(DateTime.Now.Millisecond + rnd_tik *
rnd_tik);
```

```
                    if (rnd.NextDouble() < (Double)parens_generation[i].get_fitness() /
general_fitness)
                    {
                        parents.Add(parens_generation[i]);
                        parens_generation[i].inkr_used();
                        break;
                    }
                    rnd_tik += 7*i;
                }
            }
            return parents;
        }
        public List<Gen> get_parents_rul()
        {
            List<Gen> parents = new List<Gen>();
            int rnd_tik = 17;
            while (parents.Count < 2)
            {
                Random rnd = new Random(DateTime.Now.Millisecond + rnd_tik * rnd_tik);
                int a=rnd.Next() % 50;
                parents.Add(parens_generation[a]);
                parens_generation[a].inkr_used();
                rnd = new Random(DateTime.Now.Millisecond - rnd_tik * rnd_tik);
                a = rnd.Next() % 50;
                parents.Add(parens_generation[a]);
                parens_generation[a].inkr_used();
                rnd_tik += 19;
            }
            return parents;
        }
        public Gen get_child(List<Gen> perants)
        {
            int rnd_tik = 17;
            List<int> cild = new List<int>();
            for (int i = 0; i < 1000; i++)
            {
                Random rnd = new Random(DateTime.Now.Millisecond - rnd_tik * rnd_tik);
                if (rnd.Next(0, 2) == 1)
                    cild.Add(perants[0].get_gen(i));
                else
                    cild.Add(perants[1].get_gen(i));
                rnd_tik *= i;
            }
            Gen child = new Gen(cild);
            return child;
        }
        public Gen get_child(List<Gen> perants,int k)
        {
            Random rnd = new Random(DateTime.Now.Millisecond+(k*k*k*k)-k);
            List<int> cild = new List<int>();
            int edge = rnd.Next(0, 1000);
            for (int i = 0; i < 1000; i++)
            {
                if (i <= edge )
                    cild.Add(perants[0].get_gen(i));
                else
                    cild.Add(perants[1].get_gen(i));
            }
            Gen child = new Gen(cild);
            return child;
        }
        public void new_generation()
        {
            List<Gen> new_generation = new List<Gen>();
            int new_fitnes = 0;
```

```
            for (int i = 0; i < 100; i++)
            {
                Gen temp = new Gen(get_child(get_parents_rul()));
                new_generation.Add(temp);
                new_fitnes += temp.get_fitness();
            }
            this.general_fitness = new_fitnes;
            parens_generation = new_generation;
            sort();
        }
    }
}
```

uniform crossover for roulette selection method

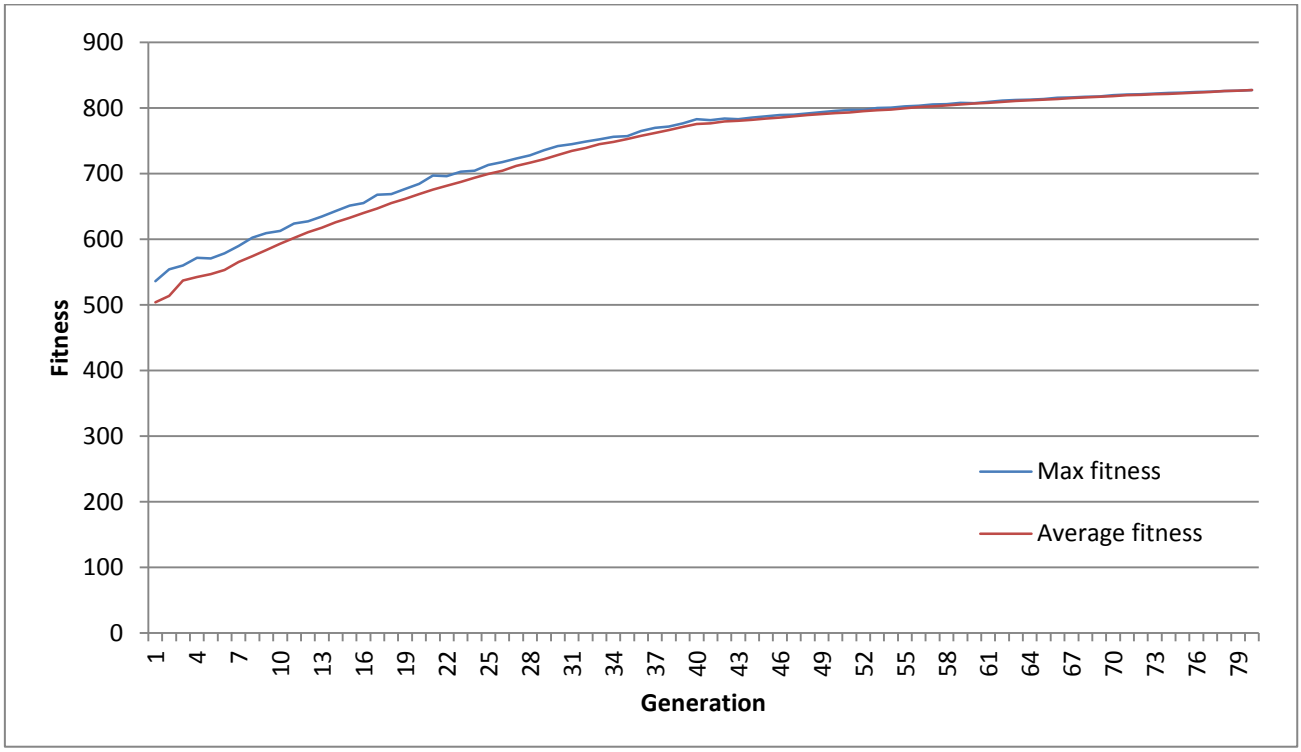

table of values from the graph

Max fitness	Average fitness	General fitness
535	502,50	50250
553	513,04	51304
560	536,83	53683
571	541,52	54152
571	546,85	54685
580	553,96	55396
590	565,61	56561
603	574,63	57463
610	584,00	58400
612	593,09	59309
624	602,72	60272
626	609,21	60921
634	616,81	61681
643	625,76	62576

651	633,31	63331
657	641,43	64143
667	646,02	64602
667	653,79	65379
677	661,38	66138
685	669,08	66908
697	675,81	67581
696	681,69	68169
705	689,29	68929
705	694,30	69430
712	698,61	69861
720	706,59	70659
721	710,49	71049
729	717,86	71786
736	722,32	72232
741	727,31	72731
745	734,30	73430
748	738,65	73865
753	745,45	74545
758	750,82	75082
758	753,41	75341
765	758,19	75819
771	762,45	76245
770	765,40	76540
776	770,58	77058
782	775,27	77527
780	775,84	77584
784	779,71	77971
783	780,42	78042
785	781,82	78182
787	783,81	78381
789	785,58	78558
790	787,60	78760
792	789,29	78929
794	790,84	79084
795	792,35	79235
797	793,41	79341
797	795,04	79504
800	796,60	79660
801	797,79	79779
803	799,71	79971
803	801,44	80144
805	802,76	80276
806	804,17	80417
808	805,59	80559
808	806,73	80673
809	808,00	80800

811	809,38	80938
812	810,72	81072
813	811,81	81181
814	812,77	81277
816	813,92	81392
816	815,18	81518
817	816,09	81609
818	817,04	81704
820	818,18	81818
820	819,35	81935
821	820,10	82010
822	820,91	82091
823	821,71	82171
824	822,58	82258
824	823,49	82349
825	824,57	82457
826	825,85	82585
827	826,61	82661
827	827,38	82738

1-point crossover for roulette selection method

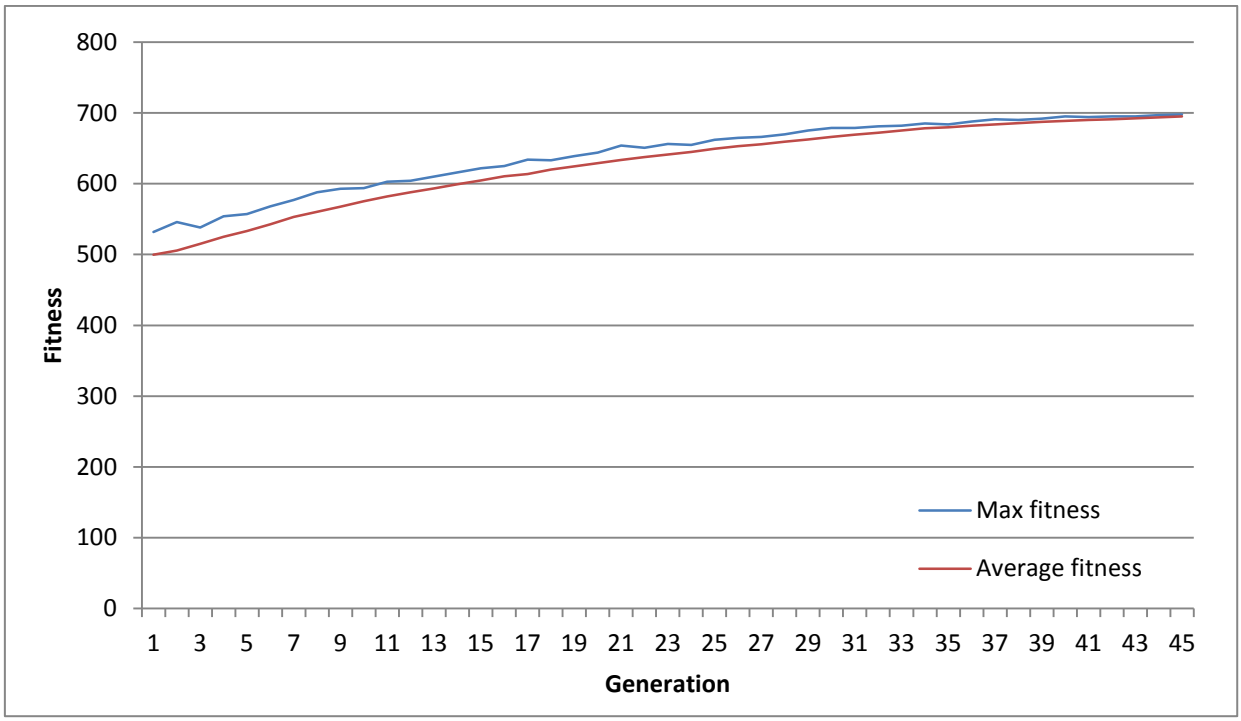

table of values from the graph

Max fitness	Average fitness	General fitness
532	499,87	49987
546	505,79	50579
538	515,14	51514

554	525,27	52527
557	533,36	53336
568	542,76	54276
577	552,91	55291
588	560,41	56041
593	567,41	56741
594	575,23	57523
603	581,97	58197
604	587,73	58773
610	593,41	59341
616	599,43	59943
622	604,60	60460
625	610,36	61036
634	613,86	61386
633	620,15	62015
639	624,52	62452
644	629,26	62926
654	633,78	63378
651	637,70	63770
656	641,37	64137
655	645,04	64504
662	649,36	64936
665	652,91	65291
666	655,94	65594
670	659,60	65960
675	662,72	66272
679	666,18	66618
679	669,55	66955
681	672,14	67214
682	675,17	67517
685	678,17	67817
684	679,76	67976
688	681,78	68178
691	683,73	68373
690	685,44	68544
692	687,21	68721
695	688,63	68863
694	689,92	68992
695	691,20	69120
695	692,57	69257
697	693,85	69385
698	694,96	69496