**МИНОБРНАУКИ РОССИИ**
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**
**Кафедра МО ЭВМ**

# ОТЧЕТ

**по учебной практике**

**Тема: Генетические алгоритмы. Задача о рюкзаке**

Студент гр. 0303 _____ Афанасьев Д.В.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2022

**Цель работы.**

Разработать и реализовать программу, решающую задачу о рюкзаке с использованием генетических алгоритмов (ГА), а также визуализирующую работу алгоритма.

**Задание.**

Задача о рюкзаке. Требуется уложить как можно большее число ценных вещей в рюкзак при условии, что вместимость рюкзака ограничена.

Входные данные:

- Размер рюкзака;
- Количество предметов, их веса и стоимость.

В качестве языков программирования можно выбрать: C++, C#, Python, Java, Kotlin. Все параметры ГА необходимо определить самостоятельно исходя из выбранного варианта. Разрешается брать один и тот же вариант разным бригадам, но при условии, что будут использоваться разные языки программирования.

Требования к программе:

- Наличие графического интерфейса (GUI);
- Возможность ввода данных через GUI или файла;
- Пошаговая визуализация работы ГА. С возможностью перейти к концу алгоритма;
- Настройка параметров ГА через GUI. Например, размер популяции;
- Одновременное отображение особей популяции с выделением лучшей особи;
- Визуализация кроссинговера и мутаций в популяции;
- Наличие текстовых логов с пояснениями к ГА;
- Программа после выполнения не должна сразу закрываться, а должна давать возможность провести ГА заново, либо ввести другие данные.

**Описание представления данных**

Для представления хромосомы используется булевый вектор, в котором каждый элемент — это предмет (ген), который можно положить в рюкзак. Если в хромосоме на i-ой позиции стоит 1, то это значит, что i-ый предмет положили

в рюкзак, если 0, то нет. Стоит отметить, что при такой реализации хромосомы, не каждая хромосома будет принадлежать допустимому решению, то есть вес всех предметов в хромосоме может быть больше, чем вместимость рюкзака.

**Описание алгоритма**

Перед началом работы алгоритма происходит генерации n хромосом (n задается пользователем), удовлетворяющих допустимому решению.

Ключевая часть алгоритма состоит из набора последовательных действий, описанных ниже, которые повторяются до тех пор, пока в последних полученных потомствах максимальная приспособленность не перестанет меняться (количество потомств может задаваться пользователем) или количество потомств не достигнет количества, ограниченного пользователем.

Действия:

1. Определение приспособленности каждой хромосомы. Определяется при помощи фитнес-функции, которая вычисляется по формуле $f(\vec{x})=\sum_{i=1}^{n} c_i \cdot x_i$ , где $x_i$ — наличие i-го предмета в хромосоме, $x_i \in \{0,1\}$ , а $c_i$ — стоимость i-го предмета.
2. Турнирный отбор. При помощи отбора определяются особи для кроссинговера (у пользователя есть возможность определить количество особей участвующих в турнире). Отбор происходит по следующем правилу: отбирается t хромосом и из них выбирается хромосома с максимальной приспособленностью, после чего она добавляется в список для кроссинговера. Такая операция повторяется n раз. (возможно добавятся другие виды отборов).
3. Кроссинговер. После отбора особей происходит одноточечный кроссинговер между ними. Берутся две случайные особи, первая половина генов первой особи соединяется со второй поливной второй особи и наоборот. Таким образом, получаются две новые особи, которые будут входить в потомство.

4. После кроссинговера происходит проверка потомства на допустимость решения каждой особи. Если особь недопустима, то случайный не нулевой ее ген меняется на 0. Данная операция повторяется до тех пор, пока особь не станет допустимой.

5. Завершающим этапом является мутация потомства, которая с определенной вероятностью инвертирует один случайный ген в хромосоме. Вероятность мутации может задаваться пользователем.

После выполнения пятого действия получается итоговое потомство от текущей популяции.

**Описание взаимодействия пользователя с программой**

1) В начале пользователь попадает на главное меню (рис. 1), где ему предоставляется выбор способа ввода входных данных: из файла или при помощи GUI.

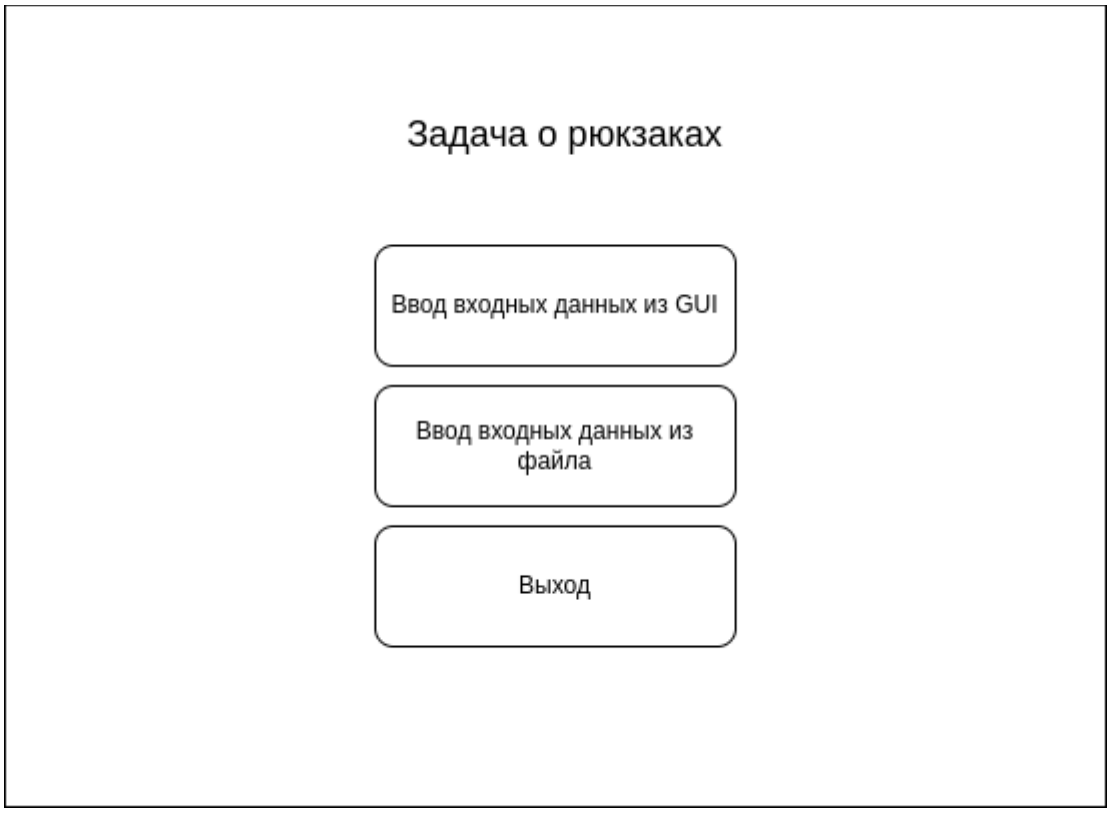


Рисунок 1 — Главное меню

2) Если пользователь выбрал ввод данных при помощи GUI, то перед ним откроется меню (рис. 2), где он сможет ввести вместимость рюкзака,

количество предметов, их вес и стоимость. Если он выбрал ввод входных данных из файла, то перед ним откроется меню (рис. 3), где он сможет ввести путь до нужного файла.

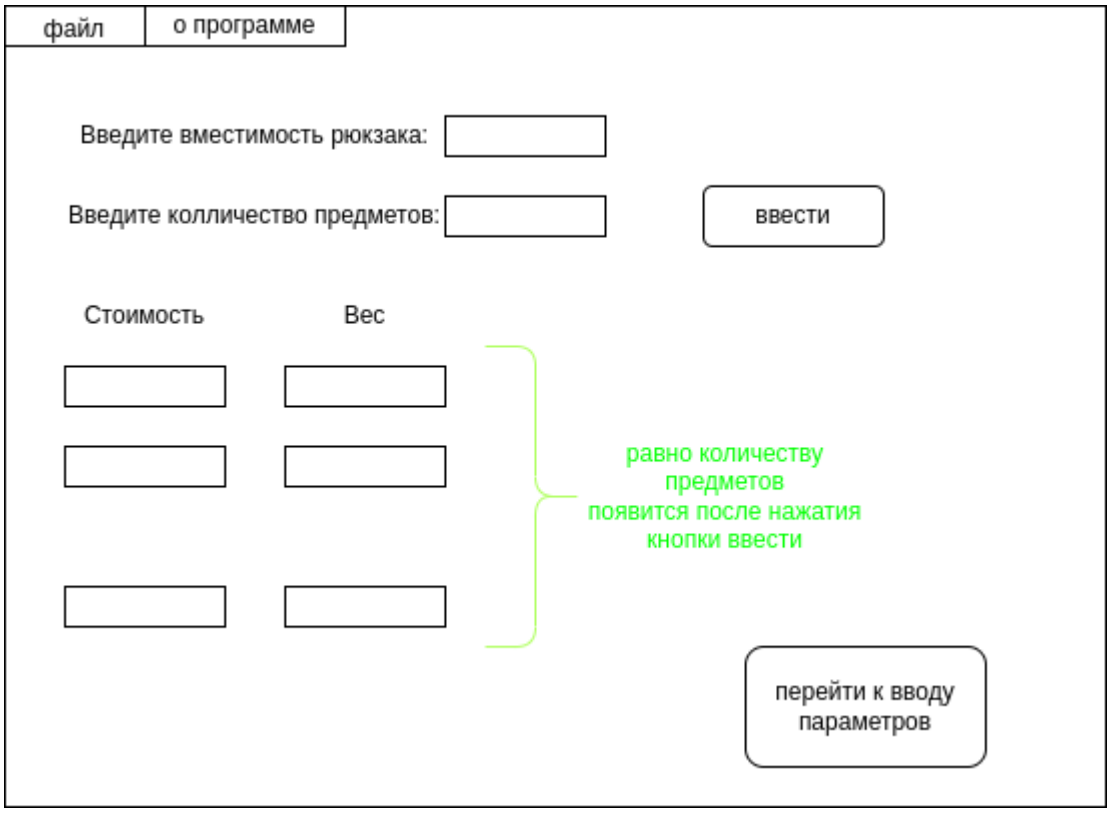


Рисунок 2 — Меню ввода данных при помощи GUI

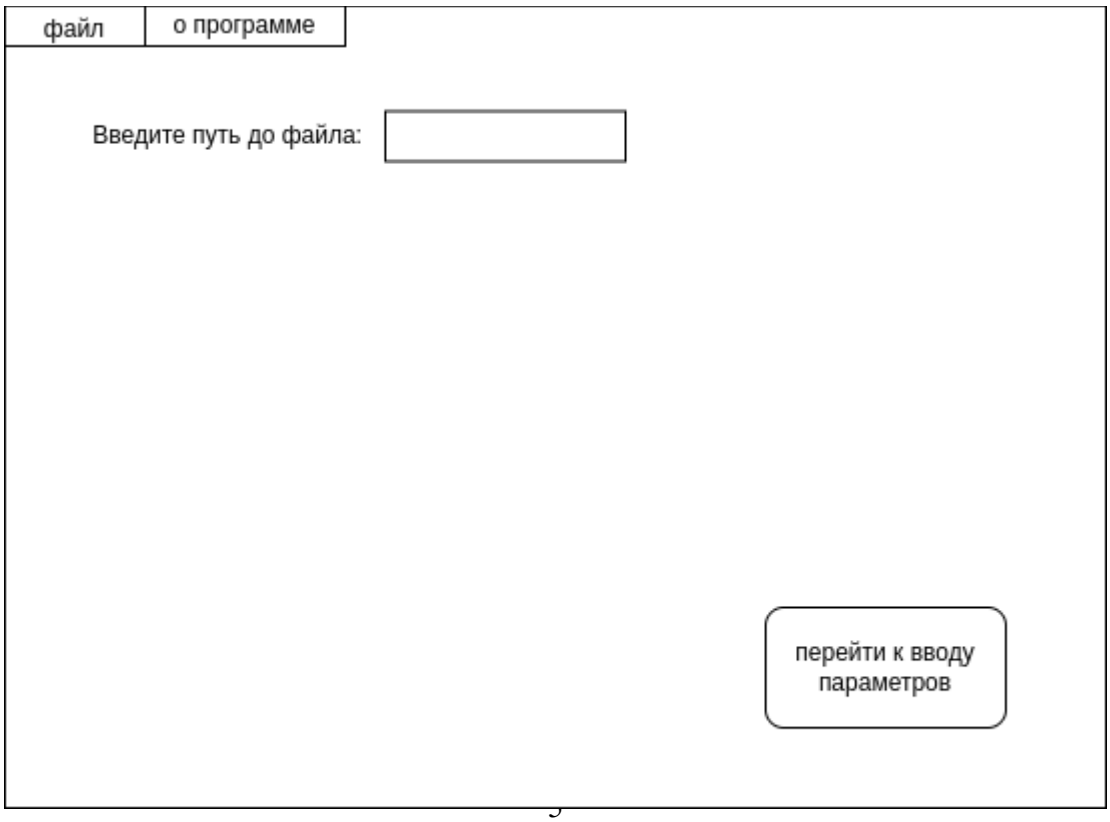

Рисунок 3 — Меню для указания пути до файла

3) После ввода входных данных пользователю предоставляется возможность выбрать параметры ГА (рис. 4). Список параметров: размер популяции, количество элементов для участия в турнирном отборе, вероятность мутации, условие для завершения работы алгоритма.

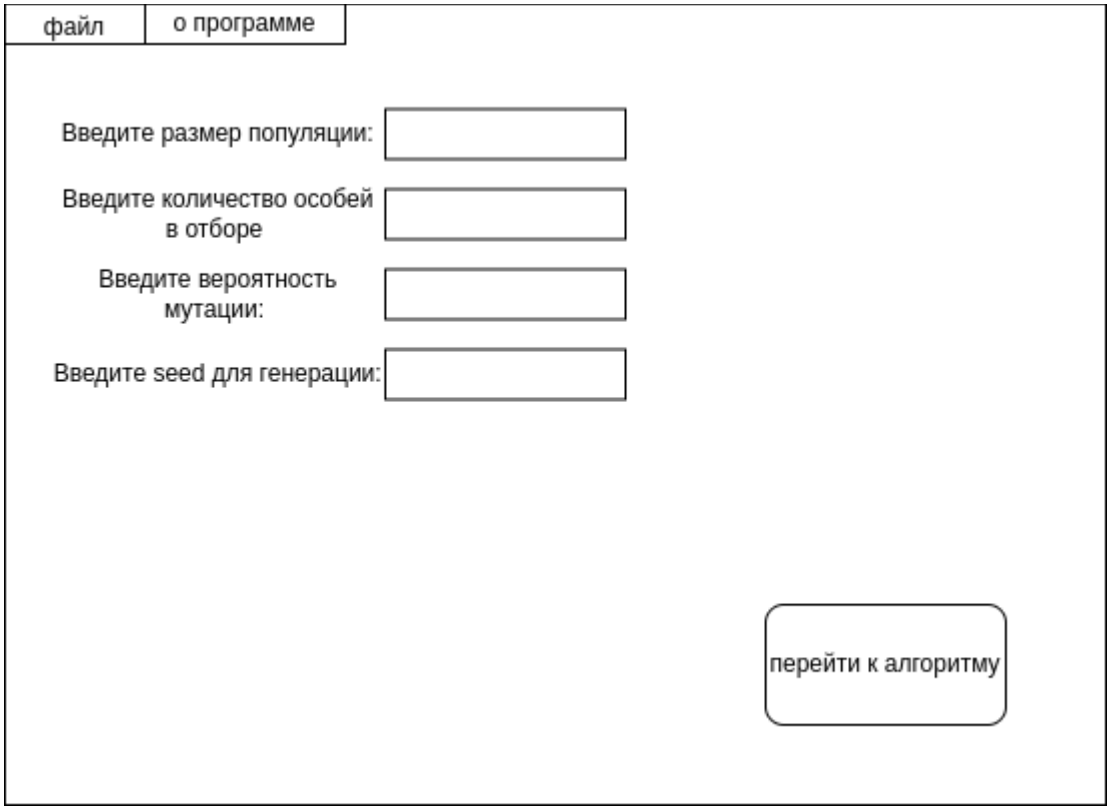


Рисунок 4 — Меню для указания параметров ГА

4) Когда указаны все параметры входных данных, происходит переход к окну выполнения алгоритма (рис. 5). У пользователя есть возможность перехода к следующему ходу или к итоговому решению (возможно будет возможность возвращаться к предыдущим шагам). Во время работы алгоритма пользователю также будут представлены текущие особи, их потомства, лучшие особи, а также информация о том, между какими особями происходит кроссинговер.

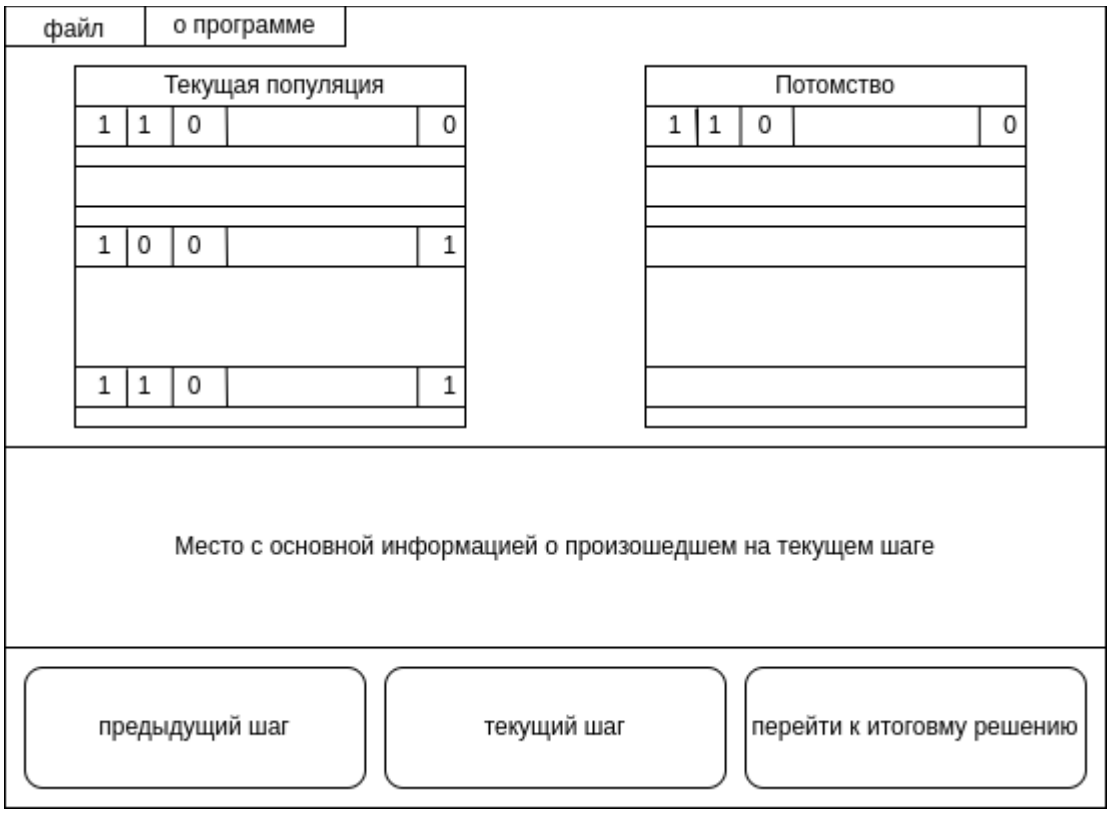


Рисунок 5 — Окно работы ГА

5) После заверения алгоритма пользователь будет перенаправлен на первоначальное меню.

**Основные классы, их поля и методы, для ралиазации ГА**

- *class DataGA* — класс предназначенный для хранения основных данных, которые задает пользователь.

  Поля:

  - std::vector<std::pair<unsigned int, unsigned int>> items — массив предметов, которые можно положить в рюкзак. Каждый предмет задан парой, где первое число — стоимость, второе — вес;
  - unsigned int sizePopulation — размер популяции;
  - unsigned int capacity — объем рюкзака;
  - unsigned int seed — число для генерации данных;
  - unsigned int numIndividualsSelection — число особей участвующих в отборе;

- float probabilityMutation — вероятность того, что одна особь мутирует, задается число от 0 до 1.

Методы — для каждого поля существует, соотвествущий метод get и set. Такой принцип был выбран для того, чтобы в класс, выполняющий ГА, не мог никак изменить входные данные, так как ему будет передан константный экзепляр класса DataGA.

- class Chromosome — класс, описывающий хромосому (особь).

Поля:

- const DataGA& data — набор параметров, введенных пользователем;
- std::vector<bool> genes — массив генов, где каждый ген — это предмет, который можно положить в рюкзак. 0 означает его остутсвие в рюкзаке, а 1 наличие.

Методы:

- unsigned int fitnessFunction() — фитнес-функция, показывает приспособленность особи;
- void correction() — метод для коррекции генов особи, так как особь может не принадлежать допустимому решению;
- void mutation() — метод, производящий мутацию особи с заданной вероятностью;
- bool isAcceptable() — метод проверяет принадлежит ли особь допустимому решению;
- std::vector<bool> getGenes() — метод, возвращающий массив генов особи.

- class GA — класс, описывающий основные действия работы ГА.
- Поля:
  - const DataGA& data — набор параметров, введенных пользователем;
  - std::vector<Chromosome> population — текущая популяция, на основе, которой будет получено потомство;
  - std::vector<Chromosome> child — потомство;

- std::vector<int> inSelection — особи, участвующие в кроссинговере;

- Методы:
  - void selection() — метод, производящий отбор особей для кроссинговера;
  - void mutation() — метод, производящий мутации потомство;
  - void crossing() — кроссинговер, между особями полученными при мощи метода *selection*;
  - void transfer() — метод для замены родителей текущим потомством;
  - void pick() — метод для отбора лучшей особи среди t случайных особей.

**GUI** (раздел будет изменяться, по скольку претерпевает значительные изменения )

В ходе работы был разработан интерфейс представлен на рис. 6-9.

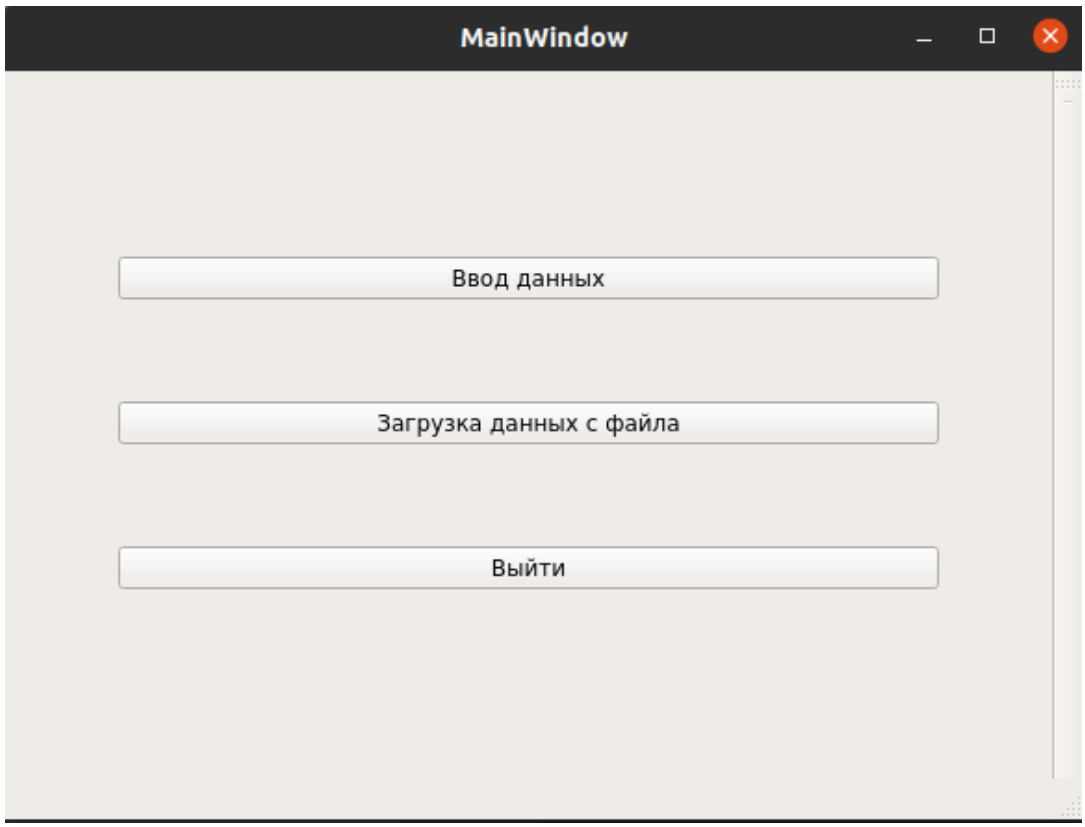


Рисунок 6 — Окно работы ГА

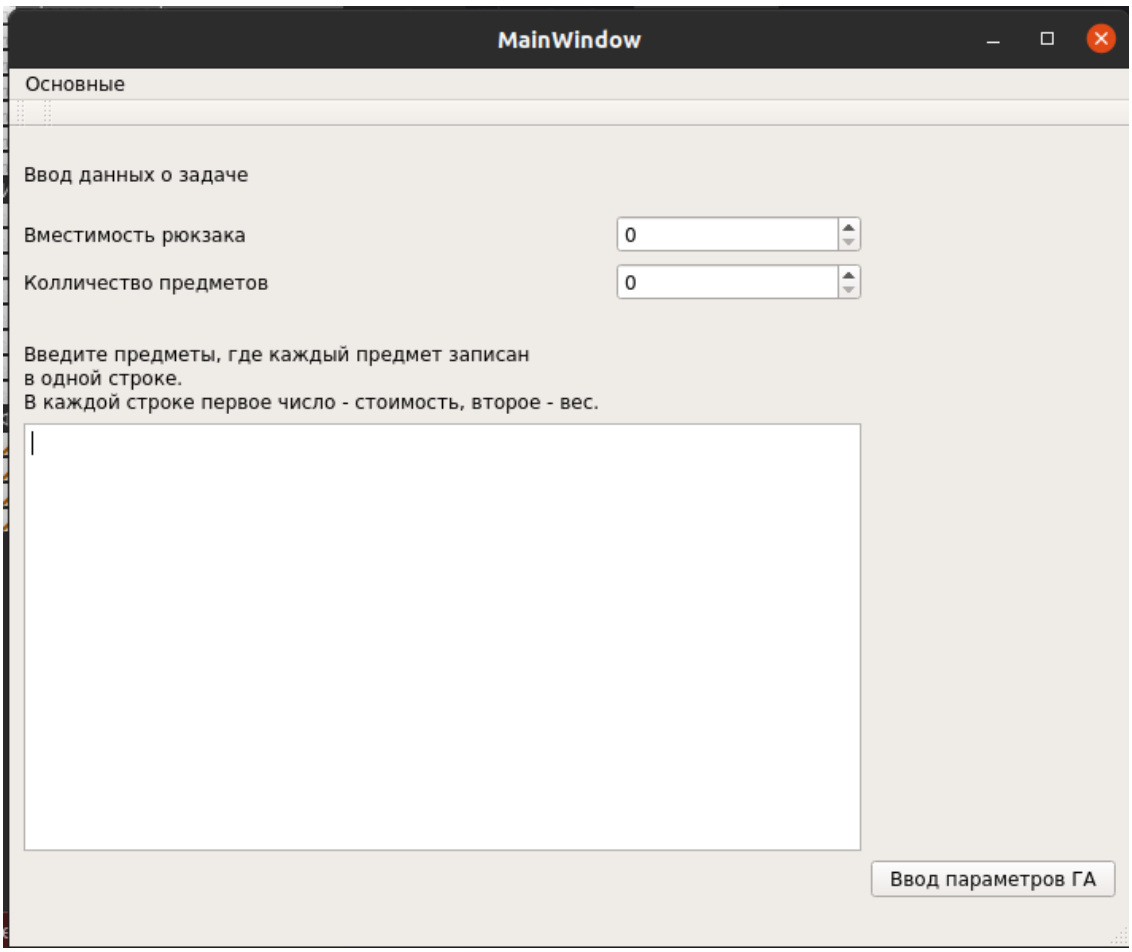


Рисунок 7 — Окно работы ГА


Рисунок 8 — Окно работы ГА

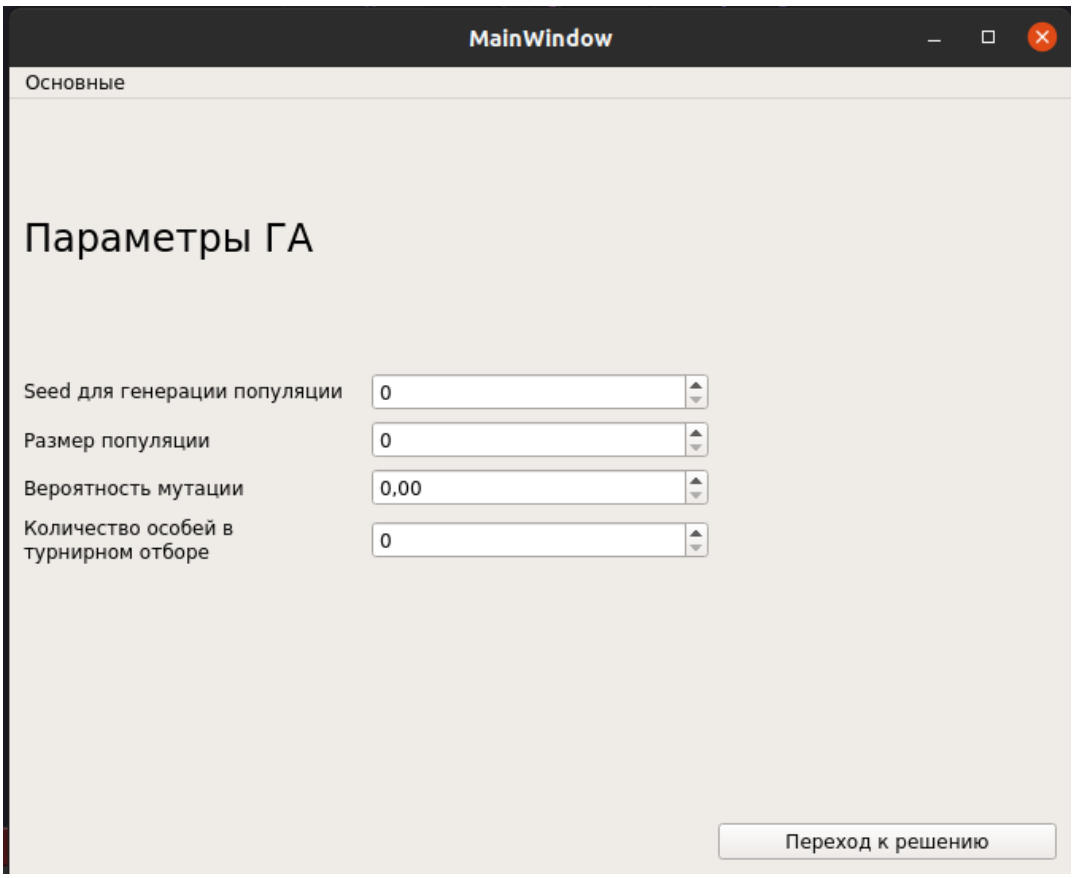


Рисунок 9 — Окно работы ГА

**Тестирование**

Будет добавлено в ходе последней итерации. Будет показана сходимость работы алгоритма.

**Выводы**

Будет добавлено в ходе последней итерации