ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

|  |  |
| --- | --- |
| СОГЛАСОВАНО  заведующий отделением  Программной инженерии,  факультета Бизнес-информатики,  профессор кафедры УРПО  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Авдошин С.М.  «\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2013 г. | УТВЕРЖДАЮ  заведующий отделением  Программной инженерии,  факультета Бизнес-информатики,  профессор кафедры УРПО  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Авдошин С.М.  «\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2013 г. |

**программа оптимизации, Инспирированная поведением лягушек**

**Пояснительная записка**

**Лист утверждения**

**А.В.00001-01 ТЗ 01**

**RU.17701729.503200-01 81 01-1**

Инв. № подл.

Подпись и дата

Взам. инв. №

Инв. № дубл.

Подпись и дата

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  | | |
|  |  | | | \_ |
|  |  |  | | |
|  |  |  | | |
|  |  | Исполнитель: студент группы 171ПИ | | |
|  |  | \_\_\_\_\_\_\_\_\_\_\_\_\_/Ремнев Н.В./ | | |
|  |  | | “\_\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_2013 г. | |
|  |  |  | | |
|  |  |  | | |

**УТВЕРЖДЕНО**

RU.17701729.503200-01 81 01-1

**программа оптимизации, Инспирированная поведением лягушек**

**Пояснительная записка**

**RU.17701729.503200-01 81 01-1**

Инв. № подл.

Подпись и дата

Взам. инв. №

Инв. № дубл.

Подпись и дата

# Содержание

1. Введение 3
2. Назначение и область применения 4
   1. Назначение программы 4
   2. Область применения 4
3. Технические характеристики 5
   1. Постановка задачи на разработку программы 5
   2. Описание применяемых математических методов 5
   3. Допущения и ограничения в программе 6
   4. Описание алгоритма функционирования программы 6
   5. Описание метода организации входных и выходных данных 7
   6. Описание и обоснование выбора состава технических и программных

средств 7

1. Ожидаемые технико-экономические показатели 8
2. Источники, использованные при разработке 9

Приложение 1. Описание и функциональное назначение классов 10

Приложение 2. Описание и функциональное назначение методов, полей,

свойств 11

# ВВЕДЕНИЕ

«Программа оптимизации, инспирированная поведением лягушек» - программа, которая будет применяться студентами и учеными при изучении популяционных алгоритмов глобальной поисковой оптимизации и изучении ряда алгоритмов, вдохновленных живой природой.

Наименование программы: «Программа оптимизации, инспирированная поведением лягушек».

Имя запускаемого файла – Визуализация.exe.

Разработка ведется на основании задания на курсовую работу. Тема работы: «Программа оптимизации, инспирированная поведением лягушек». Национальный исследовательский университет – Высшая школа экономики, факультет Бизнес-информатики, отделение программной инженерии, кафедра управления разработкой программного обеспечения.

# НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ

* 1. **Назначение программы**

Программа ищет максимум и минимум заданной непрерывной функции от двух переменных на отрезке, с использованием алгоритма, инспирированного поведением лягушек, а также выполняет визуализацию поиска и построение графика функции от двух переменных.

* 1. **Область применения**

Программа будет использоваться при изучении популяционных алгоритмов, инспирированных живой природой.

# ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

* 1. **Постановка задачи на разработку программы**

Написать программу, которая будет выполнять поиск максимума и минимума функции от двух переменных на заданных пользователем промежутках, с помощью алгоритма, инспирированного поведением лягушек, а также визуализировать поиск экстремумов и строить график выбранной функции.

* 1. **Описание применяемых математических методов**

Основная цель разрабатываемой программы – реализация алгоритма, инспирированного поведением лягушек. Алгоритм был предложен Юсуфом (Eusuff M.) и соавторами в 2003 году. Данный алгоритм включает в себя следующие основные шаги:

1. инициализация популяции: размер популяции определяется случайным образом, координаты агентов популяции определяются также случайно в пределах заданных пользователем промежутках;
2. разделение агентов на мемеплексы (термин введен авторами алгоритма), каждый из которых содержит количество элементов, которое является делителем размера популяции. Агенты распределяются по мемеплексам согласно правилу соседства;
3. выполняется поиск лучшего агента во всей популяции;
4. для каждого из мемеплексов выполняется процедура меметической эволюции:

3.1. выполняется поиск лучшего и худшего агентов;

3.2. попытка улучшения позиции худшего агента по направлению к лучшему – если улучшение прошло успешно – фиксируем;

3.3. если последняя операция не привела к успеху – выполняется попытка улучшения позиции худшего агента по направлению к глобально лучшему агенту – если улучшение прошло успешно – фиксируем;

3.4. если предыдущие две операции не привели к успеху – взамен худшего агента случайным образом формируется новый агент, координаты которого лежат в заданных пользователем промежутках;

1. выполняется процедура тасования популяции – в результате чего, при повторном разделении популяции на мемеплексы, повторяться они не будут;
2. если выполнено условие поиска (в разрабатываемой программе – выполнение алгоритма заданное пользователем число раз) – выполнение алгоритма завершено, в противном случае, алгоритм повторяется, начиная со 2 шага (за исключением поиска количества элементов в мемеплексах – оно остается прежним).

Блок-схема данного алгоритма содержится в приложении 3 документа «Техническое задание».

Применительно к оптимизации функции данный алгоритм работает следующим образом:

1. формируется массив точек, границы которых лежат в заданных пользователем рамках;
2. для поиска максимума лучший агент определяется наибольшим значением фитнесс-функции, для поиска минимума – наименьшим;
3. в остальном алгоритм выполняется по отношению к оптимизации функции так как он есть: массив точек делится на мемеплексы, каждый из которых проходит процедуру меметической эволюции. Таким образом, в каждом мемеплексе точки с худшим значением фитнесс-функции «подтягиваются» к точкам с лучшим значением и постепенно весь массив точек «улучшается» в направлении к искомому минимуму/максимуму функции.
   1. **Допущения и ограничения в программе**

Допускается добавление функций в программу согласно приложению 1 документа «Руководство программиста».

В программе присутствует ряд ограничений. Работа программы допускает поиск максимума и минимума только у непрерывных функций от двух переменных заданных в программе. Количество итераций запуска алгоритма ограничено программой и находится в пределах от 1 до 1000. Интервалы, в которых лежат переменные X и Y ограничены 64-разрядным типом double с плавающей запятой, стандартным для языка программирования C#. Количество отображаемых знаков дробной части у чисел ограничено 15 знаками после запятой в окне действующей популяции и окне графика состояния популяции. Автозапуск выполнения итераций может производиться с разными интервалами между 1 и 5 секундами. Шкала поворота графика функции ограничена возможностями подключенной библиотеки, используемой для построения графика функции. В алгоритме, инспирированном поведением лягушек, количество агентов в популяции генерируется в пределах от 2 до 99 агентов.

* 1. **Описание алгоритма функционирования программы**

После ввода пользователем требуемых данных программа позволяет либо вывести результат, автоматически запустив указанное количество выполнений алгоритма пользователем, либо выполнить визуализацию алгоритма, запуская итерации последовательно. Более подробно с описанием функционирования программы можно ознакомиться в документе «Руководство оператора».

* 1. **Описание метода организации входных и выходных данных**

Для выполнения программы пользователю необходимо задать несколько

параметров:

1. поиск максимума либо минимума функции;
2. выбрать количество итераций запуска алгоритма;
3. задать границы интервалов по переменной X и по переменной Y;
4. выбрать функцию для анализа.

На выходе пользователь должен получить максимум или минимум

функции, найденный с помощью алгоритма, инспирированного поведением лягушек.

* 1. **Описание и обоснование выбора состава технических и программных средств**

Для выполнения программы требуется Microsoft .NET Framework 2.0. Не могут использоваться более ранние версии .NET, в связи с тем, что версия 2.0 в отличии от предыдущих, содержит DataGridView, ToolStrip и некоторые другие элементы WinForms, использованные при создании программы.

1. **ОЖИДАЕМЫЕ ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ**

Данная программа использует две сторонние библиотеки ZedGraph.dll и Chart3DLib.dll для построения графиков.

Библиотека ZedGraph.dll была использована для построения графика состояния популяции. Ее выбор обоснован удобством использования библиотеки и наличием стандартных базовых функций у графика, таких как масштабирование при прокручивании колеса мыши, сохранения графика как картинки и вывод графика на печать.

Библиотека Chart3DLib.dll была использована для построения графика функции. Ее выбор обоснован удобством использования библиотеки и наглядностью построенного графика.

Программа обладает следующими преимуществами:

1. построение графика исследуемой функции;
2. наглядность визуализации исполнения алгоритма, инспирированного поведением лягушек.
3. **ИСТОЧНИКИ, ИСПОЛЬЗОВАННЫЕ ПРИ РАЗРАБОТКЕ**

Список научно-исследовательских работ, использованных в разработке, содержится в Приложении 1 документа «Техническое задание».

Список литературы, использованной в разработке, содержится в Приложении 2 документа «Техническое задание».

Список ссылок на библиотеки, использованные в программе:

1. Jack Xu (2008) Download Example Code of the Book // Сайт drxudotnet.com. 10 апреля (<http://www.drxudotnet.com/csharp_download101a.html>)
2. Библиотека ZedGraph.dll // Сайт sourceforge.net. 2 апреля (<http://sourceforge.net/projects/zedgraph/files/>)

**ПРИЛОЖЕНИЕ 1**

**ОПИСАНИЕ И ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ КЛАССОВ**

|  |  |
| --- | --- |
| **Класс** | **Назначение** |
| Functions | Класс, отвечающий за представление функций в программе. |
| OptimizationAlgorithms | Статический класс, отвечающий за хранение методов для выполнения алгоритма. |
| FunctionGraph | Класс для представления формы, необходимой для построения графика функции. |
| Graphiks | Класс для представления формы, необходимой для построения графика состояния популяции. |
| MainForm | Класс для представления формы, необходимой для ввода пользователем данных необходимых для запуска алгоритма. |
| PopulationShow | Класс для представления формы, необходимой для вывода действующей популяции на данной итерации. |

**ПРИЛОЖЕНИЕ 2**

**ТАБЛИЦА С ОПИСАНИЕМ ЧЛЕНОВ ВСЕХ КЛАССОВ**

Класс *Functions*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Методы** | | | | |
| ***Имя*** | ***Мод. Доступа*** | ***Тип*** | ***Аргументы*** | ***Назначение*** |
| Function1 | public static | double | double,  double | Представление одной из функций |
| Function2 | public static | double | double,  double | Представление одной из функций |
| Function3 | public static | double | double,  double | Представление одной из функций |
| Function4 | public static | double | double,  double | Представление одной из функций |
| Function5 | public static | double | double,  double | Представление одной из функций |
| Function6 | public static | double | double,  double | Представление одной из функций |
| Function7 | public static | double | double,  double | Представление одной из функций |
| Function8 | public static | double | double,  double | Представление одной из функций |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Поля** | | | | |
| ***Имя*** | ***Мод. Доступа*** | ***Тип*** | ***Аргументы*** | ***Назначение*** |
| FunctionVyb | public delegate | double | double,  double | Делегат для представления выбранной функции |

Класс *OptimizationAlgorithms*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Методы** | | | | |
| ***Имя*** | ***Мод. Доступа*** | ***Тип*** | ***Аргументы*** | ***Назначение*** |
| NewPopulation | public static | double[,] | double,double  double,double | Формирование новой популяции |
| MemplexNumber | public static | int | int | Формирование количества эл-тов в мемеплексе |
| PopulationGood-Modernization | public static | double[,] | double[,],  double,double, double,double,  int,int,  Functions.FunctionVyb | Улучшение популяции для поиска максимума |
| PopulationBad-  Modernization | public static | double[,] | double[,],  double,double,double,double,  int,int,  Functions.FunctionVyb | Улучшение популяции для поиска минимума |
| Population-Shuffle | public static | double[,] | double[,] | Перемешивание популяции |
| Maximum | public static | double[] | double[,],  Functions.FunctionVyb | Поиск максимума функции в популяции |
| Minimum | public static | double[] | double[,],  Functions.FunctionVyb | Поиск минимума функции в популяции |
| FunctionMinimum | public static | double[] | int,  double,double, double,double,  Functions.  FunctionVyb | Полное выполнение всего алгоритма |
| FunctionMaximum | public static | double[] | int,  double,double, double,double,  Functions.  FunctionVyb | Полное выполнение всего алгоритма |

Класс *MainForm*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Методы** | | | | |
| ***Имя*** | ***Мод. Доступа*** | ***Тип*** | ***Аргументы*** | ***Назначение*** |
| MainForm() | public | конструктор | - | Создание формы и задание начальных знач. |
| MinX\_TextChanged | private | void | object,  EventArgs | Проверка корректности для мин. границы X |
| MaxX\_TextChanged | private | void | object,  EventArgs | Проверка корректности для макс. границы X |
| MinY\_TextChanged | private | void | object,  EventArgs | Проверка корректности для мин. границы Y |
| MaxY\_TextChanged | private | void | object,  EventArgs | Проверка корректности для макс. границы Y |
| MinX\_Leave | private | void | object,  EventArgs | Очищение ошибок для поля |
| MaxX\_Leave | private | void | object,  EventArgs | Очищение ошибок для поля |
| MinY\_Leave | private | void | object,  EventArgs | Очищение ошибок для поля |
| MaxY\_Leave | private | void | object,  EventArgs | Очищение ошибок для поля |
| GraphOpen\_Click | private | void | object,  EventArgs | Открытие окна графика |
| Function\_Click | private | void | object,  EventArgs | Выбор функции из доступных |
| Kol\_It\_TrackBar\_Scroll | private | void | object,  EventArgs | Изменения числа итераций через ползунок |
| Kol\_It\_Number\_  TextChanged | private | void | object,  EventArgs | Изменение числа итераций через текстовое поле |
| Result\_Click | private | void | object,  EventArgs | Вывод результата выполнения алгоритма |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Поля** | | | | |
| ***Имя*** | ***Мод. Доступа*** | ***Тип*** | ***Аргументы*** | ***Назначение*** |
| KolIt | private | int | - | Хранение количества итераций |
| MiX | private | double | - | Минимальная  граница Х |
| MiY | private | double | - | Минимальная  граница Y |
| MaX | private | double | - | Максимальная граница Х |
| MaY | private | double | - | Максимальная граница Y |
| newForm | public | Graphiks | int, double, double,  double,  double,  bool,  bool,  Functions.  FunctionVyb | Экземпляр открываемой формы |
| Fun | private | Functions.  FunctionVyb | double,  double | Хранение выбранной функции |
| clickitem | private | int | - | Счетчик для выбора функции |

Класс *PopulationShow*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Методы** | | | | |
| ***Имя*** | ***Мод. Доступа*** | ***Тип*** | ***Аргументы*** | ***Назначение*** |
| PopulationShow | public | конструктор | int,  double[,],  Functions.  FunctionVyb | Создание формы, задание нач. значений |
| KolZnakov\_ValueChanged | private | void | object,  EventArgs | Изменение количества знаков дробной части |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Поля** | | | | |
| ***Имя*** | ***Мод. Доступа*** | ***Тип*** | ***Аргументы*** | ***Назначение*** |
| newpop | private | double[,] | - | Хранение переданной в форму популяции |
| Fun | private | Functions.  FunctionVyb | double,  double | Хранение выбранной функции |

Класс *Graphiks*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Методы** | | | | |
| ***Имя*** | ***Мод. Доступа*** | ***Тип*** | ***Аргументы*** | ***Назначение*** |
| Postroenye | public | void | - | Построение осей графика |
| Setka | public | void | - | Построение сетки для графика |
| Points | public | void | double[,] | Вывод точек на график |
| ResultVyvod | public | void | - | Вывод результата при завершении итераций |
| ZnakVyvod | public | void | double[,] | Вывод максимума-минимума пред. итерации |
| ZnakVyvodNextIt | public | void | - | Вывод максимума-минимума, выполнение алгоритма |
| CopyLastMassiv | public | void | - | Копирование массива |
| Graphiks | public | конструк-тор | int, double,  double,  double,  double,  bool,  bool,  Functions.  FunctionVyb | Создание формы, задание первоначальных значенией |
| VypolnitAuto\_Click | private | void | object,  EventArgs | Автовыполнение итераций |
| nextiteration\_Click | private | void | object,  EventArgs | Выполнение одной итерации |
| Timer\_Tick | private | void | object,  EventArgs | Выполнение итерации, когда прошел временной интервал |
| Stop\_button\_Click | private | void | object,  EventArgs | Кнопка остановки автозапуска |
| TimeTrackBar\_Scroll | private | void | object,  EventArgs | Изменение временного интервала автозапуска |
| lastiteration\_Click | private | void | object,  EventArgs | Вывод предыдущей итерации на график |
| Population\_Click | private | void | object,  EventArgs | Открытие окна с отображением действующей итерации |
| KolZnakov\_Value  Changed | private | void | object,  EventArgs | Изменение количества знаков дробных чисел в агентах |
| NewPopulation\_Click | private | void | object,  EventArgs | Выполнение заново алгоритма – формирование новой итерации |
| zedGraph\_PointValue  Event | private | string | object,  EventArgs | Отображение координат при наведении курсора на точку |
| GraphFunOpen\_Click | private | void | object,  EventArgs | Открытие формы – отображение графика функции |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Поля** | | | | |
| ***Имя*** | ***Мод. Доступа*** | ***Тип*** | ***Аргументы*** | ***Назначение*** |
| Kol | private | int | - | Хранение количества итераций |
| MaxX | private | double | - | Максимальная граница по X |
| MinX | private | double | - | Минимальная граница по X |
| MinY | private | double | - | Минимальная граница по Y |
| MaxY | private | double | - | Максимальная граница по Y |
| Max | private | bool | - | Логическое значение – поиск максимума |
| Min | private | bool | - | Логическое значение – поиск минимума |
| newpop | private | double[,] | - | Хранение популяции |
| KolIterations | private | int | - | Количество итераций |
| lastmassiv | private | double[,] | - | Хранение массива предыдущей итерации |
| flag | private | bool | - | Логическое значение – проверка нажатия на кнопку предыдущей итерации |
| Isk | private | double[] | - | Массив для максимума |
| Isk2 | private | double[] | - | Массив для минимума |
| Fun | private | Functions.  FunctionVyb | double,  double | Хранение выбранной функции |
| graph | private | GraphPane | - | График для построения |
| list | private | PointPairList | - | Хранение точек для графика |
| newForm | private | FunctionGraph | Functions.  FunctionVyb  double, double,  double,  double | Открытие формы |

Класс *FunctionGraph*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Методы** | | | | |
| ***Имя*** | ***Мод. Доступа*** | ***Тип*** | ***Аргументы*** | ***Назначение*** |
| FunctionGraph | public | конструктор | Functions.  FunctionVyb,  double,  double,  double,  double | Создание формы, задание нач. значений |
| AddData | private | void | - | Отрисовка графика |
| PovorotElevation\_Scroll | private | void | object,  EventArgs | Изменение значения для верт. поворота |
| PovorotAzimuth  \_Scroll | private | void | object,  EventArgs | Изменение значения для гор. поворота |
| FunctionGraphik\_Resize | private | void | object,  EventArgs | Перерисовка графика при изменении размера |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Поля** | | | | |
| ***Имя*** | ***Мод. Доступа*** | ***Тип*** | ***Аргументы*** | ***Назначение*** |
| maxX | private | double | - | Максимальная граница по X |
| minX | private | double | - | Минимальная граница по X |
| minY | private | double | - | Минимальная граница по Y |
| maxY | private | double | - | Максимальная граница по Y |
| Fun | private | Functions.  FunctionVyb | double,  double | Хранение выбранной функции |
| pts | private | Point3[,] | - | Задание матрицы точек для графика |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Лист регистрации изменений | | | | | | | | | |
| Изм. | Номера листов | | | | Всего листов (страниц) в докум. | № докум. | Входящий №  сопроводительного докум. и дата | Подп. | Дата |
| изме-ненных | заме-ненных | новых | анну-лиро-ванных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |