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

ОТЧЕТ

по учебной практике

Тема: реализация генетического алгоритма в задаче на нахождение локального и глобального максимума функции

	Борисов И.В.,
	Жданов А.К.,
Авторы	Ильин П.О.

Санкт-Петербург 2024

Ход работы.

1) Генетический алгоритм

На второй итерации программа умеет находить только один глобальный максимум функции, если же был подан промежуток с несколькими максимумами.

Для реализации алгоритма было создано несколько основных классов, таких как Generetion, GeneticAlgorithm.

Класс Generation отвечает за создание и управление поколениями геномов. Он включает в себя параметры генерации, такие как начальный размер поколения, вероятности мутации и скрещивания. Также содержит указатели на объекты для селекции, скрещивания, мутации и оценки.

Методы:

- adaptability(genom_t genom): Возвращает приспособленность конкретного генома.
- generation number(): Возвращает номер текущего поколения.
- generation(): Возвращает текущее поколение.
- next_generation(): Генерирует следующее поколение.

Для реализации генерации следующего поколения были созданы интерфейсы (ISelector, ICrossover, IMutation, IEvaluator). Выбор такой реализации был сделан исходя из того, что пользователь сможет выбирать, как и с помощью каких методов будет происходить решение.

Класс GeneticAlgorithm отвечает за выполнение генетического алгоритма для поиска оптимальных решений задачи.

Метолы:

- run(): Выполняет генетический алгоритм.
- best_solution(std::size_t generation, std::size_t count = 3): Возвращает count лучших решений для указанного поколения.

Для хранения инициализации полинома был создан класс Polynomial, который хранит коэффициенты полинома, а также имеет несколько методов для удобства работы с ним:

Методы:

- operator()(T x): Вычисляет значение полинома в точке x.
- operator[](std::size_t index): Возвращает коэффициент полинома по индексу.
- degree(): Возвращает степень полинома.

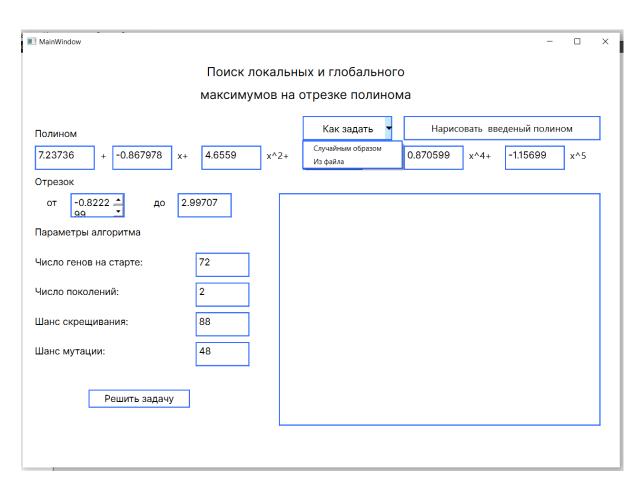
Селекция предков происходить с помощью метода рулетки, для этого был реализован класс RouletteWheel, который наследуется от ISelector. Во время этого алгоритма происходит расчет относительных процентов(исходя из их приспособленности) для каждого генома, после чего рассчитываются кумулятивные проценты для выбора некоторого набора геномов.

Поскольку для представления генома используется тип double, то для селекции был выбран метод скрещивание смешением(MixerCrossover), для этого берется два соседних генома, и из них генерируется новый ген из диапазона задаваемого родителем(mix). Вероятность скрещивания задается пользователем.

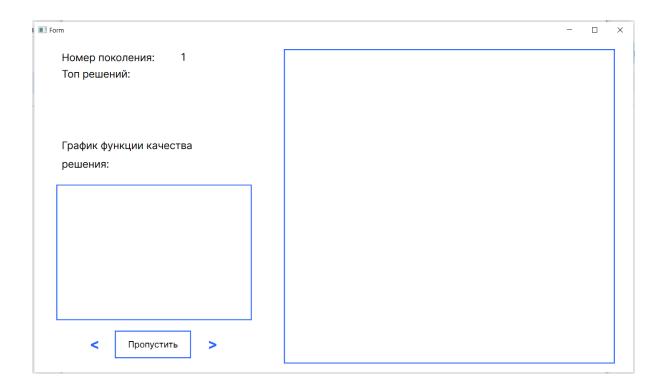
После скрещивания остается выполнить мутацию генов, для этого к гену прибавляется/вычитается некоторое случайное число, с проверкой на то, что приспособленность гена не вышла за границы изначально заданного отрезка. Для этого был создан класс SubstanceMutation, в котором с некоторой вероятностью, заданной пользователем происходит мутация каждого гена.

2) GUI

Реализован костяк интерфейса. Дизайн соблюден в соответствии с заготовленным наброском из предыдущего этапа работы с небольшими изменениями. Ввод полинома изменен на ввод коэффициентов полинома. Отрисовка графика на первом экране будет производится по нажатию кнопки (Отрисовка не реализована). Реализовано случайное заполнение формы для задачи. Данные с заполненной формы считываются и обрабатываются по нажатию кнопки решения задачи.



На втором окне изменение текущего рассматриваемого поколения ограничено логически возможными значениями (от 0 до заданного числа). Если пользователь добирается до последнего поколения, то автоматически срабатывает кнопка пропустить для перехода к конечным результатам.



На третьем окне добавлена кнопка для быстрого завершения текущего решения (закрываются третье и второе окно сразу).

