**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра САПР**

отчет

**по лабораторной работе №2**

**по дисциплине «Основы искусственного интеллекта»**

Тема: Генетический алгоритм

|  |  |  |
| --- | --- | --- |
| Студенты гр. 9302 |  | Ширнин К.В. |
|  |  | Квитко Д.В. |
| Преподаватель |  | Новакова Н.Е. |

Санкт-Петербург

2022

**Оглавление**

[Цель работы 3](#_Toc115821449)

[Реализация алгоритма 3](#_Toc115821450)

[Ход работы 4](#_Toc115821451)

[Пример работы программы 5](#_Toc115821452)

[Вывод 6](#_Toc115821453)

[Список используемой литературы 6](#_Toc115821454)

# Цель работы

Изучить принцип работы генетического алгоритма. Написать ПО с пользовательским графическим интерфейсом, находящее максимум заданной функции, с использованием генетического алгоритма

# Реализация алгоритма

Для реализации генетического алгоритма ПО было разделено на две составляющие, а именно на GUI и сам программный код алгоритма. Программный комплекс был написан в рамках правил ООП и без особых трудностей может использоваться как отдельный виджет в других ПО или как библиотека.

В начале алгоритм генерирует популяцию, размер которых задан пользователем. Популяция состоит из хромосом, размер которых фиксирован и равен двум, так как для данной задачи этого будет более чем достаточно – в дальнейшем будем называть это “гены”. Хромосомы в одном своём гене будут хранить *x,* а в другом *y*. При инициализации алгоритма их значения заполняются случайным образом.

После генерации происходит оценка поколения на “выживаемость”. В нашем случае берется каждая хромосома со своими параметрами *x*’ов и *y*’ов и подставляется в заданную изначально функцию. Чем больше будет результат – тем лучше. (Так как наша задача найти максимум).  
 Далее происходит генерация следующего поколения. Хромосома, давшая лучший результат, переходит в следующее поколение без изменений. Остальные хромосомы начинают скрещиваться с заданной пользователем вероятностью. Скрещивание происходит случайным образом между хромосомами – хромосоме из следующего поколения может достаться случайный *x* ген и случайный *y* от текущего поколения.

После этого происходит мутация хромосом с заданной пользователем вероятностью. Мутация позволяет выйти алгоритму из зацикливаний – она разбавляет поколения новыми генами.   
 Вышеописанный алгоритм повторяется столько раз, сколько задал пользователь параметром “количество поколений”.

# UML Диаграммы

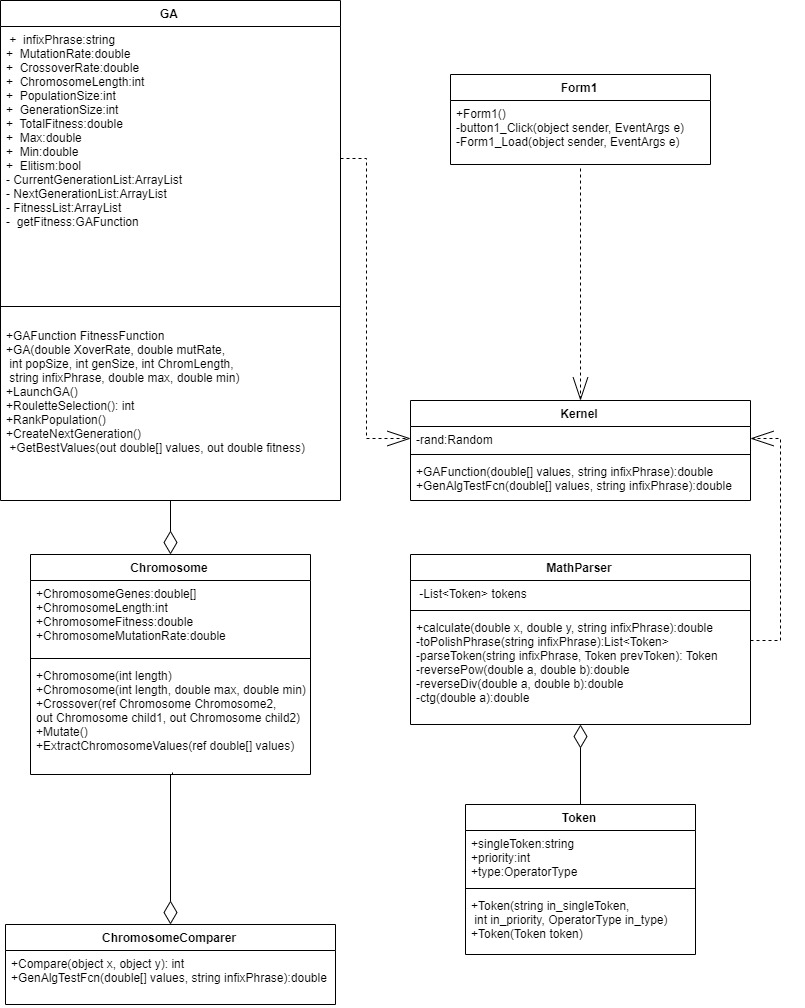


Рис.1. UML диаграмма

Таблица.1.Методы и их описание

|  |  |
| --- | --- |
| Метод | Описание |
| public GA(double XoverRate, double mutRate, int popSize, int genSize, int ChromLength, string infixPhrase, double max, double min) | Конструктор класса GA |
| public void LaunchGA() | Запуск генетического алгоритма |
| private int RouletteSelection() | Метод которые выполняет рулеточную селекцию |
| private void RankPopulation() | Высчитывает значение приспособленности |
| private void CreateNextGeneration() | Запускает одну итерацию цикла |
| public void GetBestValues(out double[] values, out double fitness) | Получает наибольшее значение приспособленности |
| public Chromosome(int length) | Конструктор класса Chromosome |
| public Chromosome(int length, double max, double min) | Конструктор класса Chromosome |
| public void Crossover(ref Chromosome Chromosome2, out Chromosome child1, out Chromosome child2) | Скрещивание хромосом |
| public void Mutate() | Мутация хромосом |
| public int Compare(object x, object y) | Сравнивание приспособленности хромосом |
| public Token(string in\_singleToken, int in\_priority, OperatorType in\_type) | Конструктор класса Token |
| public Token(Token token) | Конструктор класса Token |
| public static double calculate(double x, double y, string infixPhrase) | Вычисление значения заданной функции |
| private static List<Token> toPolishPhrase(string infixPhrase) | Переводит функцию в польскую нотацию |
| private static Token parseToken(string infixPhrase, Token prevToken) | Считывает токен из инфиксного математического выражения |
| private static double reversePow(double a, double b) | Возведение в квадрат |
| private static double reverseDiv(double a, double b) | Деление числа |
| private static double ctg(double a) | Вычисление ctg |

# Ход работы

Для выполнения данной лабораторной работы был выбран язык C#.

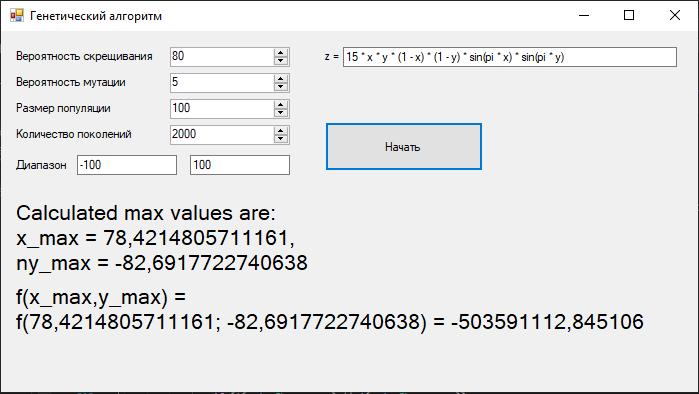


Рис.2. Главное окно приложения

При выполнении работы, были взяты основные используемые параметры:

* Вероятность скрещивания (0-100)
* Вероятность мутации (0-100)
* Размер популяции (3-5000)
* Количество поколений (1-5000)

В качестве решаемой задачи была взята функция

Для задачи параметров используются элементы WinForms “NumericUpDown”, что позволяет избежать ошибок с некорректным вводом данных.

Пользователю достаточно заполнить все предложенные поля и нажать на кнопку “Начать”, после чего программа выдаст рассчитанные значения для *x* и *y*, при выбранных параметрах.

При выполнении работы также были учтены крайние значения, вводимые пользователем в программу.

# Пример работы программы

Пример работы программы представлен на рисунках ниже.

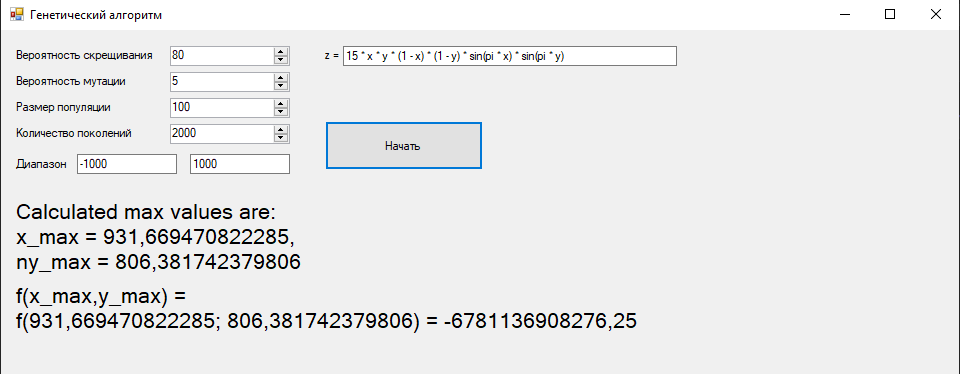


Рис. 3. Пример работы программы 1

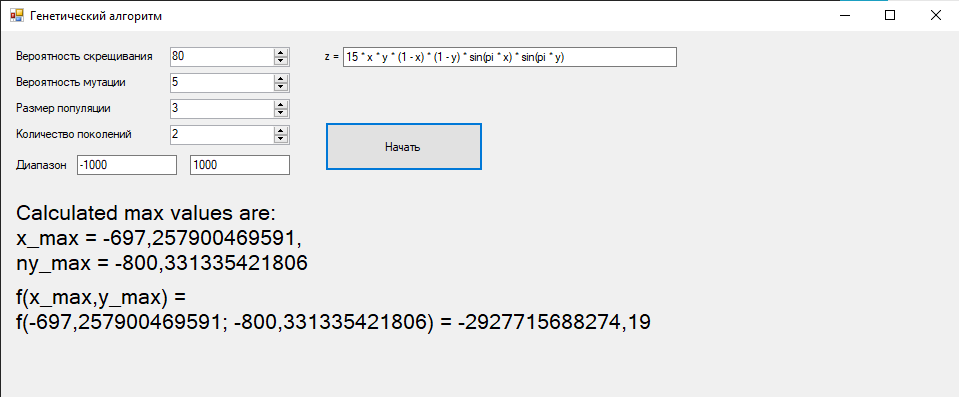


Рис.4. Пример работы программы 2

# Вывод

В ходе выполнения работы стало ясно, что генетический алгоритм очень сильно зависим от заданных параметров. К каждой функции придется прдбирать свои индивидуальные параметры. Если подобрать конфигурацию неверно – алгоритм может “свалиться” в локальный максимум, что не имеет никакой практической ценности для решения поставленных задач. Алгоритм был успешно реализован на языке C#. В лабораторной работе были достигнуты все цели.

# Список используемой литературы

1. Генетический алгоритм // Википедия URL: <https://ru.wikipedia.org/wiki/%D0%93%D0%B5%D0%BD%D0%B5%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC> (дата обращения: 0410.2022).