**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра САПР**

**отчет**

**по лабораторной работе №2**

**по дисциплине «Оптимальное проектирование»**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 3301 |  | Малинина Е.А. |
| Преподаватель |  | Каримов А.И. |

Санкт-Петербург

2018

**Лабораторная работа №2**   
Изучение различных кодировок генотипа

Цель работы

Исследование двух основных способов кодирования генотипа хромосом в генетическом алгоритме и их эффективности.

Задание

Популяция 100 особей, турнирный отбор (размер 5). Для целочисленной кодировки:   
1-точечное скрещивание (*pc* = 0,75), инверсия (*pI* = 0,07). Для вещественной кодировки: 1-точечное скрещивание (*pc* = 0,75), мутация (*pm* = 0,1). Условие остановки - 50 итераций;

Для всех вариантов: Длина гена *m* = 12 (при целочисленном кодировании).

Таблица тестовых функций

|  |  |  |
| --- | --- | --- |
| № | Функция *y*(***x***) | Поисковый интервал |
| (12) | 4*x*12 + 3*x*22 – 4*x*1*x*22 + *x*1 | *x*1∈ (-5; 10); *x*2 ∈ (0; 10); |
| (17) | 100(*x*2 – *x*12)2 + (1 – *x*1)2 + 90(*x*4 – *x*32)2 + (1 – *x*3)3 +  10.1[(*x*2 – 1)2 + (*x*4 – 1)2] + 19.8(*x*2 – 1)(*x*4 – 1)  (функция имеет несколько локальных минимумов) | *x*1∈ (–5,12; 5,12); *x*2 ∈ (–5,12; 5,12);  *x*3∈ (–5,12; 5,12); *x*4 ∈ (–5,12; 5,12); |

Минимум функции 12 в точке ( 0 ; 0 )

Минимум функции 17 в точке ( 1 ; 1 ; 1 ; 1)

Кодирование

Выбор способа кодирования является одним из важнейших этапов при использовании генетического алгоритма. В хромосоме кодируются параметры решения. Для этого возможно использование целочисленного и вещественного кодирования.

Целочисленное кодирование

В классическом ГА хромосома представляет собой битовую строку, в которой закодированы параметры решения поставленной задачи. Как правило, считают, что каждому параметру соответствует свой ген. Несмотря на то, что каждый параметр закодирован в хромосоме целым числом в двоичном представлении, его можно раскодировать в вещественное число, принадлежащее пространству поиска.

Для применения ГА в задачах оптимизации необходимо определить диапазоны, в пределах которых лежат значения параметров задачи. Затем каждый диапазон разбивается на (2*m*) отрезков, где *m* – разрядность гена, и каждому отрезку соответствует определенный целочисленный код – значение гена. При этом для раскодирования значений из генотипа в фенотип и обратно применяют следующие формулы:

, (1)

,

где *r* – вещественное (декодированное) значение параметра, *g* – целочисленное (закодированное) значение параметра, *x*max и *x*min – соответственно максимальное и минимальное значения из диапазона декодированного параметра.

Точность закодированной сетки в таком случае определяется выражением

 (2)

Вещественное кодирование

Часто бывает удобно кодировать в гене не целое число, а вещественное. Данный тип кодирования позволяет избавиться от операций кодирования/декодирования, используемых в целочисленном кодировании, а также увеличить точность найденного решения. Также необходимо заметить, что для обеих схем кодирования применимы совершенно разные генетические операторы.

Алгоритм генетический

Блок-схема классического ГА изображена на рис. 1.

НЕТ

Инициализация – выбор исходной популяции хромосом

Оценивание приспособленности популяции

Критерий остановки

Селекция хромосом

Применение генетических операторов

Создание новой популяции

Выбор «лучшей» хромосомы

ДА

Рис. 1

Классический ГА состоит из следующих шагов:

1. инициализация, или выбор исходной популяции хромосом;
2. оценка приспособленности хромосом в популяции;
3. проверка условия остановки алгоритма;
4. селекция хромосом;
5. применение генетических операторов;
6. формирование новой популяции;
7. выбор «наилучшей» хромосомы.

Спецификация

|  |  |
| --- | --- |
| Класс | Назначение |
| class Chromosom | Работает с целочисленной хромосомой |
| class Individ | Работает с особью содержащий целочисленную хромосому |
| class Population | Работает с несколькими особями |
| class VeshChromosom | Работает с вещественной хромосомой |
| class VeshIndivid | Работает с особью содержащий вещественною хромосому |
| class VeshPopulation | Работает с несколькими особями |
| class Parametrs | Содержит все переменные-параметры алгоритма |

График функции

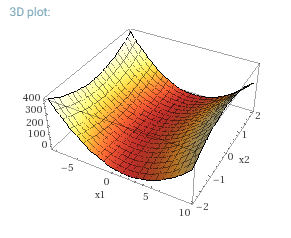
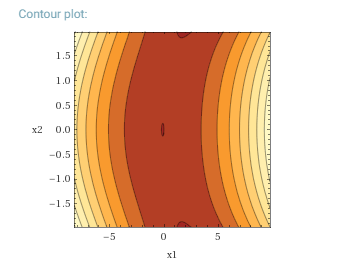
 

Рис.2. График 1 исследуемой функции

Рис.3. График 2 исследуемой функции

Результат тестирования

|  |  |
| --- | --- |
| Функция 12 минимум точка ( 0 ; 0 ) | Функция 17 минимум точка ( 1 ; 1 ; 1 ; 1 ) |
| Ген 12 Мутация 0,00 Инверсия 0,07  Интервал (-5;10) (-5;10)  Целочисленная кодировка  ------- + 0 + -------  Min -0,029375933856069  Min 1 -0,029375933856069  Min 2 -0,029375933856069  Min 3 -0,029375933856069  Min 4 -0,029375933856069  Min 5 -0,029375933856069  Min 6 -0,029375933856069  Min 7 -0,029375933856069  Min 8 -0,029375933856069  Min 9 -0,029375933856069  Min 10 -0,029375933856069  Min 11 -0,029375933856069  Min 12 -0,029375933856069  Min 13 -0,029375933856069  Min 14 -0,029375933856069  Min 15 -0,029375933856069  Min 16 -0,029375933856069  Min 17 -0,029375933856069  Min 18 -0,029375933856069  Min 19 -0,029375933856069  Min 20 -0,029375933856069  Min 21 -0,029375933856069  Min 22 -0,029375933856069  Min 23 -0,029375933856069  Min 24 -0,029375933856069  Min 25 -0,029375933856069  Min 26 -0,029375933856069  Min 27 -0,029375933856069  Min 28 -0,029375933856069  Min 29 -0,029375933856069  Min 30 -0,029375933856069  Min 31 -0,029375933856069  Min 32 -0,029375933856069  Min 33 -0,029375933856069  Min 34 -0,029375933856069  Min 35 -0,029375933856069  Min 36 -0,029375933856069  Min 37 -0,029375933856069  Min 38 -0,029375933856069  Min 39 -0,029375933856069  Min 40 -0,029375933856069  Min 41 -0,029375933856069  Min 42 -0,029375933856069  Min 43 -0,029375933856069  Min 44 -0,029375933856069  Min 45 -0,029375933856069  Min 46 -0,029375933856069  Min 47 -0,029375933856069  Min 48 -0,029375933856069  Min 49 -0,029375933856069  Min 50 -0,029375933856069  -------END-------  1) -0,212454212454213  2) -0,0256410256410256 | Ген 12 Мутация 0,00 Инверсия 0,07  Интервал (-5,12;5,12) (-5,12;5,12)  (-5,12;5,12) (-5,12;5,12)  Целочисленная кодировка------- + 0 + -------  Min 111,491403281254  Min 1 92,935359341385  Min 2 50,0610589250657  Min 3 7,31785968738227  Min 4 7,31785968738227  Min 5 11,6965770883161  Min 6 4,01205267946573  Min 7 4,01205267946573  Min 8 4,01205267946573  Min 9 4,01205267946573  Min 10 4,01205267946573  Min 11 4,01205267946573  Min 12 4,01205267946573  Min 13 7,31785968738227  Min 14 4,01205267946573  Min 15 4,01205267946573  Min 16 4,01205267946573  Min 17 4,01205267946573  Min 18 4,01205267946573  Min 19 4,01205267946573  Min 20 4,01205267946573  Min 21 4,01205267946573  Min 22 4,01205267946573  Min 23 4,01205267946573  Min 24 4,01205267946573  Min 25 4,01205267946573  Min 26 4,01205267946573  Min 27 7,31785968738227  Min 28 4,01205267946573  Min 29 4,01205267946573  Min 30 4,01205267946573  Min 31 7,31785968738227  Min 32 4,01205267946573  Min 33 4,01205267946573  Min 34 4,01205267946573  Min 35 4,01205267946573  Min 36 4,01205267946573  Min 37 4,01205267946573  Min 38 4,01205267946573  Min 39 4,01205267946573  Min 40 7,31785968738227  Min 41 4,01205267946573  Min 42 7,31785968738227  Min 43 4,01205267946573  Min 44 4,01205267946573  Min 45 4,01205267946573  Min 46 4,01205267946573  Min 47 4,01205267946573  Min 48 4,01205267946573  Min 49 4,01205267946573  Min 50 4,01205267946573  -------END-------  1) 0,826451751979134  2) 0,783941374418936  3) 0,828952362423851  4) 0,806446868421394 |
| Ген 12 Мутация 0,00 Инверсия 0,70  Интервал (-5;10) (-5;10)  Вещественная кодировка  ------- + 0 + -------  Min -1,59810982603032  Min 1 -0,933007996020906  Min 2 -0,0483123481118102  Min 3 -0,0245815012079504  Min 4 -0,0170844015686375  Min 5 -0,00636484750460917  Min 6 0,000123620445647848  Min 7 -0,00585762174184686  Min 8 0,0123769449307532  Min 9 0,0126786125158233  Min 10 0,0129795167486821  Min 11 0,0130827184608228  Min 12 0,0129595200157274  Min 13 0,0129928898491624  Min 14 0,0129984958070028  Min 15 0,012998658804328  Min 16 0,0129995725773658  Min 17 0,013001711069673  Min 18 0,013001711069673  Min 19 0,013001711069673  Min 20 0,013001711069673  Min 21 0,013001711069673  Min 22 0,0130017431334153  Min 23 0,0130017478719973  Min 24 0,0130017536206343  Min 25 0,0130017544457647  Min 26 0,0130017545834698  Min 27 0,0130017546304016  Min 28 0,0130017546493496  Min 29 0,0130017546645543  Min 30 0,0130017546682543  Min 31 0,0130017546741418  Min 32 0,0130017546741418  Min 33 0,0130017546748701  Min 34 0,0130017546752086  Min 35 0,013001754675324  Min 36 0,0130017546753634  Min 37 0,0130017546753752  Min 38 0,0130017546753923  Min 39 0,0130017546753923  Min 40 0,0130017546753948  Min 41 0,0130017546753952  Min 42 0,0130017546753952  Min 43 0,0130017546753953  Min 44 0,0130017546753954  Min 45 0,0130017546753954  Min 46 0,0130017546753954  Min 47 0,0130017546753954  Min 48 0,0130017546753954  Min 49 0,0130017546753954  Min 50 0,0130017546753954  -------END-------  1) 0,00927042846083074  2) -0,0338130107751237 | Ген 12 Мутация 0,10 Инверсия 0,00  Интервал (-5,12;5,12) (-5,12;5,12)  (-5,12;5,12) (-5,12;5,12)  Вещественная кодировка  ------- + 0 + -------  Min 354,905227292072  Min 1 4,80200006531175  Min 2 13,6754811897656  Min 3 12,5331124405167  Min 4 3,35978799248648  Min 5 1,65626736802464  Min 6 1,92690581310965  Min 7 1,76196739839691  Min 8 1,53161038021058  Min 9 1,47550414009696  Min 10 1,50918917697214  Min 11 1,57889227710372  Min 12 1,54146216947633  Min 13 1,57814234347513  Min 14 1,57814234347513  Min 15 1,64265454956846  Min 16 1,67249202332772  Min 17 1,72133696833486  Min 18 1,7782796469755  Min 19 1,81102967174489  Min 20 1,83456873078794  Min 21 1,87125355447712  Min 22 1,88100826082865  Min 23 1,88100826082865  Min 24 1,98193524831658  Min 25 2,02428764668645  Min 26 2,07529364653661  Min 27 2,11732799683565  Min 28 2,13730652020851  Min 29 2,1420755360058  Min 30 2,14623595191303  Min 31 2,14840098830324  Min 32 2,14894474806795  Min 33 2,14894474806795  Min 34 2,15003851158404  Min 35 2,1581577747247  Min 36 2,19231437624875  Min 37 2,29270127967049  Min 38 2,3681703477387  Min 39 2,37927716148834  Min 40 2,40905930565541  Min 41 2,42211391949294  Min 42 2,47106538736812  Min 43 2,47106538736812  Min 44 2,54198386086432  Min 45 2,54489945772952  Min 46 2,54489945772952  Min 47 2,56750997391433  Min 48 2,57325183960416  Min 49 2,57733822175011  Min 50 2,57605771946404  -------END-------  1) 1,3377560382122  2) 1,81204709240983  3) -0,242535917813003  4) -0,708464429154939 |

Анализ

Программа в дух мерном пространстве работает хорошо, как в целочисленной кодировки, так и в вещественной. При расчёте функции от 4 переменных алгоритм не находит минимум за 50 итераций. Так же сильно влияет мутация, вероятность которой 0,1. И если проанализировать все лучшие особи этих популяции, то можно заметить, что многие из них лучше итоговой особи. Тогда для сохранения таких особей можно

1. модифицировать алгоритм и оставлять 1 наилучшую особь в следящей популяции и опять проводить с ней скрещивание
2. создать «вольер» для лучшей особи. И если в популяции появиться особь лучше вольерной, то посадить ее в вольер. Тогда решением будет та особь, которая будет в вольере.
3. Уменьшить поисковый интервал

Тест на функцию 17 минимум точка ( 1 ; 1 ; 1 ; 1 )

|  |  |
| --- | --- |
| Ген 12 Мутация 0,00 Инверсия 0,07  Интервал (-2;2) (-2;2) (-2;2) (-2;2)  Целочисленная кодировка  ------- + 0 + -------  Min 26,1604982918667  Min 1 9,95164239936615  Min 2 10,1541441941737  Min 3 4,88170999177791  Min 4 5,88696137489499  Min 5 4,88170999177791  Min 6 5,34771251437175  Min 7 5,34771251437175  Min 8 3,96266345059707  Min 9 4,04765763738412  Min 10 5,34771251437175  Min 11 2,51591533636975  Min 12 2,51591533636975  Min 13 5,35353393147145  Min 14 3,83850385895827  Min 15 2,51591533636975  Min 16 2,51591533636975  Min 17 2,51591533636975  Min 18 2,51591533636975  Min 19 2,51591533636975  Min 20 2,51591533636975  Min 21 2,51591533636975  Min 22 2,51591533636975  Min 23 2,51591533636975  Min 24 2,51591533636975  Min 25 2,51591533636975  Min 26 2,51591533636975  Min 27 2,51591533636975  Min 28 2,51591533636975  Min 29 2,51591533636975  Min 30 2,51591533636975  Min 31 2,51591533636975  Min 32 2,51591533636975  Min 33 2,51591533636975  Min 34 2,51591533636975  Min 35 2,51591533636975  Min 36 2,51591533636975  Min 37 2,51591533636975  Min 38 2,51591533636975  Min 39 2,51591533636975  Min 40 2,51591533636975  Min 41 2,51591533636975  Min 42 2,51591533636975  Min 43 2,51591533636975  Min 44 2,51591533636975  Min 45 2,51591533636975  Min 46 2,51591533636975  Min 47 2,51591533636975  Min 48 2,51591533636975  Min 49 2,51591533636975  Min 50 2,51591533636975  -------END-------  1) -0,330647130647131  2) 0,158730158730159  3) -1,18827838827839  4) 1,44713064713065 | Ген 12 Мутация 0,10 Инверсия 0,00  Интервал (-2;2) (-2;2) (-2;2) (-2;2)  Вещественная кодировка  ------- + 0 + -------  Min 20,3015025700197  Min 1 4,79718636550281  Min 2 11,3107892920147  Min 3 5,12155029048401  Min 4 4,31470032277592  Min 5 4,38529950395385  Min 6 5,11457220298425  Min 7 5,35224985279087  Min 8 5,19999861439855  Min 9 5,17015183844337  Min 10 5,20693209261628  Min 11 5,11130943959496  Min 12 5,13639911291056  Min 13 5,13735109021761  Min 14 5,10556833294359  Min 15 5,11331365741651  Min 16 5,08660907439405  Min 17 5,09579851122699  Min 18 5,21481370761289  Min 19 5,2172869298328  Min 20 5,46763192902507  Min 21 5,50056846921721  Min 22 5,55935939509322  Min 23 5,70117207549415  Min 24 5,73664796637932  Min 25 5,75296146434009  Min 26 5,76174449714984  Min 27 5,84019140584178  Min 28 5,88318541032445  Min 29 5,94483402465193  Min 30 5,95488061877418  Min 31 5,95996738275899  Min 32 5,96252663217168  Min 33 5,96501511231543  Min 34 5,97168954407448  Min 35 5,9895555949439  Min 36 5,98971325266137  Min 37 6,00247111020698  Min 38 6,00246492510588  Min 39 6,00791594972661  Min 40 6,01065575739668  Min 41 6,0179815306346  Min 42 6,01820906443541  Min 43 6,01832537483092  Min 44 6,02389661114489  Min 45 6,05956115740995  Min 46 6,05956115740995  Min 47 6,30999114484001  Min 48 6,43999814581904  Min 49 6,85652971045588  Min 50 6,84975868016511  -------END-------  1) -0,678733661754439  2) 0,418180603620175  3) 0,971894662196614  4) 0,838870420754584 |

Преимущества и недостатки двоичного кодирования

Двоичное представление хромосом влечет за собой определенные трудности при поиске в непрерывных пространствах большой размерности (большое количество генов *G*), и когда требуется высокая точность найденного решения (большое количество разрядов в каждом гене *m*). Формулы (1, 2) указывают на то, что *p* сильно зависит от *m*, т.е. точность представления определяется количеством разрядов, используемых для кодирования одного гена. Поэтому при увеличении *m* суммарная длина хромосом *L* увеличивается и становится огромной.

Например, пусть для 100 переменных, изменяющихся в интервале [-500; 500], требуется найти экстремум с точностью до шестого знака после запятой. В этом случае при использовании ГА с двоичным кодированием длина строки составит порядка 3000 элементов.

Для эффективного хранения в памяти и обработки такого количества разрядов необходимы специально разработанные программные ухищрения, не говоря уже о специфике работы ГА с многоразрядными хромосомами. На первых итерациях алгоритм потратит много усилий на оценку младших разрядов хромосомы. Но эффективность поиска на первых итерациях зависит как раз таки от старших разрядов. Следовательно, пока в процессе эволюции алгоритм не выйдет на значение старшего разряда в окрестности оптимума, операции с младшими разрядами окажутся малоэффективными. А когда это произойдет, отпадет надобность в операциях со старшими разрядами – точность решения необходимо улучшать операциями в младших разрядах. Этот механизм хорошо работает только в случае, когда длина хромосомы относительно невелика. ГА оперирует битовой строкой целиком, и в длинной хромосоме младшие разряды генов фиксируются на первых же итерациях ГА, принимая случайное значение. Вообще, в настоящее время разработаны специальные ухищрения по выходу из этой ситуации, однако их рассмотрение выходит за рамки данного курса.

Итак, основные достоинства двоичной схемы:

1. Хорошие результаты на пространствах малой размерности.
2. Устоявшиеся реализации генетических операторов, без «изобретения колеса», доказанная теоретическая база.
3. Основные недостатки:
4. Требуются специальные программные ухищрения для организации эффективного хранения и обхода структур с большими хромосомами.
5. В пространствах с большой размерностью точность алгоритма резко падает.

Вывод

В ходе проделанной работы ознакомились с вещественным кодированием в генетическом алгоритме. Сравнили работу алгоритма при целочисленном и вещественном кодирование. Модифицировали программу для поиска минимума для функций с 4 неизвестными.

Код программы

|  |  |
| --- | --- |
| class class Chromosom  {  private static int length = 15;  private bool[] chrom;  //возможность измененния длины гена  public static int Length  { get { return length; }  set { length = value; } }  //возможность обращаться к к конкретному значению в хромосоме (переопределение)  public bool this[int i]  { get { return chrom[i]; }  set { chrom[i] = value; } }  public Chromosom(Parametrs A)  { Chromosom.Length = A.Length;  // int L = BinaryChromosome.Length;  chrom = new bool[A.Length];  Randomize(A);  }  public string Prin(Parametrs A)  {  string tectik=" ";  char a;  for (int i = 0; i < Chromosom.Length; i++)  {  if (i == A.point1 || i == A.point2 && i!=0)  tectik = tectik + '\_';  a = (chrom[i] ? '1' : '0');  tectik = tectik + a ; }  return tectik;  }  //  public void Randomize(Parametrs A)  {  for (int i = 0; i < Chromosom.Length; i++)  {  if (A.rnd1.NextDouble() < 0.5)  chrom[i]= !chrom[i];  }  }  // мутация работает  public void Mutate(Parametrs A)  { for (int i = 0; i < Chromosom.Length; i++)  { if (A.rnd1.NextDouble() < A.Muttion)  chrom[i] = !chrom[i]; } }  // инвенсия работает  public void Invers(Parametrs A)  { if (A.rnd1.NextDouble() < A.Invtion)  { for (int i = 0; i < Chromosom.Length; i++)  { chrom[i] = !chrom[i]; }  } }  // кроссенговер  public static void PointCrossOver(Parametrs A, Chromosom chromA, Chromosom chromB, out Chromosom chromC, out Chromosom chromD)  if (!A.Ipoint )  {// однототечный кроссенговер  chromC = new Chromosom(A);  chromD = new Chromosom(A);  for (int i = 0; i < A.point1; i++)  { chromC[i] = chromA[i];  chromD[i] = chromB[i]; }  for (int i = A.point1; i < Chromosom.Length; i++)  { chromC[i] = chromB[i];  chromD[i] = chromA[i]; } }  else  {// двухточеный кроссенговер  chromC = new Chromosom(A);  chromD = new Chromosom(A);  int i = 0;  for (i = 0; i < A.point1; i++)  {  chromC[i] = chromA[i];  chromD[i] = chromB[i];  }  for (i = A.point1; i < A.point2; i++)  {  chromC[i] = chromB[i];  chromD[i] = chromA[i];  }  for (i = A.point2; i < Chromosom.Length; i++)  {  chromC[i] = chromA[i];  chromD[i] = chromB[i];  }  }  }  public static void CopiCh(Parametrs A, Chromosom chromA, out Chromosom chromB)  { chromB = new Chromosom(A);  for (int i = 0; i < A.Length/ A.Setka; i++)  { chromB[i] = false; }  for (int i = A.Length / A.Setka; i < Chromosom.Length; i++)  { chromB[i] = chromA[i-12]; }  } } | class VeshChrom  {  public double chrom;    //возможность измененния длины гена      public VeshChrom(Parametrs A, int i)  {    chrom = A.rnd1.Next(Convert.ToInt32( A.interM[i,0] ) , Convert.ToInt32(A.interM[i, 1]));  chrom = chrom + A.rnd1.NextDouble();  }  public string Prin()  {  return chrom.ToString();  }    // мутация работает  public void Mutate(Parametrs A)  {  if (A.rnd1.NextDouble() < A.Muttion)  chrom = chrom - A.rnd1.NextDouble();  }  // инвенсия работает  public void Invers(Parametrs A)  {  // double iii = chrom \* 100 / (Math.Pow(2.0, VeshChrom.Length) - 1);  if (A.rnd1.NextDouble() < A.Invtion)  chrom = chrom + A.rnd1.Next();  }    } |
| class Individ  { // private  public Chromosom[] xrom = new Chromosom[4];  public double Fank\_out;  // Parametrs A  public Individ(Parametrs A) //double min, double max,  { if (A.Chom == 2)  { xrom[0] = new Chromosom(A);  xrom[1] = new Chromosom(A); }  if (A.Chom == 4)  {  xrom[0] = new Chromosom(A);  xrom[1] = new Chromosom(A);  xrom[2] = new Chromosom(A);  xrom[3] = new Chromosom(A);  }  public string Prin\_Ind(Parametrs A)  { if (A.Chom == 2)  {return "1)" + xrom[0].Prin(A) + Environment.NewLine + "2)" + xrom[1].Prin(A) + Environment.NewLine; }  if (A.Chom == 4)  {return "1)" + xrom[0].Prin(A) + Environment.NewLine + "2)" + xrom[1].Prin(A) + Environment.NewLine +  "3)" + xrom[2].Prin(A) + Environment.NewLine + "4)" + xrom[3].Prin(A) + Environment.NewLine;  }  else return "0\_0";  private void Mutate(Parametrs p)  {  if (p.Chom == 2)  {  xrom[0].Mutate(p);  xrom[1].Mutate(p);  }  if (p.Chom == 4)  {  xrom[0].Mutate(p);  xrom[1].Mutate(p);  xrom[2].Mutate(p);  xrom[3].Mutate(p);  }    }  private void Invers(Parametrs p)  {  if (p.Chom == 2)  {  xrom[0].Invers(p);  xrom[1].Invers(p);  }  if (p.Chom == 4)  {  xrom[0].Invers(p);  xrom[1].Invers(p);  xrom[2].Invers(p);  xrom[3].Invers(p);  }    }  public void MutateAll(Parametrs p)  {  Mutate(p);  Invers(p);  }    public double Fenotip\_1(Parametrs a)  { return Genot\_in\_Fenotip(xrom[0], a.interM[0, 0], a.interM[0, 1]); }  public double Fenotip\_2(Parametrs a)  { return Genot\_in\_Fenotip(xrom[1], a.interM[1, 0], a.interM[1, 1]); }  public double Fenotip\_3(Parametrs a)  { return Genot\_in\_Fenotip(xrom[2], a.interM[2, 0], a.interM[2, 1]); }  public double Fenotip\_4(Parametrs a)  { return Genot\_in\_Fenotip(xrom[3], a.interM[3, 0], a.interM[3, 1]); }  double Genot\_in\_Fenotip(Chromosom ch, double Min, double Max)  {  Int32 gen = 0;  for (int i = 0; i < Chromosom.Length; i++)  {if (ch[i])  gen += 1 << i;}  return Min + gen\*(Max - Min) / (Math.Pow(2.0, Chromosom.Length) - 1);  }  public string Prin\_Fen(Parametrs a)  {if (a.Chom == 2)  {return "1) " + Fenotip\_1(a) + Environment.NewLine + " 2) " + Fenotip\_2(a) + Environment.NewLine;}  if (a.Chom == 4)  {  return "1) " + Fenotip\_1(a) + Environment.NewLine + " 2) " + Fenotip\_2(a) + Environment.NewLine+  "3) " + Fenotip\_3(a) + Environment.NewLine + " 4) " + Fenotip\_4(a) + Environment.NewLine;  }  return "0\_0 ";  }  public static Individ[] PointCrossOver(Parametrs a, Individ A, Individ B)  { Individ[] C = new Individ[] { new Individ(a), new Individ(a) };  if (a.Chom == 2)  { Chromosom.PointCrossOver(a, A.xrom[0], B.xrom[0], out C[0].xrom[0], out C[1].xrom[0]);  Chromosom.PointCrossOver(a, A.xrom[1], B.xrom[1], out C[0].xrom[1], out C[1].xrom[1]);  }  if (a.Chom == 4)  Chromosom.PointCrossOver(a, A.xrom[0], B.xrom[0], out C[0].xrom[0], out C[1].xrom[0]);  Chromosom.PointCrossOver(a, A.xrom[1], B.xrom[1], out C[0].xrom[1], out C[1].xrom[1]);  Chromosom.PointCrossOver(a, A.xrom[2], B.xrom[2], out C[0].xrom[2], out C[1].xrom[2]);  Chromosom.PointCrossOver(a, A.xrom[3], B.xrom[3], out C[0].xrom[3], out C[1].xrom[3]);  }  return C;  }  public static Individ CopiInd(Parametrs a, Individ A, Individ B)  {  B = new Individ(a);  if (a.Chom == 2)  {//CopiCh(Parametrs A, Chromosom chromA, out Chromosom chromB)  Chromosom.CopiCh(a, A.xrom[0], out B.xrom[0] );  Chromosom.CopiCh(a, A.xrom[1], out B.xrom[1] );  }  if (a.Chom == 4)  {  Chromosom.CopiCh(a, A.xrom[0], out B.xrom[0]);  Chromosom.CopiCh(a, A.xrom[1], out B.xrom[1]);  Chromosom.CopiCh(a, A.xrom[2], out B.xrom[2]);  Chromosom.CopiCh(a, A.xrom[3], out B.xrom[3]);  } return B; } } | class VeshIndivid  { public VeshChrom[] xrom = new VeshChrom[4];  public double Fank\_out;  public VeshIndivid(Parametrs A)  {  if (A.Chom == 2) {  xrom[0] = new VeshChrom(A, 0);  xrom[1] = new VeshChrom(A, 1); }  if (A.Chom == 4) {  xrom[0] = new VeshChrom(A, 0);  xrom[1] = new VeshChrom(A, 1);  xrom[2] = new VeshChrom(A, 2);  xrom[3] = new VeshChrom(A, 3);  } }  public string Prin\_Ind(Parametrs A)  {  if (A.Chom == 2)  {  return "1)" + xrom[0].Prin() + Environment.NewLine + "2)" + xrom[1].Prin() + Environment.NewLine; }  if (A.Chom == 4)  { return "1)" + xrom[0].Prin() + Environment.NewLine + "2)" + xrom[1].Prin() + Environment.NewLine +  "3)" + xrom[2].Prin() + Environment.NewLine + "4)" + xrom[3].Prin() + Environment.NewLine;  }  else return "0\_0";}  private void Mutate(Parametrs p)  {if (p.Chom == 2)  { xrom[0].Mutate(p);  xrom[1].Mutate(p);  }  if (p.Chom == 4)  {  xrom[0].Mutate(p);  xrom[1].Mutate(p);  xrom[2].Mutate(p);  xrom[3].Mutate(p);  }  }  private void Invers(Parametrs p)  {  if (p.Chom == 2)  {  xrom[0].Invers(p);  xrom[1].Invers(p);  }  if (p.Chom == 4)  {  xrom[0].Invers(p);  xrom[1].Invers(p);  xrom[2].Invers(p);  xrom[3].Invers(p);  }  }  public void MutateAll(Parametrs p)  {  Mutate(p);  Invers(p);  }  public static VeshIndivid[] PointCrossOver(Parametrs a, VeshIndivid A, VeshIndivid B)  {  VeshIndivid[] C = new VeshIndivid[] { new VeshIndivid(a), new VeshIndivid(a) };  double[] s = new double[2];  if (a.Chom == 2)  { s=Point\_CrossOver(a, A.xrom[0], B.xrom[0]);  C[0].xrom[0].chrom = s[0];  C[1].xrom[0].chrom = s[1];  s = Point\_CrossOver(a, A.xrom[1], B.xrom[1]);  C[0].xrom[1].chrom = s[0];  C[1].xrom[1].chrom = s[1];  }  if (a.Chom == 4)  {s = Point\_CrossOver(a, A.xrom[0], B.xrom[0]);  C[0].xrom[0].chrom = s[0];  C[1].xrom[0].chrom = s[1];  s = Point\_CrossOver(a, A.xrom[1], B.xrom[1]);  C[0].xrom[1].chrom = s[0];  C[1].xrom[1].chrom = s[1];  s = Point\_CrossOver(a, A.xrom[2], B.xrom[2]);  C[0].xrom[2].chrom = s[0];  C[1].xrom[2].chrom = s[1];  s = Point\_CrossOver(a, A.xrom[3], B.xrom[3]);  C[0].xrom[3].chrom = s[0];  C[1].xrom[3].chrom = s[1];  }  return C; }  // кроссенговер VeshChrom  private static double[] Point\_CrossOver(Parametrs A, VeshChrom chromA, VeshChrom chromB)  {  // однототечный кроссенговер  double[] chrom = new double[2];  chrom[0] = A.point0 \* chromA.chrom + (1 - A.point0) \* chromB.chrom;  chrom[1]= (1 - A.point0) \* chromA.chrom + A.point0 \* chromB.chrom;  return chrom;  }  public double Fenotip\_1(Parametrs a)  { return xrom[0].chrom; }  public double Fenotip\_2(Parametrs a)  { return xrom[1].chrom; }  public double Fenotip\_3(Parametrs a)  { return xrom[2].chrom; }  public double Fenotip\_4(Parametrs a)  { return xrom[3].chrom; }    public string Prin\_Fen(Parametrs a)  {  if (a.Chom == 2)  {  return "1) " + Fenotip\_1(a) + Environment.NewLine + " 2) " + Fenotip\_2(a) + Environment.NewLine;  }  if (a.Chom == 4)  {  return "1) " + Fenotip\_1(a) + Environment.NewLine + " 2) " + Fenotip\_2(a) + Environment.NewLine +  "3) " + Fenotip\_3(a) + Environment.NewLine + " 4) " + Fenotip\_4(a) + Environment.NewLine;  }  return "0\_0 ";  }  } |
| class Population  {  public Individ[] Popul\_all;  public Population(Parametrs a)  {Popul\_all = Popul\_0(a); }  public double Fank\_(Individ s, Parametrs a)//double Fenotip\_01, double Fenotip\_02)  {  double x1=0, x2=0, x3=0, x4=0;  if (a.Chom == 4)  { x1 = s.Fenotip\_1(a);  x2 = s.Fenotip\_2(a);  x3 = s.Fenotip\_3(a);  x4 = s.Fenotip\_4(a);  }  else  {  x1 = s.Fenotip\_1(a);  x2 = s.Fenotip\_2(a);  }  double ouT;    switch (a.Fanc)  {  case 1:  ouT = (x2 - x1 \* x1) \* (x2 - x1 \* x1) + (1 - x1) \* (1 - x1);  break;  case 4:  ouT= 4 \* (x1 - 5) \* (x1 - 5) + (x2 - 6) \* (x2 - 6);  break;  case 5:  ouT = (x2 - x1 \* x1) \* (x2 - x1 \* x1) + (1 - x1) \* (1 - x1);  break;  case 12:  ouT = 4 \* x1 \* x1 + 3 \* x2 \* x2 - 4 \* x1 \* x2 \* x2 + x1;  break;  case 15:  ouT = (1.5 - x1 \* (1 - x2)) \* (1.5 - x1 \* (1 - x2)) + (2.25 - x1 \* (1 - x2 \* x2)) \* (2.25 - x1 \* (1 - x2 \* x2)) +  (2.625 - x1 \* (1 - x2 \* x2 \* x2)) \* (2.625 - x1 \* (1 - x2 \* x2 \* x2));  break;  case 17:  ouT = 100 \* (x2 - x1 \* x1) \* (x2 - x1 \* x1) + (1 - x1) \* (1 - x1) + 90 \* (x4 - x3 \* x3) \* (x4 - x3 \* x3) +  (1 - x3) \* (1 - x3) \* (1 - x3) + 10.1\*((x2 - 1) \* (x2 - 1) + (x4 - 1) \* (x4 - 1)) + 19.8\*(x2 - 1) \* (x4 - 1);  break;  default:  ouT = 4 \* (x1 - 5) \* (x1 - 5) + (x2 - 6) \* (x2 - 6);  break;  }  return ouT;  }  Individ[] Popul\_0(Parametrs a) // новая популяция (прям новая первая)  {  Individ[] Ind\_popul = new Individ[a.Popultion];  for (int i = 0; i < a.Popultion; i++)  {  Ind\_popul[i] = new Individ(a);  Ind\_popul[i].Fank\_out = Fank\_( Ind\_popul[i], a );  }  return Ind\_popul;  }  public Individ Turnir(Individ[] Tur)  { if (Tur.Length > 2)  {  int i;  Individ[] Tur2 = new Individ[Tur.Length - 1];  for (i = 0; i < Tur2.Length - 1; i++)  Tur2[i] = Tur[i];  if (Tur[Tur.Length - 1].Fank\_out > Tur[Tur.Length - 2].Fank\_out)  Tur2[i] = Tur[Tur.Length - 2];  else Tur2[i] = Tur[Tur.Length - 1];  return Turnir(Tur2);  } else  { if (Tur[1].Fank\_out > Tur[0].Fank\_out)  return Tur[0];  else return Tur[1];}}  public string Print(Parametrs A)  {  string s = "";  for (int i = 0; i < A.Popultion; i++)  {s = s  Popul\_all[i].Fank\_out + Environment.NewLine;  }  return s; }  public void Select(Parametrs a)  Individ[] Ind\_popul = new Individ[a.Popultion];  Individ f, g;  Individ[] C = { new Individ(a), new Individ(a) };  for (int i = 0; i < (int)Math.Floor(a.Popultion/2.0); i++)  { f = Turnir2(Rand\_Turnir(a));  g = Turnir2(Rand\_Turnir(a));  C = Individ.PointCrossOver(a, f, g);  C[0].MutateAll(a);  C[1].MutateAll(a);  Ind\_popul[2 \* i] = C[0];  Ind\_popul[2 \* i + 1] = C[1];  Ind\_popul[2 \* i].Fank\_out = Fank\_(Ind\_popul[2 \* i], a);  Ind\_popul[2 \* i + 1].Fank\_out = Fank\_(Ind\_popul[2 \* i + 1], a);  }  Popul\_all = Ind\_popul; }  public Individ[] Rand\_Turnir(Parametrs a)  {  Individ[] Tur = new Individ[a.Tutu];  int ii = 0;  for (int i = 0; i < a.Tutu; i++)  {  ii = a.rnd1.Next(0, a.Popultion - 1);  Tur[i] = Popul\_all[ii];  }  return Tur;  }  public void Mut\_all(Parametrs a)  {  for (int i = 0; i < Popul\_all.Length; i++)  {  Popul\_all[i].MutateAll(a);    }  }  public Individ Turnir2(Individ[] Tur)  {  if (Tur.Length > 2)  {  int i;  Individ[] Tur2 = new Individ[Tur.Length - 1];  for (i = 0; i < Tur2.Length - 1; i++)  Tur2[i] = Tur[i];  if (Math.Abs(Tur[Tur.Length - 1].Fank\_out) > Math.Abs(Tur[Tur.Length - 2].Fank\_out))  Tur2[i] = Tur[Tur.Length - 2];  else Tur2[i] = Tur[Tur.Length - 1];  return Turnir2(Tur2);  }  else  {  if (Math.Abs(Tur[1].Fank\_out) > Math.Abs(Tur[0].Fank\_out))  return Tur[0];  else return Tur[1];  }  }  public void Copi(Parametrs a, Population c)  {  for (int i = 0; i < a.Popultion; i++)  Popul\_all[i] = Individ.CopiInd(a, c.Popul\_all[i], Popul\_all[i]);  } | class VeshPopulation  {  public VeshIndivid[] Popul\_all;  public VeshPopulation(Parametrs a)  {Popul\_all = Popul\_0(a); }  VeshIndivid[] Popul\_0(Parametrs a) // новая популяция (прям новая первая)  {  VeshIndivid[] Ind\_popul = new VeshIndivid[a.Popultion]; //(L, Mut, Inv, point, point2, inter);  for (int i = 0; i < a.Popultion; i++)  {Ind\_popul[i] = new VeshIndivid(a);  Ind\_popul[i].Fank\_out = Fank\_(Ind\_popul[i], a);}  return Ind\_popul;  }  public double Fank\_(VeshIndivid s, Parametrs a)//double Fenotip\_01, double Fenotip\_02)  {double x1 = 0, x2 = 0, x3 = 0, x4 = 0;  if (a.Chom == 4)  {  x1 = s.Fenotip\_1(a);  x2 = s.Fenotip\_2(a);  x3 = s.Fenotip\_3(a);  x4 = s.Fenotip\_4(a);  }  else  {  x1 = s.Fenotip\_1(a);  x2 = s.Fenotip\_2(a);  }  double ouT;  switch (a.Fanc)  {  case 1:  ouT = (x2 - x1 \* x1) \* (x2 - x1 \* x1) + (1 - x1) \* (1 - x1);  break;  case 4:  ouT = 4 \* (x1 - 5) \* (x1 - 5) + (x2 - 6) \* (x2 - 6);  break;  case 5:  ouT = (x2 - x1 \* x1) \* (x2 - x1 \* x1) + (1 - x1) \* (1 - x1);  break;  case 12:  ouT = 4 \* x1 \* x1 + 3 \* x2 \* x2 - 4 \* x1 \* x2 \* x2 + x1;  break;  case 15:  ouT = (1.5 - x1 \* (1 - x2)) \* (1.5 - x1 \* (1 - x2)) + (2.25 - x1 \* (1 - x2 \* x2)) \* (2.25 - x1 \* (1 - x2 \* x2)) +  (2.625 - x1 \* (1 - x2 \* x2 \* x2)) \* (2.625 - x1 \* (1 - x2 \* x2 \* x2));  break;  case 17:  ouT = 100 \* (x2 - x1 \* x1) \* (x2 - x1 \* x1) + (1 - x1) \* (1 - x1) + 90 \* (x4 - x3 \* x3) \* (x4 - x3 \* x3) +  (1 - x3) \* (1 - x3) \* (1 - x3) + 10.1 \* ((x2 - 1) \* (x2 - 1) + (x4 - 1) \* (x4 - 1)) + 19.8 \* (x2 - 1) \* (x4 - 1);  break;  default:  ouT = 4 \* (x1 - 5) \* (x1 - 5) + (x2 - 6) \* (x2 - 6);  break;} return ouT}  public VeshIndivid Turnir(VeshIndivid[] Tur)  {  if (Tur.Length > 2)  {  int i;  VeshIndivid[] Tur2 = new VeshIndivid[Tur.Length - 1];  for (i = 0; i < Tur2.Length - 1; i++)  Tur2[i] = Tur[i];  if (Math.Abs(Tur[Tur.Length - 1].Fank\_out) > Math.Abs(Tur[Tur.Length - 2].Fank\_out))  Tur2[i] = Tur[Tur.Length - 2];  else Tur2[i] = Tur[Tur.Length - 1];  return Turnir(Tur2);  }  else  {  if (Math.Abs(Tur[1].Fank\_out) > Math.Abs(Tur[0].Fank\_out))  return Tur[0];  else return Tur[1];  } }  public string Print(Parametrs A)  { string s = "";  for (int i = 0; i < A.Popultion; i++)  { s = s +  Popul\_all[i].Fank\_out + Environment.NewLine; }  return s;}  public void Select(Parametrs a)  {  VeshIndivid[] Ind\_popul = new VeshIndivid[a.Popultion];  VeshIndivid f, g;  VeshIndivid[] C = { new VeshIndivid(a), new VeshIndivid(a) };  // Individ[] q = new Individ[a.Tutu];  for (int i = 0; i < (int)Math.Floor(a.Popultion / 2.0); i++)  {  f = Turnir(Rand\_Turnir(a));  g = Turnir(Rand\_Turnir(a));  C = VeshIndivid.PointCrossOver(a, f, g);  C[0].MutateAll(a);  C[1].MutateAll(a);  Ind\_popul[2 \* i] = C[0];  Ind\_popul[2 \* i + 1] = C[1];  Ind\_popul[2 \* i].Fank\_out = Fank\_(Ind\_popul[2 \* i], a);  Ind\_popul[2 \* i + 1].Fank\_out = Fank\_(Ind\_popul[2 \* i + 1], a); }  Popul\_all = Ind\_popul; }  public VeshIndivid[] Rand\_Turnir(Parametrs a)  {  VeshIndivid[] Tur = new VeshIndivid[a.Tutu];  int ii = 0;  for (int i = 0; i < a.Tutu; i++)  {  ii = a.rnd1.Next(0, a.Popultion - 1);  Tur[i] = Popul\_all[ii];  }  return Tur;  }  public void Mut\_all(Parametrs a)  {  for (int i = 0; i <= Popul\_all.Length - 1; i++)  {  Popul\_all[i].MutateAll(a);  }  } |
| // целочисленная кодировка  private void Test\_Popultion()  {  Population test = new Population(Aa);  // создание популяции  //int Namber0, int nIters0, double pMut0, double pInv0  // ( (int L, float Mut, float Inv, int point, int point2, int Namber0, int nIters0)  Individ test\_Ind = test.Turnir2(test.Popul\_all);  // MtextBox.Text = MtextBox.Text + "------- + 0 + -------" + Environment.NewLine;  // MtextBox.Text = MtextBox.Text + test.Print( point, point2) + Environment.NewLine;  // MtextBox.Text = MtextBox.Text + test\_Ind.Fank\_out + Environment.NewLine;  // MtextBox.Text = MtextBox.Text + test\_Ind.Prin\_Fen(Aa) + Environment.NewLine;  Pr\_param();  MtextBox.Text = MtextBox.Text + "------- + 0 + -------" + Environment.NewLine  // + test.Print(Aa) //;  + Environment.NewLine+ "Min " + test\_Ind.Fank\_out+Environment.NewLine;  for (int i = 0; i < Aa.Itertion; i++)  {  test.Mut\_all(Aa);  test.Select(Aa);  test\_Ind = test.Turnir2(test.Popul\_all);  MtextBox.Text = MtextBox.Text + //"-------" + (i+1) + "-------" + Environment.NewLine +  // test.Print(point, point2)+ Environment.NewLine + "-----" + Environment.NewLine +  // test\_Ind.Fank\_out + Environment.NewLine;  // test.Print(Aa) + Environment.NewLine +  "Min " + (i + 1) +" " + test\_Ind.Fank\_out + Environment.NewLine;  test.Mut\_all(Aa);  //MtextBox.Text = MtextBox.Text + test\_Ind.Prin\_Fen() + Environment.NewLine;    }    //test\_Ind = test.Turnir(test.Popul\_all);  MtextBox.Text = MtextBox.Text + "-------END-------" + Environment.NewLine +  // test.Print(point, point2) + Environment.NewLine + "-----" +  Environment.NewLine + test\_Ind.Prin\_Fen(Aa) + Environment.NewLine;  // MtextBox.Text = MtextBox.Text + test\_Ind.Prin\_Fen() + Environment.NewLine;  MtextBox.Text = MtextBox.Text + "-------------" + Environment.NewLine +  test\_Ind.xrom[0].Prin(Aa) + Environment.NewLine + test\_Ind.xrom[1].Prin(Aa) + Environment.NewLine;  /////для 4  x1 = test\_Ind.Fenotip\_1(Aa);  x2 = test\_Ind.Fenotip\_2(Aa);  } | //вещественная кодировка  private void Test\_VeshPopultion()  {  VeshPopulation test = new VeshPopulation(Aa);  // создание популяции  //int Namber0, int nIters0, double pMut0, double pInv0  // ( (int L, float Mut, float Inv, int point, int point2, int Namber0, int nIters0)  VeshIndivid test\_Ind = test.Turnir(test.Popul\_all);  // MtextBox.Text = MtextBox.Text + "------- + 0 + -------" + Environment.NewLine;  // MtextBox.Text = MtextBox.Text + test.Print( point, point2) + Environment.NewLine;  // MtextBox.Text = MtextBox.Text + test\_Ind.Fank\_out + Environment.NewLine;  // MtextBox.Text = MtextBox.Text + test\_Ind.Prin\_Fen(Aa) + Environment.NewLine;  MtextBox.Text = MtextBox.Text + "------- + 0 + -------" + Environment.NewLine  // + test.Print(Aa) //;  + Environment.NewLine + "Min " + test\_Ind.Fank\_out + Environment.NewLine;  for (int i = 0; i < Aa.Itertion; i++)  {  test.Select(Aa);  test\_Ind = test.Turnir(test.Popul\_all);  MtextBox.Text = MtextBox.Text + //"-------" + (i+1) + "-------" + Environment.NewLine +  // test.Print(point, point2)+ Environment.NewLine + "-----" + Environment.NewLine +  // test\_Ind.Fank\_out + Environment.NewLine;  // test.Print(Aa) + Environment.NewLine +  "Min " + (i + 1) + " " + test\_Ind.Fank\_out + Environment.NewLine;  test.Mut\_all(Aa);  //MtextBox.Text = MtextBox.Text + test\_Ind.Prin\_Fen() + Environment.NewLine;  }  //test\_Ind = test.Turnir(test.Popul\_all);  MtextBox.Text = MtextBox.Text + "-------END-------" + Environment.NewLine +  // test.Print(point, point2) + Environment.NewLine + "-----" +  Environment.NewLine + test\_Ind.Prin\_Fen(Aa) + Environment.NewLine;  // MtextBox.Text = MtextBox.Text + test\_Ind.Prin\_Fen() + Environment.NewLine;  MtextBox.Text = MtextBox.Text + "-------------" + Environment.NewLine +  test\_Ind.xrom[0].Prin() + Environment.NewLine + test\_Ind.xrom[1].Prin() + Environment.NewLine;  /////для 4  x1 = test\_Ind.xrom[0].chrom;  x2 = test\_Ind.xrom[1].chrom;  } |
| class Parametrs  {  // private static int \_seed = Environment.TickCount;  public Random rnd1 = new Random(0);  int LeGen, Popul, Iter, fanc;  int Tut; // проверка турника  public bool Kod = false;  public bool Ipoint = true;  int[] point = new int[] { 0 , 0 }; //point1 , point2,  public double point0;    //Уплотнение сетки  /////////  public int DopIter, Setka;  public bool Set = false;  /////////  float Mut, Inv;  public float[,] interM = new float[,] { { 0, 0 }, { 0, 0 }, { 0, 0 }, { 0, 0 } };  public int Chom=2;  public string form = "1) (x2 - x1\* x1)\*(x2 - x1\* x1) + (1 - x1)\*(1 - x1)" + Environment.NewLine +  " 4)= 4\*(x1 - 5)^2 + (x2 - 6)^2; (5 ; 6 ) Min=0" + Environment.NewLine +  " 5)= (x2 -x1^2)^2 + (1 - x1)^2; (1 ; 1 ) Min=0 " + Environment.NewLine +  "12)= 4\*x1^2 + 3\*x2^2 - 4\*x1\*x2^2 + x1 (0 ; 0 ) Min=0 " + Environment.NewLine +  "15)= (1.5-x1\*(1-x2))^2+(2.25-x1\*(1-x2^2))^2" + Environment.NewLine +  " +(2.625 - x1\*(1-x2^3))^2; (3 ; 0,5 ) Min=0 " + Environment.NewLine +  "17)= 100\*(x2 - x1^2)^2 + (1-x1)^2+ " + Environment.NewLine +  " 90\*(x4-x3^2)^2+(1-x3)^2 +10.1((x2 - 1)^2" + Environment.NewLine +  " + (x4-1)^2) + 19.8(x2-1)\*(x4-1); (1; 1; 1; 1) Min=0 ";  //xrom\_1 = new Chromosom(L);  //xrom\_2 = new Chromosom(L);  //Min = inter0[0, 0]; //new float[,] { { inter0[0, 0], inter0[0, 1] } };  //Max = inter0[0, 1];  public Parametrs()  { }    //возможность измененния длины гена  public int Length  {  get  {  return LeGen;  }  set  {  LeGen = value;  }  }  public int Popultion  {  get  {  return Popul;  }  set  {  Popul = value;  }  }  public int Itertion  {  get  {  return Iter;  }  set  {  Iter = value;  }  }  public float Muttion  {  get  {  return Mut;  }  set  {  Mut = value;  }  }  public float Invtion  {  get  {  return Inv;  }  set  {  Inv = value;  }  }  public int point1  {  get  {  return point[0];  }  set  {  point[0] = value;  }  }  public int point2  {  get  {  return point[1];  }  set  {  point[1] = value;  }  }  public int Tutu  {  get  {  return Tut;  }  set  {  Tut = value;  }  }  public int Fanc  {  get  {  return fanc;  }  set  {  fanc = value;  }  } | |