Министерство науки и высшего образования

Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Кузбасский государственный технический университет

имени Т.Ф. Горбачева»

Кафедра прикладных информационных технологий

Пояснительная записка к курсовой работе на тему:

**«Разработка программного модуля для поиска экстремумов функций многих переменных с использованием генетических алгоритмов»**

Выполнил студент группы ПИб – 182  
Пономаренко Д.В.

Преподаватель:

Ещин Е.К.

Кемерово 2019

# Реферат

Цель курсовой работы: изучить процесс поиска экстремумов функций двух переменных с использованием генетических алгоритмов. Основные результаты курсовой работы: найти значение максимума или минимума функции и построить график исследуемой функции.

Объем пояснительной записки: 18 страниц, 3830 слов.

Количество рисунков: 13

Ключевые слова: Программа, максимум, минимум, экстремум, Delphi, функция, генетический алгоритм, отбор, скрещивание, мутация.

**СОДЕРЖАНИЕ**

[Реферат 1](#_Toc9849721)

[ВВЕДЕНИЕ 3](#_Toc9849722)

[2. Основная часть 4](#_Toc9849723)

[2.1 Техническое задание 4](#_Toc9849724)

[2.2 Описание используемого математического аппарата. 5](#_Toc9849725)

[2.3 Руководство пользователя. 8](#_Toc9849726)

[2.3.1 Общие сведения 8](#_Toc9849727)

[2.3.2 Системные требования 8](#_Toc9849728)

[2.3.3 Установка программного продукта. 9](#_Toc9849729)

[2.3.4 Использование программного продукта. 9](#_Toc9849730)

[2.3.5 Удаление программного продукта. 10](#_Toc9849731)

[2.4 Руководство программиста. 10](#_Toc9849732)

[2.4.1 Введение. 10](#_Toc9849733)

[2.4.2 Общие сведения 11](#_Toc9849734)

[2.4.3 Структура программного продукта 11](#_Toc9849735)

[2.4.4 Описание типов данных 12](#_Toc9849736)

[2.4.5 Описание процедур и функций. 13](#_Toc9849737)

[2.4.6 Результаты тестирования и опытной эксплуатации 15](#_Toc9849738)

[ЗАКЛЮЧЕНИЕ 17](#_Toc9849739)

[СПИСОК ЛИТЕРАТУРЫ 18](#_Toc9849740)

[Приложение А 19](#_Toc9849741)

[Приложение Б 2](#_Toc9849742)

# ВВЕДЕНИЕ

Генетические алгоритмы – это адаптивные методы поиска, которые в последнее время используются для решения задач оптимизации. В них используются как аналог механизма генетического наследования, так и аналог естественного отбора. [1, c. 6]

Цель разработки: разработать программный модуль для поиска экстремумов функций двух переменных с использованием генетических алгоритмов.

Генетические алгоритмы успешно применяются в различных областях деятельности: экономика, физика, технические науки и т.п.

Авторское наименование программного продукта: программапоиска экстремумов функций двух переменных с использованием генетических алгоритмов.

# 2. Основная часть

## 2.1 Техническое задание

При выполнении курсового проекта необходим решить следующие задачи:

* рассмотреть три генетических оператора: отбора (select), скрещивания (crossover)[[1]](#footnote-1) и мутации[[2]](#footnote-2) (mutation). Выбрать оптимальные методы их реализации.
* Найти локальный минимум и максимум функции (1) и функции (2) в заданной области;

|  |  |
| --- | --- |
|  | (1) |
|  | (2) |
| * построить график для функции; * разработать удобный и понятный интерфейс; * реализовать защиту от ввода некорректных данных; |  |

**Функциональное назначение программы:**

Исходя из выбора пользователя графика функции, программа строит её график и находит точку максимума или минимума.

**Определение структуры входных и выходных данных:**Для работы программы необходимо ввести интервал поиска, количество поколений, размер популяции, вероятность мутации (в %), вероятность сращивания (в %), выбрать график и выбрать что необходимо искать (минимум или максимум).

После окончания работы программы будут выведены следующие данные: график функции и значение максимума или минимума;

## 2.2 Описание используемого математического аппарата.

Основные принципы работы генетического алгоритма (ГА) заключены в следующей схеме (смотреть также рисунок 1):

1. Генерируем начальную популяцию[[3]](#footnote-3) из n хромосом[[4]](#footnote-4).
2. Вычисляем для каждой хромосомы ее пригодность[[5]](#footnote-5).
3. Выбираем пару хромосом-родителей с помощью одного из способов отбора.
4. Проводим кроссинговер двух родителей с определённой вероятностью, производя двух потомков.
5. Проводим мутацию потомков с вероятностью.
6. Повторяем шаги 3–5, пока не будет сгенерировано новое поколение популяции, содержащее n хромосом.
7. Повторяем шаги 2–6, пока не будет достигнут критерий окончания процесса. [1, c. 13-14]

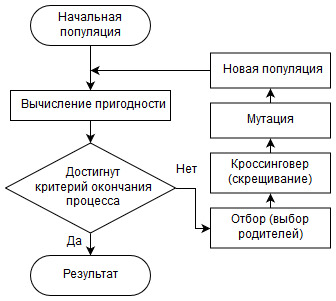


Рисунок 1 – Схема Генетического алгоритма

Критерием окончания процесса может служить заданное количество поколений или схождение (convergence) популяции. Схождением называется такое состояние популяции, когда все строки популяции почти одинаковы и находятся в области некоторого экстремума. В такой ситуации кроссинговер практически никак не изменяет популяции, так как создаваемые при нем потомки представляют собой копии родителей с перемененными участками хромосом. Вышедшие из этой области за счет мутации особи склонны вымирать, так как чаще имеют меньшую приспособленность. Таким образом, схождение популяции обычно означает, что найдено лучшее или близкое к нему решение. [1, c. 14-15]

Основными операторами ГА являются кроссинговер, мутация, выбор родителей и селекция (отбор хромосом в новую популяцию).

**Создание начальной популяции**

Перед первым шагом нужно случайным образом создать начальную популяцию; даже если она окажется совершенно неконкурентоспособной, генетический алгоритм все равно достаточно быстро переведет ее в жизнеспособную популяцию. Таким образом, на первом шаге можно особенно не стараться сделать слишком уж приспособленных особей, достаточно, чтобы они соответствовали формату особей популяции, и на них можно было подсчитать функцию приспособленности (Fitness).

**Оператор выбора**

В данной курсовой работе было решено использовать «турнирный отбор» (tournament selection). Основной сутью данного метода отбора заключается в том, что из популяции, содержащей n особей, выбираются случайным образом t особей, и лучшая из них особь записывается в промежуточный массив (рисунок 2.). [1, c. 16-17]



F=72.9

F=12.8

F=72.9

Размер турнира t=2

Популяция

Выбраны особи с пригодностью

В новую популяцию выбрана особь

Рисунок 2 – Турнирный отбор

**Кроссинговер (скрещивание)**

Рекомбинацию бинарных строк принято называть кроссинговером (кроссовером) или скрещиванием. В данном проекте используется Одноточечный кроссинговер (Single-point crossover) моделируется следующим образом. Пусть имеются две родительские особи с хромосомами XXXXXX и YYYYYY. Случайным образом определяется точка внутри хромосомы (точка разрыва), в которой обе хромосомы делятся на две части и обмениваются ими (Рисунок 3).

XX.XXXXX

XX.XXXXX

XXXXXXX

XXXXXXX

Рисунок 3 – Кроссинговер.

**Мутация**

После процесса скрещивания происходят мутации (mutation). Данный оператор необходим для «выбивания» популяции из локального экстремума и препятствует преждевременной сходимости. Это достигается за счет того, что изменяется случайно выбранный ген в хромосоме. [1, c. 23-24]

## 2.3 Руководство пользователя.

### 2.3.1 Общие сведения

Данный программный продукт позволят найти локальный максимум или мини функции. Задача поиска экстремума функции двух переменных встречается в самых различных областях физики, математики, техники и др.

### 2.3.2 Системные требования

Для нормального функционирование программы необходимы следующие аппаратные и технические средства:

* монитор или какой-либо дисплей для восприятия информации;
* компьютерная «мышь», включающая в себя кнопку и колесико для удобства работы (также подойдет «тачпад», сенсор);
* клавиатура для ввода данных;
* компьютер или иное устройство с минимальными системными требованиями: (процессор – частота 500 МГц, ОЗУ – 2 ГБ, свободное место на жестком диске 1 ГБ), а также необходима операционная система Microsoft Windows XP/7/8/10 или платформа Linux.

### 2.3.3 Установка программного продукта.

Для установки программного продукта необходимо загрузить с внешнего носителя файл «Project1.exe» на свой компьютер.

### 2.3.4 Использование программного продукта.

Запуск программы происходит двойным кликом по файлу «Project1.exe». После чего откроется главная форма (пользовательский интерфейс) программы (Рисунок 4).

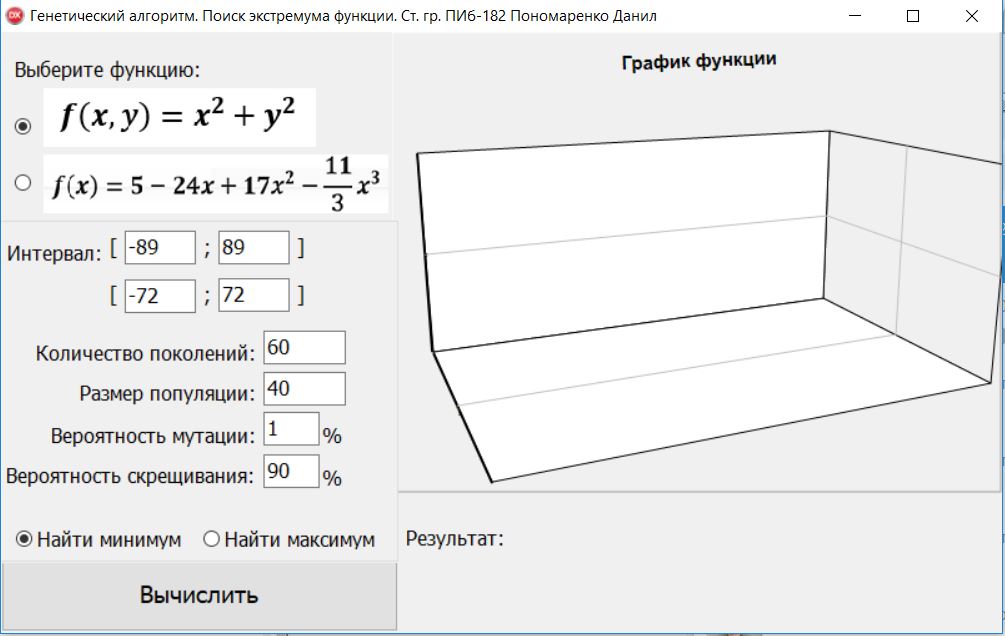


Рисунок 4 – Пользовательский интерфейс

После заполнения всех необходимых необходимо нажать на кнопку «Вычислить». На рисунке 5 показана форма после нажатии на кнопку «Вычислить», где показан результат работы программы для функции (1).

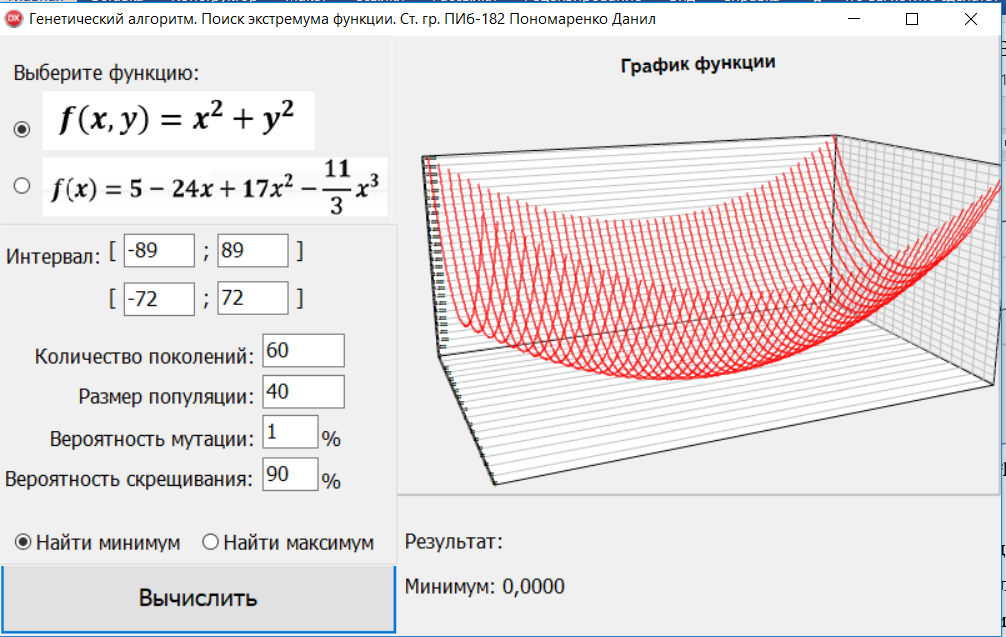


Рисунок 5 Результат работы программы для функции (1).

### 2.3.5 Удаление программного продукта.

Чтобы удалить программный продукт с компьютера необходимо файл «Project1» переместить в «Корзину» ее или же нажать на него правой кнопкой мыши и выбрать функцию «Удалить», после чего программы переместится в корзину. А для того, чтобы удалить программу без помещения в корзину необходимо нажать сочетание клавиш shift+del.

## 2.4 Руководство программиста.

### 2.4.1 Введение.

В разделе «[Общие сведения](#общсв)» будут приведены: объем, который занимает программа на жестком диске, язык программирования и его версия.

В разделе «[Структура программного продукта](#структура)» будет приведены ссылки на приложения с кодом программы и блок-схемами процедур.

В разделе «[Описание типов данных](#описание)» будет представлено краткое описание использованных типов данных.

В разделе «Описание процедур и функций», будет представлено описание всех пользовательских процедур и функций используемы в программе.

В разделе «[Результаты тестирования и опытной эксплуатации](#резтестов)» Надежность программного продукта.

### 2.4.2 Общие сведения

При разработке программы использовался компьютер со следующими техническими характеристиками:

* процессор Intel Core i7-8550U, частота 4.0 ГГц, 4 ядра;
* оперативная память: (ОЗУ) 16 ГБ, DDR4, 2400 МГц;
* видеочип / Видеокарта: NVIDIA GeForce MX150 2 Гб GDDR5;
* дисплей: матрица IPS, диагональ15.6 дюймов, разрешение FHD (1920 × 1080).

Программный продукт написан на языке программирования Delphi в среде программирования RAD (Rapid Application Development) Studio 10.1. Berlin. Файл «Project1.exe» занимает место на жестком диске 15,256 МБ.

### 2.4.3 Структура программного продукта

Программный продукт имеет следующую структуру (рисунок 6):

* файл «Unit1.pas» - cодержат исходный код программ на языке Delphi.
* Project1.dpr - файл проекта Delphi. Delphi создаёт файл проекта автоматически при первом сохранении нового приложения.
* «Unit1.dfm» - файлы описания формы. Содержат информацию о свойствах экранной формы и включённых в неё объектов, а также о событиях и процедурах их обработки.
* Project1.res. – файл ресурсов. Содержит двоичные ресурсы (пиктограммы и прочие растровые изображения), используемые в проекте.
* в папке Win32 находится сама программа «Project1.exe».

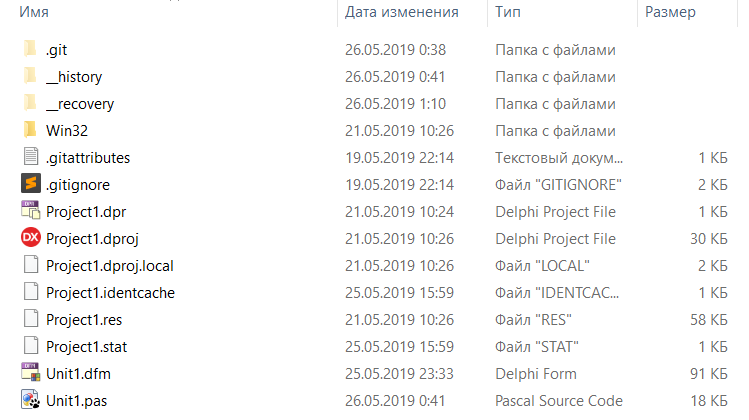


Рисунок 6 – структура программного продукта

Блок-схемы процедур и функций программы представлены в [приложении А.](#БЛОК_СХЕМЫ)

Структура программного кода представлена в [приложении Б](#КОД). Программный код также доступен в репозитории Github по следующей ссылки: <https://github.com/danilpon98/delphi_Genetic_some>.

### 2.4.4 Описание типов данных

Используются следующие типы данных:

* Boolean – позволяет принимать только True и False значения;
* Double – тип с плавающей запятой, поддерживающий приблизительно 15 цифр точности;
* Integer – основной тип целого числа минимальное значение -2147483648, а максимальное значение 2147483647.
* Record (запись)– это сложный тип данных, позволяющие объединить данные разных типов. Одна запись соответствует одной строке данных: она имеет несколько полей, каждое из которых хранит своё значение.

### 2.4.5 Описание процедур и функций.

Процедура InitPop создает начальную популяцию случайным образом с помощью вспомогательной функции Flip, которая генерирует случайное число из вариантов 0 и 1. Процедура выполняется при запуске программы и вычисляет функцию пригодности. Процедуры Select, Mutation и Crossover реализуют 3 генетических оператора.

Процедура Select реализует оператор турнирного отбора.

Функция Select1 осуществляет отбор наилучших особей для популяции. В функции происходит перемешивание популяции и выбор особей, которые перейдут в следующее поколение (имеющие наилучшее значение целевой функции). Выбираются две рядом стоящих особи и в следующее поколение переходит особь с наилучшим значением целевой функции. Отбор происходит PopSize раз (количество особей в популяции).

Функция Mutation реализует процесс инверсионной мутации. Мутация происходит с некоторой вероятностью. В процедуре используются следующие переменные и константы:

* alleleval – ген для мутации;
* PMutation – вероятность мутации;
* NMutation – счетчик мутаций;

Инверсионная мутация является самой простой мутацией. В ее случае выбирается произвольным образом ген и инвертируется его значение. Однако мутация происходит только с вероятностью PMutation т.к. в природе мутации наступают не часто. Вероятность мутации в программе можно изменять.

Процедура Crossover реализует оператор одноточечного скрещивания. Скрещивание происходит с некоторой вероятностью pcross.

Каждая выбранная таким образом пара строк скрещивается следующим образом: если скрещивание должно произойти, то наращивается счетчик скрещиваний, и потомки наследуют гены путем обмена между всеми элементами.

Случайным образом выбирается положение точки сечения (целое число в промежутке от 1 и flchrom-1, где flchrom – длина строки). Затем, путем обмена всеми элементами до и после точки сечения рождаются две новых строки. В процессе скрещивания происходит мутация.

После этого потомки замещают родительские особи в промежуточной популяции. Если же скрещивание не происходит, то потомками становятся сами родители.

Данная процедура использует следующие переменные:

* parent1, parent2 – хромосомы родителей;
* child1,child2 – хромосомы потомков;
* flchrom – длина хромосомы (количество генов);
* ncross, nmutation – счетчики количества скрещиваний и мутаций;
* jcross – точка сечения.

Процедура Generation используется для генерации нового поколения при помощи операторов отбора, скрещивания и мутации.

Для декодирования строки битов в вещественное число используется процедура Decode. Битовую строку длины N можно рассматривать как целое двоичное число I, которому соответствует некоторое вещественное значение r из заданного диапазона [Min, Max]. Это соответствие устанавливается формулой (3).

|  |  |
| --- | --- |
|  | (3) |

В данной курсовой работе n=20.

Для получения статистических данных: минимального, максимального и среднего значений целевой функции (в программе задана в виде функции ObjFunc) используется процедура Statistics. С помощью данной процедуры находятся статистические данные. В основной программе она используется следующим образом: по всей популяции считается функция пригодности, находится максимальное и минимальное значение целевой функции а также ее среднее.

Процедура Pict строит график функции графика для двух переменных, а процедура Plotting для одной переменной. Процедура plottingDots вывод на график функции одной переменной каждого индивидуума популяции в виде точки.

### 2.4.6 Результаты тестирования и опытной эксплуатации

Программный продукт оснащен защитой от некорректных данных, веденных пользователем. Например, если оставить хотя бы одно поле пустым, то пользователь уведет соответствующее сообщение (рисунок 7).

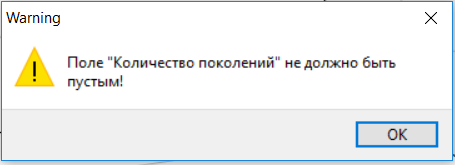


Рисунок 7 - Сообщение об ошибки

Также пользователь не сможет случайно ввести в поля буквы и не нужные символы.

Было замечено что программе необходимо какое-то время для выполнения и если несколько раз нажимать на кнопку «Вычислить», то программа будет выполнятся то количество раз сколько была нажата кнопка и продлится может это долго поэтому было решено заблокировать кнопку перед выполнением программы, а после выполнения разблокировать. Это позволило решить проблему.

# ЗАКЛЮЧЕНИЕ

В результате курсовой работы была разработана программа для поиска экстремумов функций многих переменных с использованием генетических алгоритмов. И были решены следующие задачи:

* рассмотрены три основные генетические оператора: отбора (select), скрещивания (crossover) и мутации (mutation). Выбрать оптимальные методы их реализации.
* был реализован функционал для поиска минимум и максимум функции;
* построен график для функции;
* разработан удобный и понятный интерфейс;
* реализована защиту от ввода некорректных данных;

# СПИСОК ЛИТЕРАТУРЫ

1. Панченко Т. В. Генетические алгоритмы [Текст] : учебно-методическое пособие / под ред. Ю. Ю. Тарасевича. – Астрахань: Издательский дом «Астраханский университет», 2007. — 87с.

# 

# Приложение А

**Блок-схемы программного продукта**

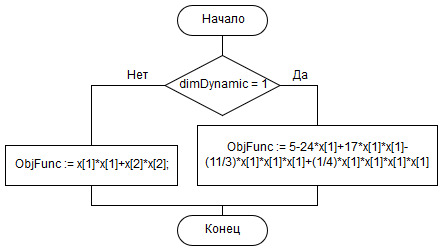


Рисунок 8 - Блок-схема функции ObjFunc

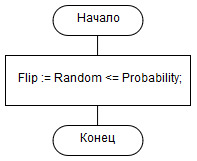


Рисунок 9 – Блок-схема функции Flip

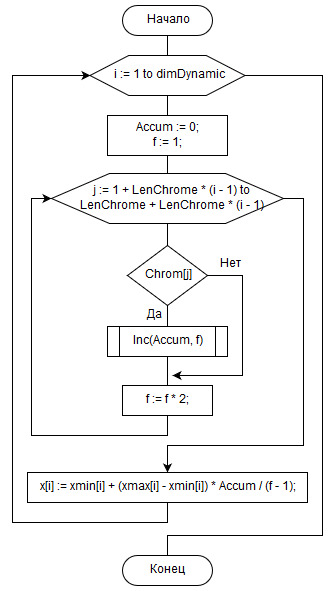


Рисунок 10 – Блок-схема процедуры Decode

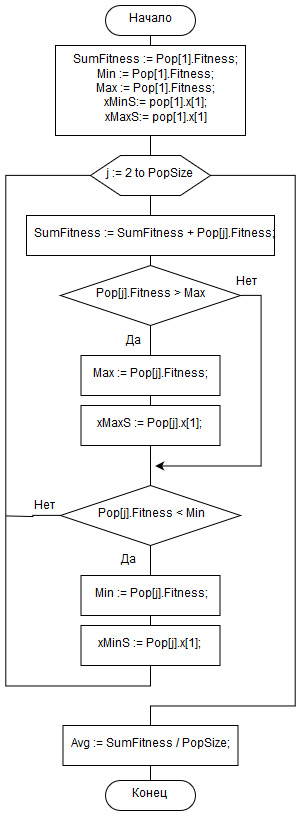


Рисунок 11 – Блок-схема процедуры Statistics

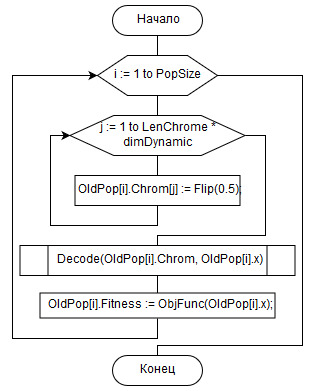


Рисунок 12 – Блок-схема процедуры InitPop

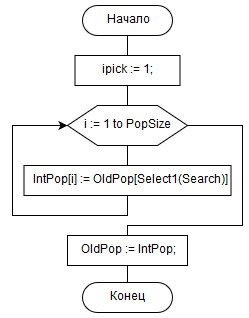


Рисунок 13 – Блок-схема процедуры Select

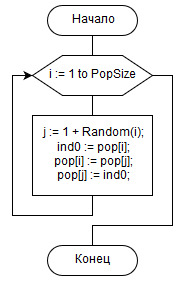


Рисунок 14 – Блок-схема процедуры Shuffle

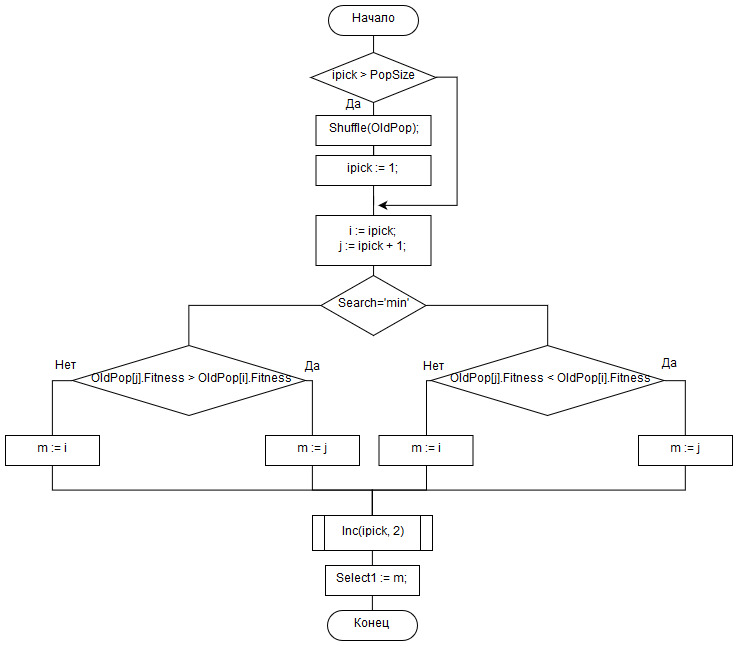


Рисунок 15 – Блок-схема процедуры Select1

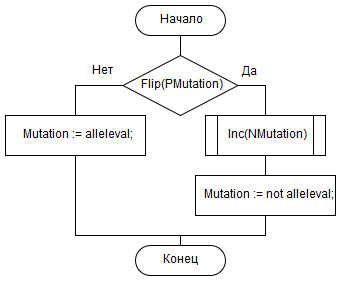


Рисунок 16 - Блок-схема процедуры Mutation

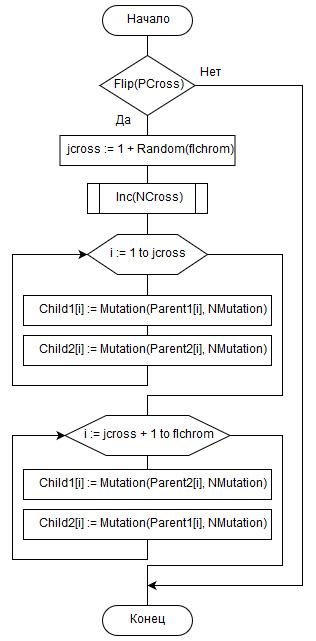


Рисунок 17 - Блок-схема процедуры Crossover

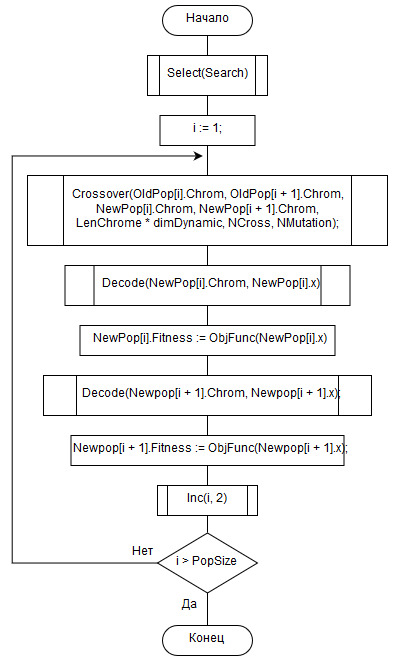


Рисунок 18 - Блок-схема процедуры Generation

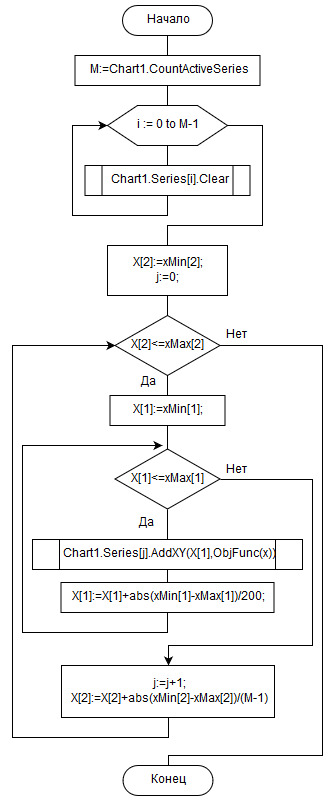


Рисунок 19 - Блок-схема процедуры Pict

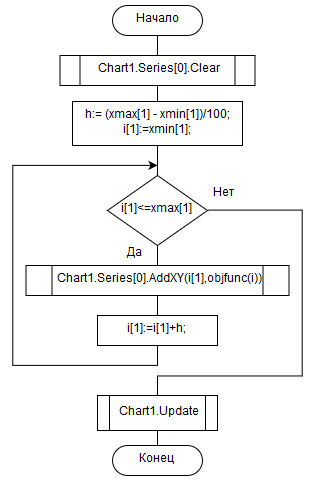


Рисунок 20 - Блок-схема процедуры plotting

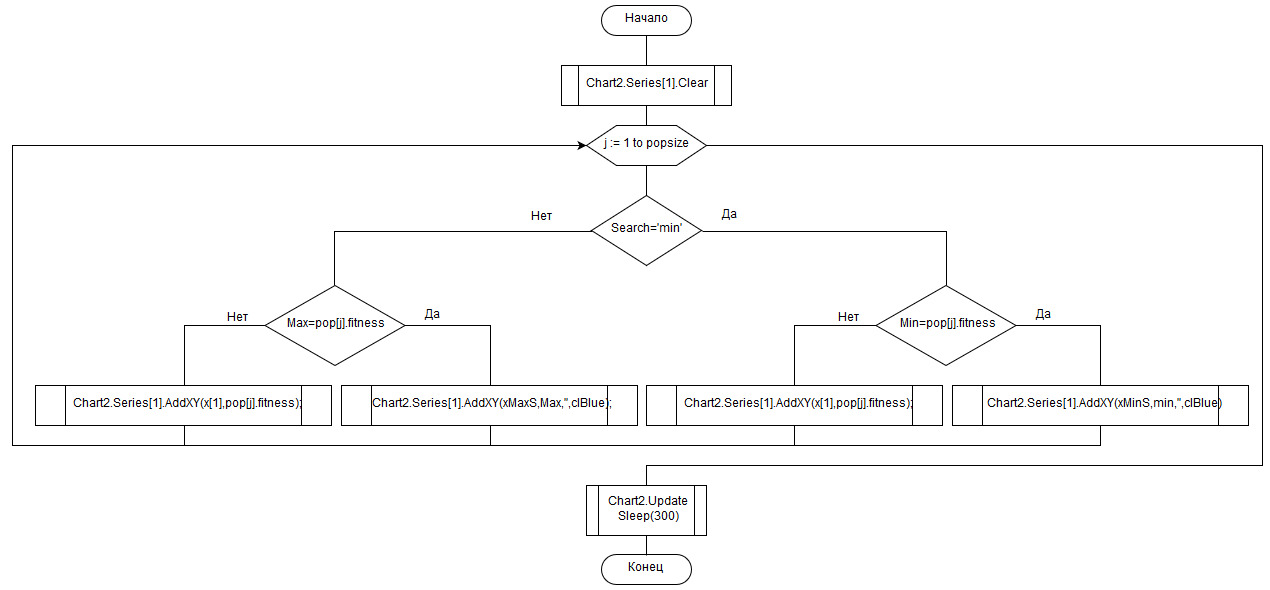


Рисунок 21 - Блок-схема процедуры plottingDots

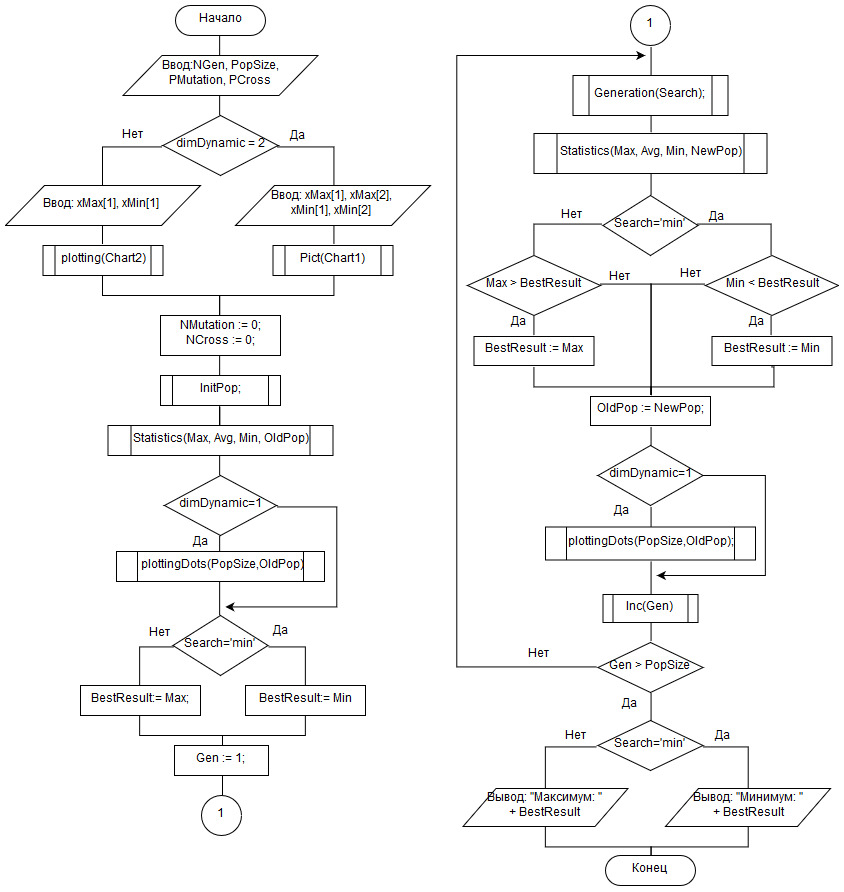


Рисунок 22 - Блок-схема алгоритма при нажатии на кнопку "Вычислить"

# Приложение Б

**Исходные тексты программного продукта**

unit Unit1;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, VclTee.TeeGDIPlus,

VCLTee.TeEngine, VCLTee.Series, Vcl.Grids, Vcl.ExtCtrls, VCLTee.TeeProcs,

VCLTee.Chart, Vcl.Imaging.jpeg, StrUtils;

const

MaxPop = 1000; { Максимальное число поколений }

LenChrome = 20; { Число битов на один кодируемый параметр }

dim = 2; { Размерность пространства поиска }

type

Allele = boolean; {Алель - позиция в битовой строке }

Chromosome = array [1..LenChrome \* Dim] of Allele; { Битовая строка }

Fenotype = array [1..Dim] of double;

Individual = record

Chrom: Chromosome; { Генотип = битовая строка }

x: Fenotype; { Фенотип = массив вещественных координат точки в пространстве поиска }

Fitness: double; { Значение целевой функции }

end;

Population = array [1..maxpop] of Individual;

TForm1 = class(TForm)

Button1: TButton;

Chart1: TChart;

Series15: TFastLineSeries;

Series1: TFastLineSeries;

Series2: TFastLineSeries;

Series3: TFastLineSeries;

Series4: TFastLineSeries;

Series5: TFastLineSeries;

Series6: TFastLineSeries;

Series7: TFastLineSeries;

Series8: TFastLineSeries;

Series9: TFastLineSeries;

Series10: TFastLineSeries;

Series11: TFastLineSeries;

Series12: TFastLineSeries;

Series13: TFastLineSeries;

Series14: TFastLineSeries;

Series16: TFastLineSeries;

Series17: TFastLineSeries;

Series18: TFastLineSeries;

Series19: TFastLineSeries;

Series20: TFastLineSeries;

Series21: TFastLineSeries;

Series22: TFastLineSeries;

Series23: TFastLineSeries;

Series24: TFastLineSeries;

Series25: TFastLineSeries;

Series26: TFastLineSeries;

Series27: TFastLineSeries;

Series28: TFastLineSeries;

Series29: TFastLineSeries;

Series30: TFastLineSeries;

Series31: TFastLineSeries;

Series32: TFastLineSeries;

Series33: TFastLineSeries;

Series34: TFastLineSeries;

Series35: TFastLineSeries;

Series36: TFastLineSeries;

Series37: TFastLineSeries;

Series38: TFastLineSeries;

Series39: TFastLineSeries;

Series40: TFastLineSeries;

Series41: TFastLineSeries;

Series42: TFastLineSeries;

Series43: TFastLineSeries;

Series44: TFastLineSeries;

Series45: TFastLineSeries;

Series46: TFastLineSeries;

Series47: TFastLineSeries;

Series48: TFastLineSeries;

Series49: TFastLineSeries;

Series50: TFastLineSeries;

Label1: TLabel;

GroupBox1: TGroupBox;

Label3: TLabel;

Edit1: TEdit;

Edit2: TEdit;

Panel1: TPanel;

Label4: TLabel;

Label5: TLabel;

Label6: TLabel;

Panel2: TPanel;

Label7: TLabel;

Label8: TLabel;

Label9: TLabel;

Edit3: TEdit;

Edit4: TEdit;

Chart2: TChart;

Series51: TLineSeries;

RadioButton1: TRadioButton;

RadioButton2: TRadioButton;

Image1: TImage;

Image2: TImage;

Panel3: TPanel;

Label10: TLabel;

Edit5: TEdit;

Edit6: TEdit;

Label11: TLabel;

Label12: TLabel;

RadioButton3: TRadioButton;

RadioButton4: TRadioButton;

Series52: TPointSeries;

Label2: TLabel;

Edit7: TEdit;

Label13: TLabel;

Edit8: TEdit;

Label14: TLabel;

Label15: TLabel;

Label16: TLabel;

procedure Button1Click(Sender: TObject);

procedure FormActivate(Sender: TObject);

procedure Image1Click(Sender: TObject);

procedure Image2Click(Sender: TObject);

procedure Edit1KeyPress(Sender: TObject; var Key: Char);

procedure Edit5KeyPress(Sender: TObject; var Key: Char);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form1: TForm1;

X: Fenotype;

i,j,n, dimDynamic : integer;

xMax: Fenotype; {массив максимальных значений для координат точки в пространстве поиска}

xMin: Fenotype; {массив минимальных значений для координат точки в пространстве поиска}

{ Три непересекающихся популяции - старая, новая и промежуточная }

OldPop, NewPop, IntPop: Population;

{ Глобальные целые переменные}

PopSize, Gen, h, s, b: integer;

{ Счетчики мутаций, скрещиваний и количество поколений }

NMutation, NCross, NGen: integer;

{ Статистические переменные }

Avg, Min, Max, SumFitness, xMinS, xMaxS, PMutation, PCross: double;

Search:string;

implementation

{$R \*.dfm}

function ObjFunc( x: Fenotype): real; { Функция }

begin

if dimDynamic = 1 then

ObjFunc := 5-24\*x[1]+17\*x[1]\*x[1]-(11/3)\*x[1]\*x[1]\*x[1]+(1/4)\*x[1]\*x[1]\*x[1]\*x[1]

else

ObjFunc := x[1]\*x[1]+x[2]\*x[2];

end;

{ Подбрасывание монетки - true если орел }

function Flip(Probability: double): boolean;

begin

Flip := Random <= Probability;

end;

{ Декодирование строки в массив вещественных координат }

procedure Decode(Chrom: Chromosome; var x: fenotype);

var

i, j, f, accum: longint;

begin

for i := 1 to dimDynamic do

begin

Accum := 0;

f := 1;

for j := 1 + LenChrome \* (i - 1) to LenChrome + LenChrome \* (i - 1) do

begin

if Chrom[j] then

Inc(Accum, f);

f := f \* 2;

end;

x[i] := xmin[i] + (xmax[i] - xmin[i]) \* Accum / (f - 1);

end;

end;

{ Расчет статистических величин }

procedure Statistics(var Max, Avg, Min: double; Pop: Population);

var

j: integer;

SumFitness: double;

begin

{Инициализация }

SumFitness := Pop[1].Fitness;

Min := Pop[1].Fitness;

Max := Pop[1].Fitness;

xMinS:= pop[1].x[1];

xMaxS:= pop[1].x[1];

{Цикл для max, min, sumfitness }

for j := 2 to PopSize do

with Pop[j] do

begin

{ Накопление суммы значений функции пригодности }

SumFitness := SumFitness + Fitness;

if Fitness > Max then begin

Max := Fitness; { Новое значение Max }

xMaxS := x[1];

end;

if Fitness < Min then begin

Min := Fitness; { Новое значение Min }

xMinS := x[1];

end;

end;

Avg := SumFitness / PopSize; { Расчет среднего }

end;

{ Инициализация начальной популяции случайным образом }

procedure InitPop;

var

i, j: integer;

begin

for i := 1 to PopSize do

with OldPop[i] do

begin

for j := 1 to LenChrome \* dimDynamic do Chrom[j] := Flip(0.5); { Бросок монеты }

Decode(Chrom, x); { Декодирование строки }

Fitness := ObjFunc(x);{ Вычисление начальных значений функции пригодности }

end;

end;

{3 генетических оператора: отбора (select), скрещивания (crossover) и мутации (mutation)}

{процедура турнирного отбора}

procedure Select(Search:string);

var

ipick, i: integer;

{ Процедура перемешивания популяции в процессе отбора }

procedure Shuffle(var pop: Population);

var

i, j: integer;

ind0: Individual;

begin

for i := 1 to PopSize do

begin

j := 1 + Random(i);

{ Перемешиваем }

ind0 := pop[i];

pop[i] := pop[j];

pop[j] := ind0;

end;

end;

{ Отбор наилучших особей для популяции для перехода в следующее поколение }

function Select1(Search:string): integer;

var

i, j, m: integer;

begin

if ipick > PopSize then

begin

Shuffle(OldPop);

ipick := 1;

end;

{берутся два соседних элемента текущей популяции }

i := ipick;

j := ipick + 1;

{и лучший из них помещается в промежуточную популяцию}

if Search='min' then

begin

if OldPop[j].Fitness < OldPop[i].Fitness then m := j else m := i;

end

else

begin

if OldPop[j].Fitness > OldPop[i].Fitness then m := j else m := i;

end;

Inc(ipick, 2);

Select1 := m;

end;

begin

ipick := 1;

for i := 1 to PopSize do

IntPop[i] := OldPop[Select1(Search)];

OldPop := IntPop;

end;

{ Оператор инверсионной мутации }

{мутация одного бита в строке (аллеля) с вероятностью pmutation}

{ alleleval – ген для мутации}

function Mutation(alleleval: Allele; var NMutation: integer): Allele;

{alleleval – ген для мутации;

PMutation – вероятность мутации;

NMutation – счетчик мутаций;}

begin

if Flip(PMutation) then { Мутация с вероятностью PMutation }

begin

Inc(NMutation); { Наращиваем счетчик мутаций }

Mutation := not alleleval; { Совершаем мутацию }

end else begin

Mutation := alleleval; { Не делаем мутацию }

end;

end;

{ Оператор одноточечного скрещивания }

{ Процедцра скрещивания 2 родительских строк, результат помещается в 2 строках-потомках}

procedure Crossover(var Parent1, Parent2, Child1, Child2: Chromosome; flchrom: integer; var NCross, NMutation: integer);

{parent1, parent2 – хромосомы родителей}

{child1,child2 – хромосомы потомков}

{flchrom – длина хромосомы (количество генов)}

{ncross, nmutation – счетчики количества скрещиваний и мутаций}

{jcross – точка сечения.}

var

i, jcross: integer;

begin

if Flip(PCross) then { Выполняется скрещивание с вероятностью PCross }

begin

{ Определение точки сечения в диапазоне между 1 и flchrom-1 }

jcross := 1 + Random(flchrom);

Inc(NCross); { Наращивание счетчика скрещиваний }

{ Первая часть обмена, 1 в 1 и 2 в 2 }

{ Обмениваем часть до точки сечения }

for i := 1 to jcross do

begin

{ Заодно и мутируем с вероятностью pmutation }

{ Первый потомок }

Child1[i] := Mutation(Parent1[i], NMutation);

{ Второй потомок }

Child2[i] := Mutation(Parent2[i], NMutation);

end;

{ Вторая часть обмена, 1 в 2 и 2 в 1 }

{ Обмениваем часть после точки сечения }

for i := jcross + 1 to flchrom do

begin

{ Заодно и мутируем с вероятностью pmutation }

{ Первый потомок }

Child1[i] := Mutation(Parent2[i], NMutation);

{ Второй потомок }

Child2[i] := Mutation(Parent1[i], NMutation);

end;

end;

end;

{ Генерирование нового поколения при помощи отбора, скрещивания и мутации }

procedure Generation(Search:string);

var

i: integer;

begin

Select(Search);

i := 1;

repeat

{ Выполняются отбор, скрещивание и мутация пока полностью не

сформируется новая популяция newpop }

{ Скрещивание и мутация - мутация вставлена в процедуру скрещивания }

Crossover(OldPop[i].Chrom, OldPop[i + 1].Chrom, NewPop[i].Chrom, NewPop[i + 1].Chrom, LenChrome \* dimDynamic, NCross, NMutation);

{ Декодирование строки и вычисление пригодности }

with NewPop[i] do

begin

Decode(Chrom, x);

Fitness := ObjFunc(x);

end;

with Newpop[i + 1] do

begin

Decode(Chrom, x);

Fitness := ObjFunc(x);

end;

Inc(i, 2);

until i > PopSize;

end;

//==============================================================================

procedure Pict(Chart1:TChart); // построение графика для двух переменных

var

i,j,M : integer;

begin

M:=Chart1.CountActiveSeries; // Узнаем, сколько у нас хранилищ данных

for i := 0 to M-1 do Chart1.Series[i].Clear; // и очищаем их

X[2]:=xMin[2];

j:=0;

while X[2]<=xMax[2] do

begin

X[1]:=xMin[1];

while X[1]<=xMax[1] do {при фиксированном X[2]

заполняем Series[j]}

begin

Chart1.Series[j].AddXY(X[1],ObjFunc(x));// значениями X[1] и ObjFunc(x)

X[1]:=X[1]+abs(xMin[1]-xMax[1])/200;

end; // while X[1]<=xMax[1]

j:=j+1; // переход к очередному Series[j]

X[2]:=X[2]+abs(xMin[2]-xMax[2])/(M-1); // и X[2]

end; // while X[2]<=xMax[2]

end;

//==============================================================================

procedure plotting(Chart1:TChart); {построение графика функции для одной переменной}

var i:fenotype;

h:Real;

begin

Chart1.Series[0].Clear;

h:= (xmax[1] - xmin[1])/100;

i[1]:=xmin[1];

while i[1]<=xmax[1] do begin

Chart1.Series[0].AddXY(i[1],objfunc(i));

i[1]:=i[1]+h;

end;

Chart1.Update;

end;

//==============================================================================

procedure TForm1.Button1Click(Sender: TObject);

procedure plottingDots(popsize:integer; var pop:population); {вывод на график каждого индивидуума }

var j:integer;

begin

Chart2.Series[1].Clear;

for j := 1 to popsize do with pop[j] do begin

if Search='min' then

begin

if Min=fitness then

begin

Chart2.Series[1].AddXY(xMinS,min,'',clBlue);

end

else

begin

Chart2.Series[1].AddXY(x[1],fitness);

end;

end

else

begin

if Max=fitness then

begin

Chart2.Series[1].AddXY(xMaxS,Max,'',clBlue);

end

else

begin

Chart2.Series[1].AddXY(x[1],fitness);

end;

end;

end;

Chart2.Update;

Sleep(300);

end;

var i,j:integer;

Result,BestResult: double;

begin

// валидация всех полей на пустоту

for i:=0 to form1.ComponentCount-1 do

begin

if form1.Components[i].ClassNameIs('TEdit') then

begin

if tedit(form1.Components[i]).Text='' then

begin

MessageDlg('Поле "' + tedit(form1.Components[i]).Hint + '" не должно быть пустым!',mtWarning,[mbOK],0);

exit;

end;

end;

end;

Randomize; { Инициализация генератора случайных чисел }

NGen := StrToInt(Edit6.Text); {Размер популяции }

PopSize := StrToInt(Edit5.Text);{ Количество поколений }

PMutation := StrToFloat(Edit7.Text)/100; { Вероятность мутации }

PCross := StrToFloat(Edit8.Text)/100;; { Вероятность скрещивания }

if (PopSize <= 0) or (PopSize >= 1000) then

begin

MessageDlg('Количество поколений должно больше 0 и меньше 1000',mtWarning,[mbOK],0);

exit;

end;

if NGen<=0 then

begin

MessageDlg('Размер популяции должен больше 0',mtWarning,[mbOK],0);

exit;

end;

if (PMutation < 0) or (PMutation > 100) then

begin

MessageDlg('Вероятность мутации должно больше 0 и меньше 100',mtWarning,[mbOK],0);

exit;

end;

if (PCross < 0) or (PCross > 100) then

begin

MessageDlg('Вероятность скрещивания должно больше 0 и меньше 100',mtWarning,[mbOK],0);

exit;

end;

if RadioButton3.Checked = true then Search:='min' else Search:='max';

Result := 0; { Инициализация переменной ответа }

if dimDynamic = 2 then

begin

Chart1.Visible:=true;

Chart2.Visible:=false;

xMax[1]:= StrToFloat(Edit2.Text);

xMax[2]:= StrToFloat(Edit4.Text);

xMin[1]:= StrToFloat(Edit1.Text);

xMin[2]:= StrToFloat(Edit3.Text);

if (xMin[1] > xMax[1]) or (xMin[2] > xMax[2]) then begin

MessageDlg('Интервал введен некорректно',mtWarning,[mbOK],0);

exit;

end;

Pict(Chart1);

end

else

begin

Chart1.Visible:=false;

Chart2.Visible:=true;

xMin[1]:= StrToFloat(Edit1.Text);

xMax[1]:= StrToFloat(Edit2.Text);

if (xMin[1] > xMax[1]) then begin

MessageDlg('Интервал введен некорректно',mtWarning,[mbOK],0);

exit;

end;

plotting(Chart2);

end;

Button1.Enabled:=false;

Button1.Update;

NMutation := 0; { Инициализация счетчика мутация }

NCross := 0; { Инициализация счетчика скрещиваний }

InitPop; { Создание начальной популяции }

Statistics(Max, Avg, Min, OldPop);

if dimDynamic=1 then plottingDots(PopSize,OldPop); {вывод на график каждого индивидуума популяции}

if Search='min' then begin

BestResult:= Min

end else begin

BestResult:= Max;

end;

Gen := 1; { Установка счетчика поколений в 0 }

repeat { Главный итерационный цикл }

Generation(Search);

Statistics(Max, Avg, Min, NewPop);

if Search='min' then begin

if Min < BestResult then BestResult := Min

end

else begin

if Max > BestResult then BestResult := Max;

end;

OldPop := NewPop;

if dimDynamic=1 then plottingDots(PopSize,OldPop); {вывод на график каждого индивидуума популяции}

{переход на новое поколение }

Inc(Gen);

until Gen > PopSize;

if Search='min' then

begin

Label1.Caption:='Минимум: '+FloatToStrF(BestResult,ffFixed,10,4);

end

else

begin

Label1.Caption:='Максимум: '+FloatToStrF(BestResult,ffFixed,10,4);

end;

Application.ProcessMessages;

Button1.Enabled:=true;

end;

//==============================================================================

procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);

begin

if not (Key in ['0'..'9', #8,'-'])then Key:=#0;

end;

procedure TForm1.Edit5KeyPress(Sender: TObject; var Key: Char);

begin

if not (Key in ['0'..'9', #8])then Key:=#0;

end;

procedure TForm1.FormActivate(Sender: TObject);

var

i,m : integer;

begin

dimDynamic:=2;

M:=Chart1.CountActiveSeries; // определяем число хранилищ данных Series

for i := 0 to M-1 do

Chart1.Series[i].Clear;

end;

procedure TForm1.Image1Click(Sender: TObject);

begin

RadioButton1.Checked:=true;

Panel2.Visible:=true;

Panel3.Top:=79;

dimDynamic:=2;

Edit1.Text:=IntToStr(-89);

Edit2.Text:=IntToStr(89);

end;

procedure TForm1.Image2Click(Sender: TObject);

begin

RadioButton2.Checked:=true;

Panel2.Visible:=false;

Panel3.Top:=41;

dimDynamic:=1;

Edit1.Text:=IntToStr(0);

Edit2.Text:=IntToStr(8);

end;

end.

1. Кроссинговер (кроссовер) – операция, при которой две хромосомы обмениваются своими частями. Например, 1100&1010 1110&1000 [↑](#footnote-ref-1)
2. Мутация — случайное изменение одной или нескольких позиций в хромосоме. Например, 1010011 - 1010001 [↑](#footnote-ref-2)
3. Популяция — совокупность индивидуумов. Индивидуум (генетический код, особь) — набор хромосом (вариант решения задачи). Обычно особь состоит из одной хромосомы, поэтому особь и хромосома идентичные понятия. [↑](#footnote-ref-3)
4. Хромосома — вектор (или строка) из каких-либо чисел [↑](#footnote-ref-4)
5. Пригодность (приспособленность) — критерий или функция, экстремум которой следует найти. [↑](#footnote-ref-5)