

Государственное образовательное учреждение высшего профессионального образования



**«Московский государственный технический университет
имени Н.Э. Баумана»**

(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Системы обработки информации и управления

Р А С Ч Ё Т Н О - П О Я С Н И Т Е Л Ь Н А Я З А П И С К А

к квалификационной работе бакалавра на тему:

Автоматизированная система обработки информации поиска алгоритмов
распознавания изоморфизма графов с помощью генетического
программирования

Студент Гуша Антон Валерьевич

(Подпись, дата)

(И.О.Фамилия)

Руководитель квалификационной работы

Филиппович Юрий Николаевич

(Подпись, дата)

(И.О.Фамилия)

1 Задание на выполнение квалификационной работы

2 Реферат

Объектом разработки является «АСОИ поиска алгоритмов распознавания изоморфизма графов методом генетического программирования».

Разработка АСОИ имеет своей целью осуществлять автоматический поиск эффективных алгоритмов проверки отношения изоморфизма среди всех возможных решений данной задачи.

Разработка системы проводилась на основании задания на выпускную работу, подписанного руководителем выпускной работы и утвержденного заведующим кафедрой ИУ МГТУ им. Н.Э. Баумана и документа «Техническое задание», утвержденное руководителем выпускной работы.

Данная расчетно-пояснительная записка является важной частью квалификационной работы бакалавра. Она в четкой и краткой форме раскрывает творческий замысел и его реализацию. В ней отражены этапы работы и результаты, полученные при выполнении проекта. Расчетно-пояснительная записка состоит из нескольких частей.

Конструкторская часть состоит из описания предметной области и общего описания программы.

Технологическая часть содержит описание разработки интерфейса и графа диалога.

В исследовательской части представлены основные исследовательские задачи, решенные в рамках выполнения работы.

Организационно-экономическая часть содержит смету затрат на разработку данного программного изделия и пояснения к смете.

Расчет эргономического сертификата приведен в части с соответствующим названием.

Содержание

1 Задание на выполнение квалификационной работы	2
2 Реферат	3
3 Введение	7
4 Конструкторская часть	8
4.1 Общетехническое обоснование разработки	8
4.1.1 Постановка задачи проектирования	8
4.1.2 Описание предметной области	9
4.1.3 Перечень процессов, подлежащих автоматизации	10
4.1.4 Выбор и обоснование критериев качества	11
4.1.5 Анализ аналогов и прототипов	13
4.2 Разработка программного изделия	17
4.2.1 Разработка структуры программного изделия	17
4.2.2 Описание модулей программного изделия	19
4.2.3 Особенности выбранных технологий	24
4.2.4 Язык программирования D	24
4.2.5 Язык разметки YAML	25
4.2.6 Библиотека графических элементов GTK+	26
4.2.7 Архитектура программы. UML-диаграмма классов	26
4.2.8 Выбор программных средств	33
4.2.9 Выбор аппаратных средств	33
4.2.10 Разработка основных алгоритмов обработки информации . .	34
4.2.11 Типизированная мутация	34
4.2.12 Типизированный кроссинговер	37
4.2.13 Генерация типизированного дерева	39
4.2.14 Получение новой популяции	42
4.2.15 Выбор случайного узла дерева	44
5 Технологическая часть	46
5.1 Разработка интерфейса взаимодействия с пользователем	46
5.2 Разработка форматов входных и выходных данных программы . .	48
5.2.1 Разработка форм входных данных	48

5.2.2	Разработка форм выходных данных	49
6	Исследовательская часть	52
6.1	Разработка проблемно-ориентированного языка	52
6.2	Представление генетического кода программ	53
6.2.1	Линия	54
6.2.2	Область	55
6.2.3	Оператор	55
6.2.4	Тип и Аргумент	58
6.3	Обеспечение тьюринг-полноты DSL	59
6.3.1	Выбор базисного языка	60
6.4	Описание DSL для решения задачи	62
7	Организационно-экономическая часть	67
7.1	Обоснование сметы затрат	67
7.1.1	Расчет затрат на расходные материалы	67
7.1.2	Расчет затрат на оборудование	67
7.1.3	Расчет затрат на услуги сторонних организаций	69
7.1.4	Расчет заработной платы	69
7.1.5	Расходы на дополнительную заработную плату	70
7.1.6	Расчет отчислений на социальные нужды	71
7.1.7	Расчет накладных расходов	72
7.1.8	Расчет прочих расходов	73
7.1.9	Расчет себестоимости	73
7.1.10	Расчет прибыли	73
7.1.11	Цена (без НДС)	74
7.1.12	Цена (с НДС)	74
7.2	Эргономика рабочего места и организация рабочего пространства .	74
7.2.1	Как будет использоваться ЭВМ	75
7.2.2	Помещение и освещение	76
7.2.3	Организация рабочего стола	77
7.2.4	Правильная высота	78
7.2.5	Эргономичная мебель	78
7.2.6	Вентиляция	79
7.2.7	Шум	80
7.2.8	Рабочее кресло	80

7.2.9	Как следует сидеть за ЭВМ	81
8	Заключение	84

3 Введение

Задача проверки изоморфизма графов является актуальной и исключительно привлекательной проблемой в наше время. Нахождение эффективного алгоритма, который за полиномиальное время позволит отвечать на данный вопрос, положительным образом повлияет на такие прикладные задачи как:

- Поиск химических соединений по базам данных в хемоинформатике и математической химии
- Верификация различных представлений электронной схемы в автоматизации проектирования электронных схем
- Выделение общих подвыражений в оптимизации программ
- Сопоставление графов знаний, содержащихся в семантических сетях

Уникальность данной задачи в том, что это одна из двух задач (и одна из 12, перечисленных в [?]), для которых класс сложности не был определен. Задача проверки изоморфизма графов принадлежит классу NP задач, но не доказано, что она является NP-полной задачей, и не найден алгоритм, решающий ее за полиномиальное время.

В 60-х – 80-х годах неоднократно предпринимались попытки решить данную задачу, но они не увенчались успехом. На данный момент лучший алгоритм имеет временную оценку сложности $2^{O(\sqrt{n \log(n)})}$ [?] [?].

В наши дни информационные технологии все больше используются как научные инструменты (яркий пример - решение проблемы четырех красок [?]). С ростом вычислительной мощности растет актуальность использовать автоматические методы поиска решения, например, метод генетического программирования. В данной работе разработан инструмент, спроектированный производить поиск решения задачи проверки отношения изоморфизма для ориентированных графов с выводом преобразованных в графическую форму промежуточных результатов для анализа человеком.

4 Конструкторская часть

4.1 Общетехническое обоснование разработки

4.1.1 Постановка задачи проектирования

Задача - разработать автоматизированную информационную систему, реализующую автоматический поиск алгоритмов проверки отношения изоморфизма ориентированных графов и предоставляющую графическую информацию о промежуточных результатах пользователю для анализа.

Задачи проектирования могут быть сформулированы следующим образом:

- а) Исследование предметной области проектирования
- б) Определение функциональных задач
- в) Изучение метода «Генетическое программирование»
- г) Разработка проблемно-ориентированного языка для внутреннего представления программ
- д) Выбор и обоснование критериев качества программы и оценки работы найденных алгоритмов
- е) Разработка схемы данных
- ж) Разработка алгоритмов программы
- з) Разработка программы
- и) Отладка программы
- к) Разработка графического интерфейса пользователя
- л) Тестирование программы
- м) Разработка конструкторской и эксплуатационной документации

4.1.2 Описание предметной области

Ориентированный граф - совокупность непустого множества вершин и множества связей между вершинами, называемыми ребрами. Ребра являются упорядоченными парами вершин.

$$G \equiv (E, V)$$

$$V \equiv \{(e_1, e_2) | e_1 \in E \wedge e_2 \in E\}$$

e_1 - **начало** ребра, e_2 - **конец** ребра. Далее ребро будет обозначаться следующим образом:

$$v = e_1 \rightarrow e_2$$

Далее рассматриваются только графы с конечным множеством вершин и ребер.

Граф G называется **изоморфным** графу H , если существует биекция f из множества вершин графа G в множество вершин графа H , обладающая следующим свойством: если в графе G есть ребро из вершины A в вершину B , то в графе должно быть ребро из вершины $f(A)$ в вершину $f(B)$ и наоборот — если в графе H есть ребро из вершины A в вершину B , то и в графе G должно быть ребро из вершины $f^{-1}(A)$ в вершину $f^{-1}(B)$. Биекция также должна сохранять ориентацию ориентированного графа.

Пример изоморфных графов представлен на рисунке 1.

Матрица смежности графа G с конечным числом вершин n - это квадратная матрица A размера n , в которой значение элемента a_{ij} равно числу ребер из i -й вершины графа в j -ю вершину.

Самый прямолинейный способ установить отношение изоморфности между графиками G и H - перестановками строк и столбцов матрицы смежности графа G получить матрицу смежности графа H . Однако перебор всех возможных перестановок характеризуется вычислительной сложностью $O(N!)$, что практически

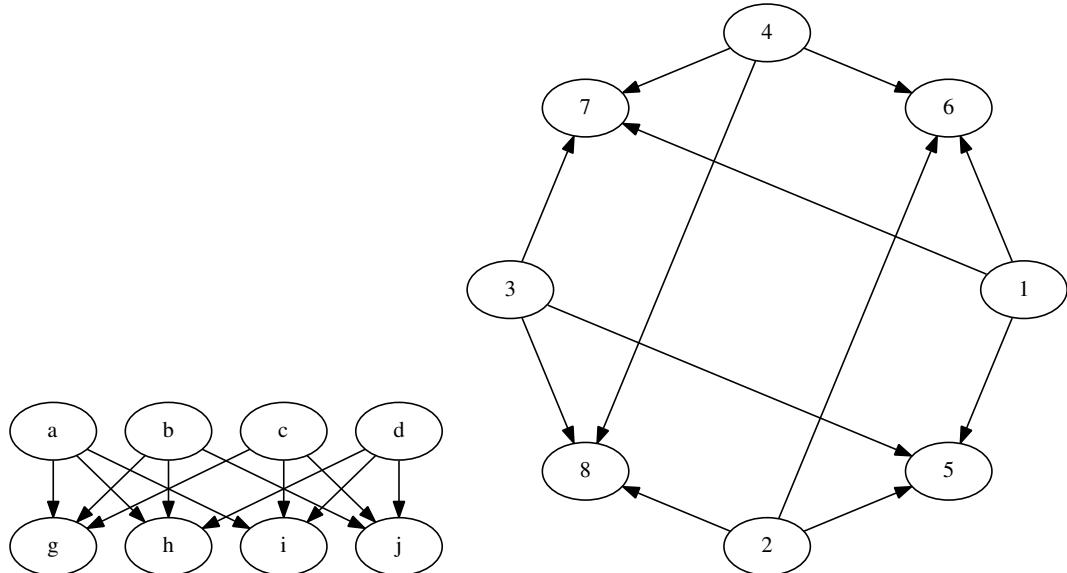


Рисунок 1 – Пример изоморфных графов

исключает применение подобного подхода на практике.

Существует набор числовых характеристик графов, называемыми полными инвариантами[?], совпадение которых у различных графов является необходимым и достаточным условием изоморфизма. Примерами таких инвариантов являются:

- Мини-код $\mu_{min}(G)$ матрицы смежности, получаемый путем выписывания двоичных значений матрицы смежности в строчку с последующим переводом полученного двоичного числа в десятичную форму. Мини-коду соответствует такой порядок следования строк и столбцов, при котором полученное значение является минимально возможным.
- Макси-код $\mu_{max}(G)$ матрицы смежности, получаемый путем выписывания двоичных значений матрицы смежности в строчку с последующим переводом полученного двоичного числа в десятичную форму. Макси-коду соответствует такой порядок следования строк и столбцов, при котором полученное значение является максимально возможным.

В настоящее время полный инвариант графа, вычислимый за полиномиальное время, неизвестен, однако не доказано, что он не существует. Попытки его отыскания неоднократно предпринимались в 60-х — 80-х годах XX века, однако не увенчались успехом. На данный момент лучший алгоритм определения

изоморфизма графов имеет временную оценку сложности $2^{O(\sqrt{n \log(n)})}$ [?] [?].

4.1.3 Перечень процессов, подлежащих автоматизации

Процессы, которые подлежат автоматизации:

- а) Генерация первичных алгоритмов проверки изоморфизма графов
- б) Эволюционный поиск и отбор наилучших алгоритм проверки изоморфизма графов
- в) Качественная оценка полученных алгоритмов проверки изоморфизма графов
- г) Визуализация полученных алгоритмов проверки изоморфизма графов

4.1.4 Выбор и обоснование критериев качества

Для проектируемой автоматизированной информационной системы приоритетными являются следующие критерии качества:

- скорость обработки эволюционного процесса
- используемая память во время обработки эволюционного процесса
- лицензия на исходный код и исполняемые файлы
- стоимость приобретения или использования программного продукта
- используемый язык программирования
- используемый подход для представления исходного кода алгоритмов
- удобство работы и документация

Скорость обработки эволюционного процесса - однозначно определяется временем, затрачиваемым на обработку одного поколения индивидов при одинаковом количестве запусков на каждого индивида и настройках эволюционных операторов. Данный параметр является самым важным, он определяет насколько быстрее программный продукт найдет качественные решения.

Используемая память во время обработки эволюционного процесса - определяется эффективностью использования адресного пространства ОЗУ вычислительного средства. Меньшее потребление памяти позволяет запускать большее число параллельных популяций для поиска алгоритмов. Является менее важным параметром, так как при плохой скорости обработки преимущество в используемой памяти несущественно.

Лицензия на исходный код и исполняемые файлы - лицензии регулируют правила пользования исходными кодами программы и исполняемыми файлами. Преимущество дают широко распространенные **свободные** лицензии, которые позволяют просматривать, модифицировать и распространять исходные коды, а к исполняемым файлам прикрепляется инструкция получения исходных кодов. Закрытые (проприетарные) лицензии зачастую используются в платном программном обеспечении, что резко ограничивает его применимость.

Стоймость приобретения или использования программного продукта - многие программные продукты спроектированы для профессионального и/или коммерческого использования, и разработчики данных программ требуют оплату при получении или использовании их программ.

Используемый язык программирования - необходимо разработать АСОИ на языке программирования **D**, который является на данный момент современным языком системного и прикладного программирования, ориентированный на создание эффективных понятных и крупных программных комплексов. Пополнение и использование экосистемы языка **D** является значительным вкладом в развитие области. Для использования других языков с программами, написанными на **D**, необходимо наличие С-API или специальной обертки, которая пишется программистом под собственные нужды, что означает дополнительные

Таблица 1 – Принятые значения весовых коэффициентов

Коэффициент	Значение	Описание
K_1	0.2	Скорость обработки
K_2	0.1	Используемая память
K_3	0.15	Лицензия
K_4	0.15	Стоимость
K_5	0.3	Язык программирования
K_6	0.05	Представление алгоритма
K_7	0.05	Удобство работы и документация

расходы. Поэтому системы, написанные на других языках и которые имеют С-API рассматриваются как более удобные, чем без данного интерфейса взаимодействия.

Используемый подход для представления исходного кода алгоритмов - различные системы генетического программирования используют различные способы представления исходного кода алгоритмов. Для решения рассматриваемой задачи древовидный подход оценивается как самый удобный, так как в процессе эволюции создаются целые алгоритмы, что отличает генетическое программирование от генетических алгоритмов и похожих методов.

Удобство работы и документация - для работы с автоматизированными информационными системами, являющимися научными инструментами, необходимы общирная документация и удобство работы, чтобы обеспечить наискорейшее получение результатов.

Итоговые значения весовых коэффициентов для метода базового критерия представлены в таблице 1. Необходимое для метода базового критерия нормировки соблюдено:

$$\sum_i K_i = 1$$

4.1.5 Анализ аналогов и прототипов

Генетическое программирование было заложено Koza, J.R. в [?] в 1992 году. С того времени появилось немало реализаций данного метода, но для сравнения были выбраны самые используемые и широко распространенные.

JGAP - библиотека, написанная на языке Java, для генетического программирования и генетических алгоритмов. Она предоставляет базовые генетические механизмы, которые могут быть легко использованы в эволюционном подходе для решений задач. JGAP спроектирована простой для использования "из коробки при этом являясь гибкой системой для того, чтобы пользователи могли легко добавлять свои генетические операторы и другие компоненты.

Документация, качество и стабильность кода являлись главными критериями при проектировании JGAP. Исходный код содержит множество тестов, документирующие комментарии и множество примеров.

Преимущества:

- Отличная документация и удобство пользования
- Открытая лицензия
- Бесплатность
- Использование необходимой модели внутреннего языка

Недостатки:

- Большое использование памяти (особенность языка)
- Использование языка, требующего трудоемкой разработки обертки

ECJ - исследовательская система, написанная на языке Java. Имеет крайне гибкую структуру, где все классы загружаются во время исполнения по заданным пользователем конфигурационным файлам. Также данная система разработана с учетом высокой производительности вычислений.

Преимущества:

- Высокая скорость обработки
- Открытая лицензия
- Бесплатность
- Использование необходимой модели внутреннего языка

Недостатки:

- Большое использование памяти (особенность языка)
- Использование языка, требующего трудоемкой разработки обертки
- Недостаточный объем документации и примеров

ECF - библиотека, написанная на языке C++, предназначенная для моделирования любого вида эволюционных вычислений. Включает в себя широкий набор методов эволюционных оптимизаций, многопоточное исполнение, хорошо конфигурируемый процесс эволюции.

Преимущества:

- Высокая скорость обработки
- Открытая лицензия
- Бесплатность

Недостатки:

- Использование языка, требующего разработки обертки
- Недостаточный объем документации и примеров
- Используется неудобный способ представления алгоритмов

Discipulus - коммерческая система генетического программирования. Предположительно написана на C#, заявляется удобство представления входных данных, интеллектуальная система настройки эволюционного процесса и встроенная защита от проблемы, называемой «переобучение».

Преимущества:

- Имеет C-API, не нужна разработка обертки
- Заявляется удобство использования, отсутствующее у конкурентов

Недостатки:

- Платность
- Проприетарная лицензия
- Недостаточный объем документации и примеров
- Мало информации о системе

Самостоятельно разработанная АСОИ - использован языке программирования **D**, используется гибкая система, позволяющая заменять любой компонент системы и добавлять свои генетические операции. Предусмотрено множество параметров системы, которые доступны для регулирования пользователем. Система разработана с учетом необходимости высокой производительности и низкого потребления памяти.

Преимущества:

- Высокая скорость обработки
- Низкое потребление памяти
- Написана на целевом языке, не требует оберток
- Открытая лицензия
- Бесплатность
- Большой объем документации различных видов

Таблица 2 – Таблица перевода качественных параметров в количественные

Качественное значение	Отлично	Хорошо	Удовлетворительно	Плохо
Количественное значение	1	0.8	0.5	0.2

Таблица 3 – Сравнение аналогов без перевода в количественные значения

Критерий	JGAP	ECJ	ECF	Discipulus	Данная АСОИ
K_1	Отлично	Хорошо	Отлично	Хорошо	Хорошо
K_2	Удовл.	Удовл.	Хорошо	Хорошо	Хорошо
K_3	Отлично	Отлично	Отлично	Удовл.	Отлично
K_4	Отлично	Отлично	Отлично	Плохо	Отлично
K_5	Удовл.	Удовл.	Хорошо	Хорошо	Отлично
K_6	Отлично	Отлично	Хорошо	Хорошо	Отлично
K_7	Отлично	Удовл.	Удовл.	Плохо	Хорошо

Недостатки:

- Проблемы со стабильность из-за малого срока разработки
- Узкая заточенность под конкретную задачу

Перевод качественных параметров в количественные производится при помощи таблицы 2.

Сперва сравним аналоги без учета весовых коэффициентов, результат сравнения представлен в таблице 3. Приведенные и нормированные значения количественных критериев с учетом весовых коэффициентов представлены в таблице 4.

Из произведенных расчетов видно, что разработка собственной информационной автоматизированной системы является целесообразным, так как она со-

Таблица 4 – Сравнение аналогов и прототипов с учетом нормировки и весовых коэффициентов

Критерий	α	JGAP	ECJ	ECF	Discipulus	Данная АСОИ
K_1	0.2	1.0	0.8	1.0	0.8	0.8
K_2	0.1	0.5	0.5	0.8	0.8	0.8
K_3	0.15	1.0	1.0	1.0	0.5	1.0
K_4	0.15	1.0	1.0	1.0	0.2	1.0
K_5	0.3	0.5	0.5	0.8	0.8	1.0
K_6	0.05	1.0	1.0	0.8	0.8	1.0
K_7	0.05	1.0	0.5	0.5	0.2	0.8
$\sum_i \alpha_i K_i$	1.0	0.8	0.735	0.885	0.635	0.93

ответствует поставленным целям лучше, чем существующие аналоги.

4.2 Разработка программного изделия

4.2.1 Разработка структуры программного изделия

АСОИ состоит из нескольких подсистем и разбита на две большие части:

- Внутренняя часть, которая разрабатывалась специально под решаемую задачу в рамках данной работы. Модули, относящиеся к этой части, будем называть **внутренние**.
- Внешняя часть - модули, относящиеся к библиотеке генетического программирования **Devol**, разработанной ранее как многофункциональное инструментальное средство для генетического программирования. Изначально **Devol** являлся коллективной разработкой (автор является одним из основных разработчиков), и в рамках данной работы эта библиотека была переработана и дополнена, чтобы отвечать поставленным целям. Данные модули будем называть **внешние**.

Графическая схема модулей АСОИ представлена на листе 2 графической части и на рис. 2.

4.2.2 Описание модулей программного изделия

Внутренние модули АСОИ:

- **main** - главный модуль системы, точка входа приложения. В нем инициализируются объект приложения, все окна и логгер
- **application** - описание класса приложения, хранит ссылки на все окна, логгер, проект и управляет механизмом завершения работы приложения

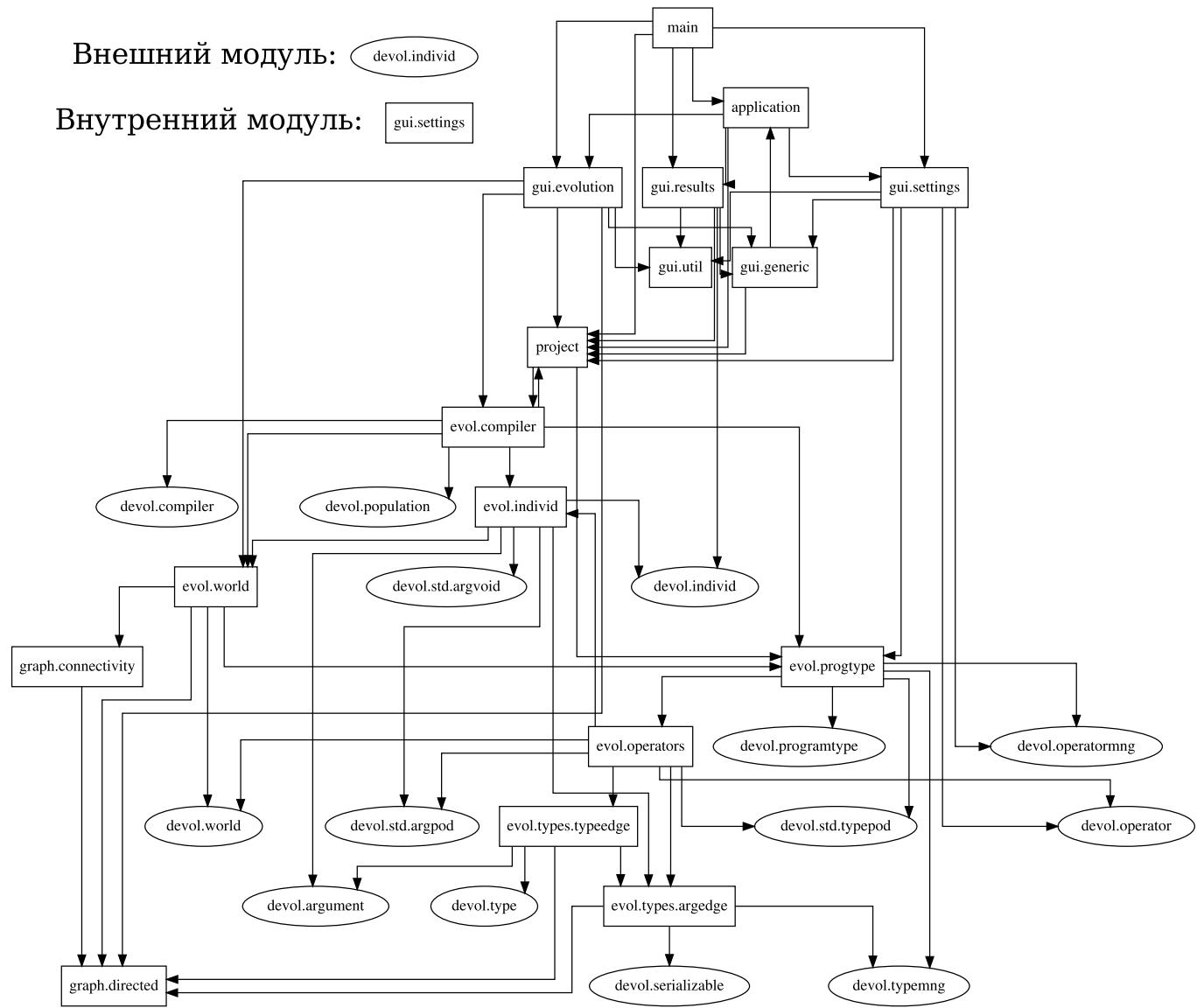


Рисунок 2 – Схема модулей АСОИ

- **project** - описание сохраняемых параметров АСОИ. Сохранению подлежат: параметры эволюции, имя популяции и популяция
- **gui** - пакет, который содержит подпакеты, относящиеся к графическому интерфейсу пользователя:
 - **evolution** - описание окна эволюции
 - **generic** - описание общего поведения для всех окон, включая главное меню
- **results** - описание окна результатов
- **settings** - описание окна настроек эволюции и просмотра проблемно - ориентированного языка

- **util** - содержит алгоритм переключения между окнами
- **graph** - пакет, содержащий подпакеты, относящиеся к реализации ориентированных графов:
 - **connectivity** - реализация ориентированного графа на списках связности
 - **directed** - описание интерфейса ориентированного графа
- **evol** - пакет, содержащий подпакеты, относящиеся к эволюционным процессам:
 - **compiler** - описание стратегии компиляции, параметризация популяции и интерпретатора проблемно-ориентированного языка
 - **individ** - описание алгоритма распознавания изоморфизма графов как индивида в популяции
 - **proctype** - описание настроек эволюционного процесса, также загружает все операторы и типы проблемно-ориентированного языка
 - **world** - описание среды, в которой выполняются программы-индивидуы. Описание функции приспособленности и генерации входных графов
 - **operators** - пакет, содержащий подпакеты, относящиеся к операторам проблемно-ориентированного языка:
 - **and** - описание оператора логического "И"
 - **answer** - описание оператора записи ответа на поставленную задачу
 - **construct** - описание создания аргумента, описывающего ребро графа

- **dist** - описание получения индекса конца ребра графа
- **div** - описание арифметического оператора деления
- **gdup** - описание операции над стеком общего назначения: дублирование вершины стека
- **gover** - описание операции над стеком общего назначения: копирование аргумента под вершиной стека и расположение копии на вершине
- **gpop** - описание операции над стеком общего назначения: снятие значения с вершины стека
- **gpush** - описание операции над стеком общего назначения: сохранение аргумента в стеке
- **grot** - описание операции над стеком общего назначения: перемещение третьего с вершины аргумента в стеке на вершину
- **gswap** - описание операции над стеком общего назначения: перемещение вершины стека под следующий за ней аргумент
- **idcast** - описание оператора преобразования целочисленной переменной в действительную
- **idup** - описание операции над входными стеками: дублирование вершины стека
- **iover** - описание операции над входными стеками: копирование аргумента под вершиной стека и расположение копии на вершине
- **ipop** - описание операции над входными стеками: снятие значения с вершины стека

- **ipush** - описание операции над входными стеками: сохранение аргумента в стеке
- **irot** - описание операции над входными стеками: перемещение третьего с вершины аргумента в стеке на вершину
- **iswap** - описание операции над входными стеками: перемещение вершины стека под следующий за ней аргумент
- **mult** - описание арифметического оператора умножения
- **not** - описание оператора логического "НЕ"
- **opif** - описание оператора логического ветвления
- **opwhile** - описание оператора цикла
- **or** - описание оператора логического "ИЛИ"
- **plus** - описание оператора арифметического сложения
- **relation** - описание операторов сравнения: больше, меньше, больше равно, меньше равно
- **round** - описание оператора округления, преобразования действительного аргумента к целочисленному
- **source** - описание получения индекса начала ребра графа

- **types** - пакет, содержащий подпакеты, относящиеся к описанию типов проблемно-ориентированного языка:
- **argedge** - описание аргумента, содержащего ребро графа

- **typedge** - описание типа ребра графа

Внешние модули АСОИ, используемые во внутренних модулях:

- **devol.compiler** - описание общих процедур интерпретации программ-индивидуов
- **devol.population** - описание популяции-контейнера, параметризируемого пользовательским типом индивидов
- **devol.individ** - описание базовой абстракции индивида-программы, перегружаемой пользователем
- **devol.std.argvoid** - описание аргумента "нулевого" типа
- **devol.programtype** - описание необходимых параметров, которые должен предоставить пользователь для работы эволюционного процесса
- **devol.operatormng** - менеджер операторов, хранящий в себе все операторы проблемно-ориентированного языка
- **devol.operator** - базовое описание оператора, которое должен перегружать пользователь для создания своих операторов
- **devol.std.typepod** - описание типа, содержащего простые типы данных
- **devol.std.argpod** - описание аргумента, содержащего значения простых типов данных
- **devol.world** - базовое описание окружения, которое перегружается пользователем
- **devol.argument** - описание аргумента, хранящего значения некоторого типа
- **devol.type** - описание типа, используемого в операторах проблемно-ориентированного языка
- **devol.serializable** - интерфейс, стандартизирующий операции сохранения/загрузки

- **devol.typepng** - менеджер типов, хранящий все типы, используемые в проблемно-ориентированном языке

4.2.3 Особенности выбранных технологий

Все компоненты АСОИ написаны на языке программирования **D**, используются следующие зависимости:

- gtk-d - привязки к библиотеке графических элементов GTK+
- dyaml - пакет, реализующий операции сохранения и загрузки и/в формат YAML
- devol - разработанный автором пакет для генетического программирования

4.2.4 Язык программирования D

D — объектно-ориентированный, императивный, мультипарадигмальный язык программирования, созданный Уолтером Брайтом из компании Digital Mars. Изначально был задуман как переработка языка C++, однако, несмотря на значительное влияние C++, не является его вариантом. В D были заново реализованы некоторые свойства C++, также язык испытал влияние концепций из других языков программирования, таких как Java, Python, Ruby, C# и Eiffel.

При создании языка D была сделана попытка соединить производительность компилируемых языков программирования с безопасностью и выразительностью динамических. Код на языке D обычно работает так же быстро как эквивалентный код на C++, при этом программа на D короче и обеспечивает безопасный доступ к памяти.

Д является мультипарадигменным языком и поддерживает в полной мере следующие парадигмы:

- процедурно-структурное программирование
- объектно-ориентированное программирование
- функциональное программирование
- контрактное программирование
- обобщенное программирование

Д получает все большую популярность из-за преимуществ над предшественником C++. Развитая система обобщенного программирования с переосмысленными шаблонами и примесями вместе с интроспекцией дает мощный инструмент для создания эффективных и гибких приложений, работающих под различными платформами. В D присутствует выполнение функций на этапе компиляции, что позволяет "программировать" компилятор и определять свои проблемно-ориентированные языки.

Д используют сборщик мусора, который можно отключить для тех частей приложения, для которых необходимо ручное управление памятью.

Объектно-ориентированная парадигма языка D является продолжателем идей, заложенных в C++ и развитых в языке Java, но имеет свои особенности. Так, например, классы передаются в функции только по ссылке, без изначально заложенной возможности копирования по значению. В языке отсутствует множественное наследование классов, но имеется множественное наследование интерфейсов.

4.2.5 Язык разметки YAML

YAML — человекочитаемый формат сериализации данных, концептуально близкий к языкам разметки, но ориентированный на удобство ввода-вывода

тических структур данных многих языков программирования. Используется в АСОИ для сохранения информации о проекте.

4.2.6 Библиотека графических элементов GTK+

GTK+ (сокращение от GIMP ToolKit) — кроссплатформенная библиотека элементов интерфейса, имеет простой в использовании API, наряду с Qt является одной из двух наиболее популярных на сегодняшний день библиотек для X Window System. Используется в АСОИ для построения графического интерфейса пользователя.

4.2.7 Архитектура программы. UML-диаграмма классов

Во время проектирования приложения были выбраны следующие директивы:

- Максимальная гибкость. Архитектура приложения должна поддерживать легкое изменение практически всех компонентов, включая добавление и удаление новых типов и операторов проблемно-ориентированного языка
- Максимальная эффективность. Используя богатые возможности языка D, программа должна выполняться как можно быстрее и потреблять как можно меньше памяти, при этом не терять в гибкости и скорости разработки
- Минимальные издержки. Все операции, которые могут быть проведены и верифицированы на этапе компиляции, должны быть перемещены на этап компиляции. Критически важные алгоритмы должны быть протестированы встроенными модульными тестами.

Классы, используемые в АСОИ, можно разделить на 3 типа:

- Ядро - классы, обеспечивающие описание графов, приложения и логики сохранения/загрузки информации
- Графический интерфейс - классы, описывающие графический интерфейс пользователя
- Эволюционные - описание внутреннего проблемно-ориентированного языка генетического программирования
- Внешние - используемые зависимости и классы библиотеки Devol

Диаграмма классов изображена на листе 6 графической части и на рис. 3.

Перечень классов:

- **Application** - содержит проект и все окна приложения, определяет алгоритмы инициализации и завершения работы приложения
- **Project** - проект, хранящий все параметры эволюционного процесса, имя популяции и последнюю найденную популяцию
- **EvolutionWindow** - окно эволюции, в котором отображается процесс эволюции
- **GenericWindow** - хранит в себе общее поведение для всех окон, включая главное меню
- **ResultsWindow** - окно результатов, в котором отображаются индивиды текущей популяции
- **SettingsWindow** - окно настроек эволюции, где пользователь может просмотреть используемые операторы и настроить процесс эволюции
- **IDirectedGraph** - интерфейс, описывающий ориентированный граф
- **ConnListGraph** - реализация ориентированного графа на основе списков смежности
- **GraphWorld** - описание окружения исполнения алгоритмов проверки изоморфизма, отвечает за генерацию входных графов

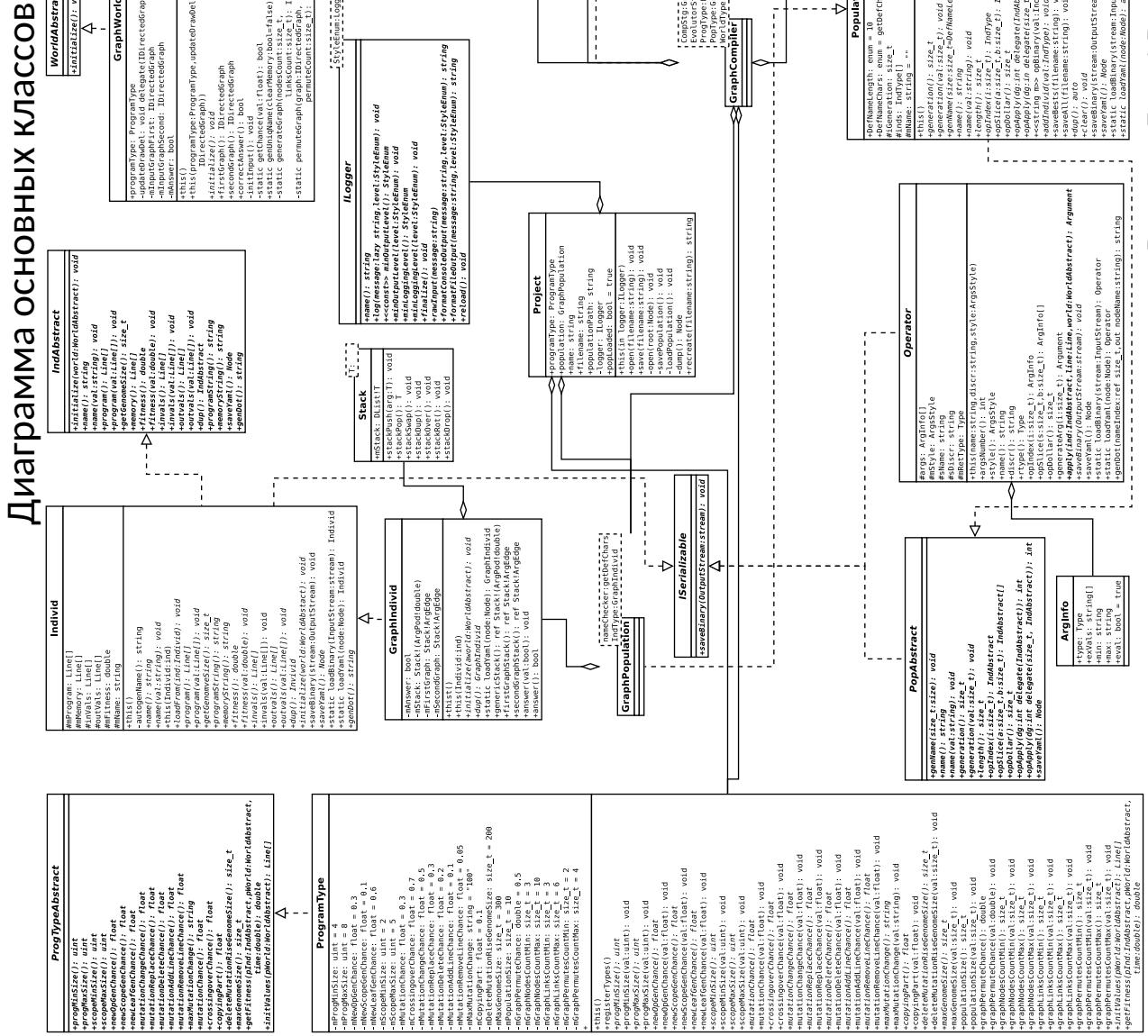


Рисунок 3 – Диаграмма основных классов АСОИ

- **ProgramType** - описание параметров эволюционного процесса, все настройки эволюции находятся в этом классе

- **GraphIndivid** - описание алгоритма проверки изоморфизма графов как индивида в популяции. Каждый индивид содержит в себе три стека:
 - Стек общего назначения с действительным типом элементов
 - Первый входной стек для первого графа
 - Второй входной стек для второго графа
- **GraphPopulation** - контейнер для индивидов, параметризованный типом GraphIndivid
- **GraphCompilation** - описание правил компиляции индивидов и популяций. Является наследником GamePopulation, использующий раундовый метод определения итогового значения функции приспособленности.
- **GraphCompiler** - интерпретатор алгоритмов в терминах проблемно-ориентированного языка для популяций GraphIndivid со стандартными алгоритмами мутации, кросинговера, нахождения нужных поддеревьев над типизированными деревьями.
- **TypeEdge** - описание типа ребра графа для внутреннего языка, является упорядоченной парой индексов вершин
- **ArgEdge** - описание контейнера для значения типа ребра графа
- **AndOperator** - описание оператора логического "И"
- **AnswerOperator** - описание оператора записи ответа на поставленную задачу
- **ConstructOperator** - описание создания аргумента, описывающего ребро графа
- **DistOperator** - описание получения индекса конца ребра графа
- **DivOperator** - описание арифметического оператора деления

- **GenericDupOperator** - описание операции над стеком общего назначения: дублирование вершины стека
- **GenericoverOperator** - описание операции над стеком общего назначения: копирование аргумента под вершиной стека и расположение копии на вершине
- **GenericpopOperator** - описание операции над стеком общего назначения: снятие значения с вершины стека
- **GenericpushOperator** - описание операции над стеком общего назначения: сохранение аргумента в стеке
- **GenericrotOperator** - описание операции над стеком общего назначения: перемещение третьего с вершины аргумента в стеке на вершину
- **GenericswapOperator** - описание операции над стеком общего назначения: перемещение вершины стека под следующий за ней аргумент
- **IntDoubleCastOperator** - описание оператора преобразования целочисленной переменной в действительную
- **InputDupFirstOperator, InputDupSecondOperator** - описание операции над входными стеками: дублирование вершины стека
- **InputOverFirstOperator, InputOverSecondOperator** - описание операции над входными стеками: копирование аргумента под вершиной стека и расположение копии на вершине
- **InputPopFirstOperator, InputPopSecondOperator** - описание операции над входными стеками: снятие значения с вершины стека
- **InputPushFirstOperator, InputPushSecondOperator** - описание операции над входными стеками: сохранение аргумента в стеке
- **InputRotFirstOperator, InputRotSecondOperator** - описание операции над входными стеками: перемещение третьего с вершины аргумента в стеке на вершину

- **InputSwapFirstOperator**, **InputSwapSecondOperator** - описание операции над входными стеками: перемещение вершины стека под следующий за ней аргумент
- **MultOperator** - описание арифметического оператора умножения
- **NotOperator** - описание оператора логического "НЕ"
- **IfOperator** - описание оператора логического ветвления
- **WhileOperator** - описание оператора цикла
- **OrOperator** - описание оператора логического "ИЛИ"
- **PlusOperator** - описание оператора арифметического сложения
- **IntEqualOperator** - описание оператора равенства целочисленных значений
- **DoubleEqualOperator** - описание оператора равенства действительных значений
- **IntGreaterOperator** - описание оператора "больше" для целочисленных значений
- **IntLesserOperator** - описание оператора "меньше" для целочисленных значений
- **IntGreaterEqualOperator** - описание оператора "больше равно" для целочисленных значений
- **IntLesserEqualOperator** - описание оператора "меньше равно" для целочисленных значений
- **DoubleGreaterOperator** - описание оператора "больше" для действительных значений
- **DoubleLesserOperator** - описание оператора "меньше" для действительных значений
- **DoubleGreaterEqualOperator** - описание оператора "больше равно" для действительных значений

- **DoubleLesserEqualOperator** - описание оператора "меньше равно" для действительных значений
- **RoundOperator** - описание оператора округления, преобразования действительного аргумента к целочисленному
- **SourceOperator** - описание получения индекса начала ребра графа
- **ProgramTypeAbstract** - базовое описание параметров, которые должны быть заданы для эволюционного процесса
- **IndAbstract** - базовое описание индивида, которое должно быть задано для эволюционного процесса
- **Individ** - стандартная реализация индивида, которое перегружается пользователем для описания своих индивидов
- **WorldAbstract** - базовое описание окружения исполнения индивида
- **ILogger** - интерфейс для системы логирования. Конкретная реализация скрыта за этим интерфейсом.
- **GameCompilation** - базовое описание стратегии компиляции "игра"
- **Singleton** - описание свойств объекта, который существует только в одном экземпляре
- **Evolutator** - содержит описание эволюционных алгоритмов
- **PopAbstract** - базовое описание контейнера для индивидов
- **Operator** - описание класса-базы для задания операторов проблемно-ориентированного языка
- **ArgInfo** - описание аргумента оператора
- **Stack** - описание стеков, хранящихся внутри индивида
- **Edge** - ребро графа
- **IndexedEdge** - ребро графа, где исходные вершины заменены на индексы, непосредственно используется операторами языка

4.2.8 Выбор программных средств

Выбор операционной системы для разработки программного продукта

Изначально АСОИ разрабатывалась для использования в окружении операционной системы GNU/Linux, но так как при проектировании не были использованы платформозависимые возможности языка и не использованы зависимости, имеющие только одну основную платформу, то данный продукт должен работать и под другими платформами: MacOS, Windows, семейство BSD. Необходимым условием является наличие компилятора языка D и реализации библиотеки GTK+.

Выбор системы спопровождения разработки В качестве системы сборки и пакетного менеджера использовался пакетный менеджер **Dub**, который на данный момент является безальтернативным практически стандартным средством для управления пакетами на D. В качестве системы контроля версий была выбрана **git** и репозиторий на домене **github.com**, так как это основной способ публикации пакетов в **Dub registry** - официальный регистр пакетов для языка программирования D.

Выбор среды разработки программного продукта Разработка велась в окружении операционной системы GNU/Linux, в которой в качестве IDE практически безальтернативно является **Eclipse** с плагином **Descent** для разработки на языке программирования D. Кроссплатформенность, интеграция с пакетным менеджером **Dub**, бесплатность, навигация по коду и интеграция с системами контроля версий делают **Eclipse** очевидным выбором.

4.2.9 Выбор аппаратных средств

АСОИ предназначена для работы на компьютерах одной из архитектур: x86, x86_64. Минимальные характеристики, необходимые для работы:

- Процессор, поддерживающий архитектуру x86_64 с тактовой частотой не менее 1.5 ГГц
- Оперативная память от 1 Гб
- Графический ускоритель и монитор, способные отображать графический интерфейс операционной системы
- Устройства ввода: мышь и клавиатура

4.2.10 Разработка основных алгоритмов обработки информации

АСОИ использует широкий спектр нетривиальных алгоритмов:

- Алгоритм типизированной мутации
- Алгоритм типизированного кроссинговера
- Алгоритм генерации типизированного дерева
- Алгоритм получения новой популяции
- Алгоритм выбора случайного узла дерева

4.2.11 Типизированная мутация

Операция состоит из двух частей:

- глобальная мутация:
 - добавление новой линии в геном
 - удаление линии из генома
- локальная мутация:

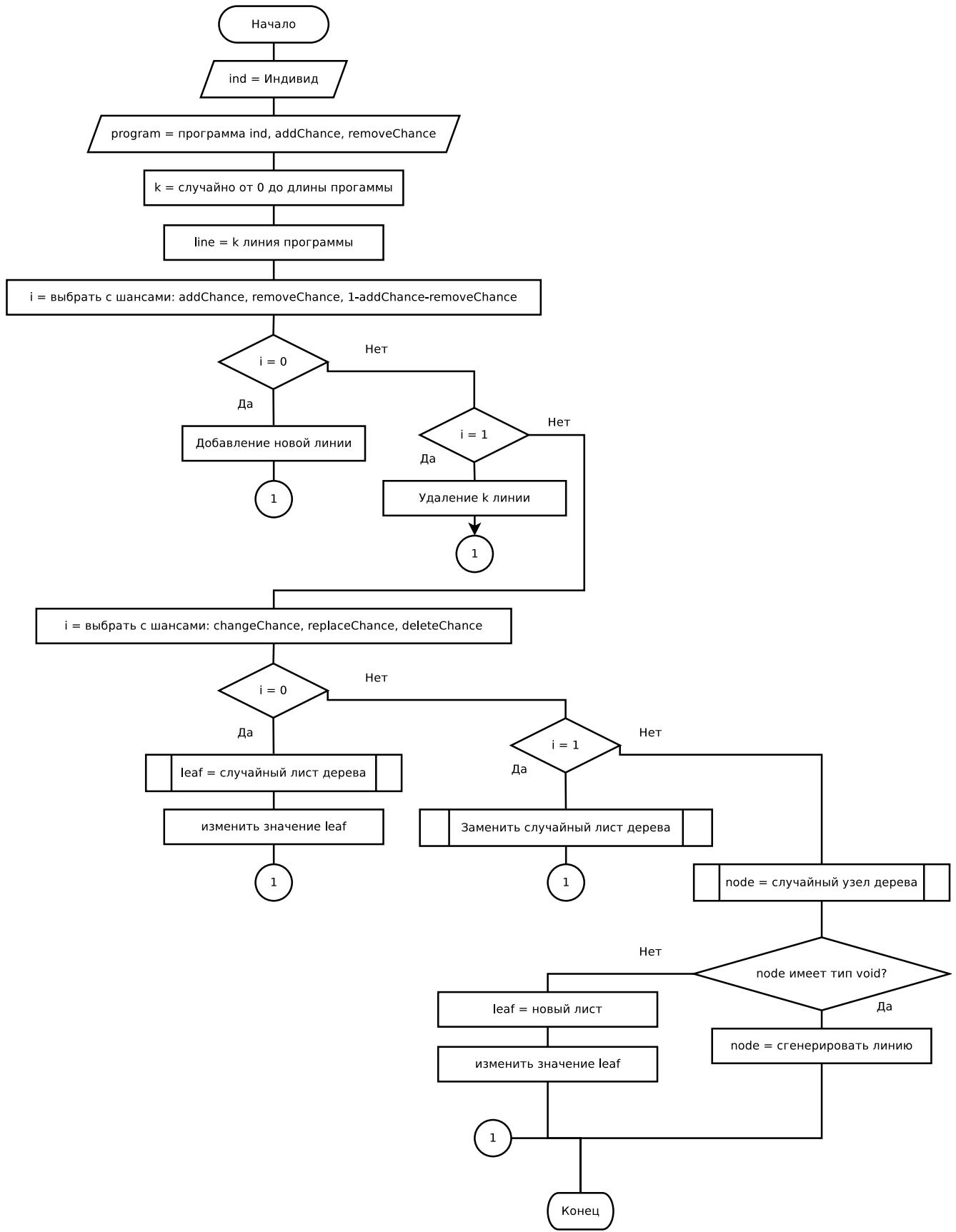
- изменение значения аргумента
- замена листа дерева
- удаление узла

Алгоритм использует функцию броска кубика с заданными вероятностями для каждой грани:

$$k = \text{randomRange}([chance_1, chance_2, chance_3])$$

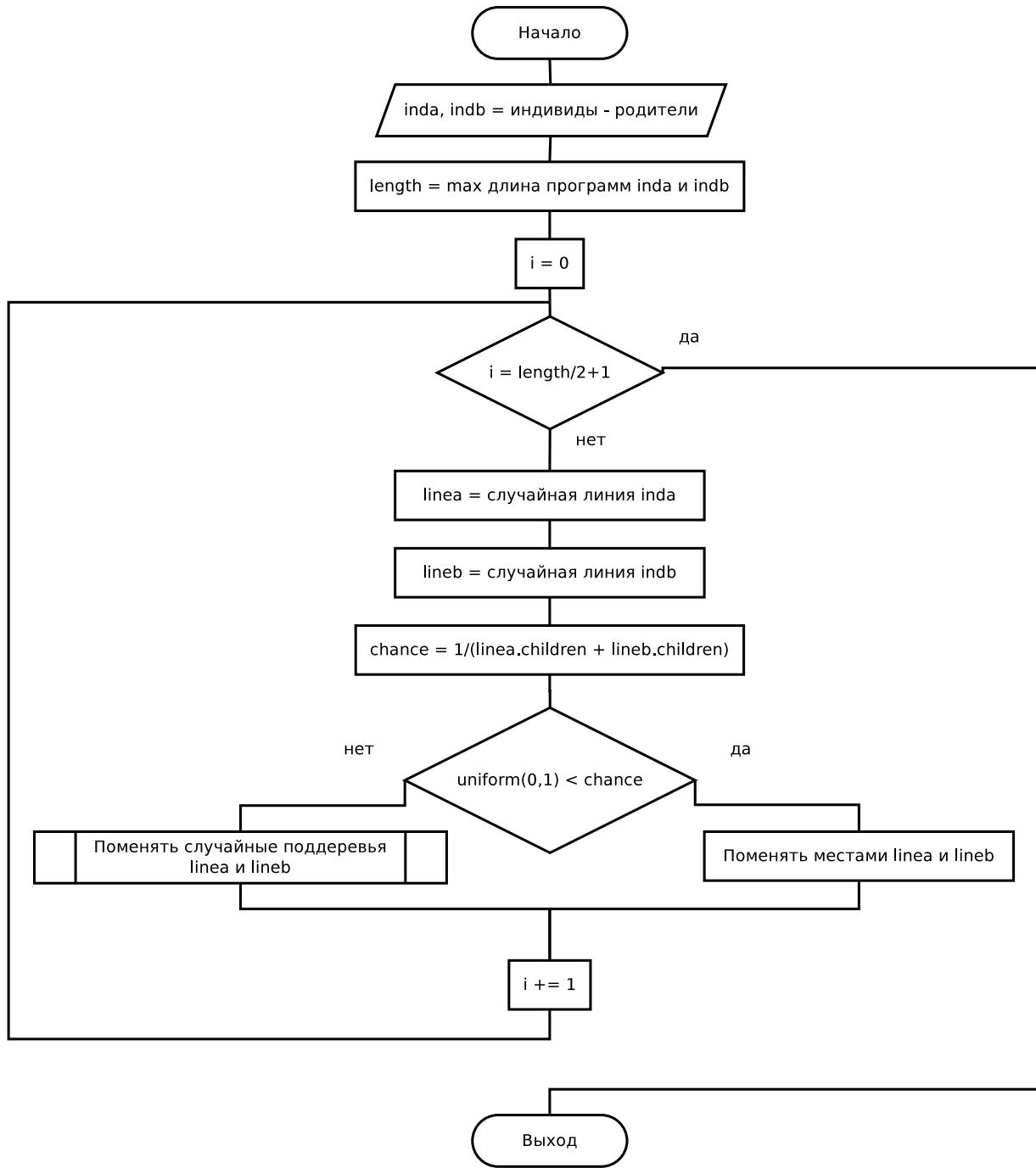
Значение k будет равно 0 с вероятностью $chance_1$, 1 с вероятностью $chance_2$, 2 с вероятностью $chance_3$.

При попытке удаления узла с типом *void* необходимо генерировать новую линию, так как простой аргумент пустого типа является исключительной ситуацией и не несет никакой информации, полезной для решения задачи.



4.2.12 Типизированный кроссинговер

Используемый алгоритм кроссинговера относится к гибридному подходу: одновременно идет обмен целыми линиями генома и обмен поддеревьями определенных линий. Но шанс обмена целыми линиями обратно пропорционален количеству узлов в деревьях. Это необходимо для обеспечения равномерности распределения выбора узлов дерева для кроссинговера.



4.2.13 Генерация типизированного дерева

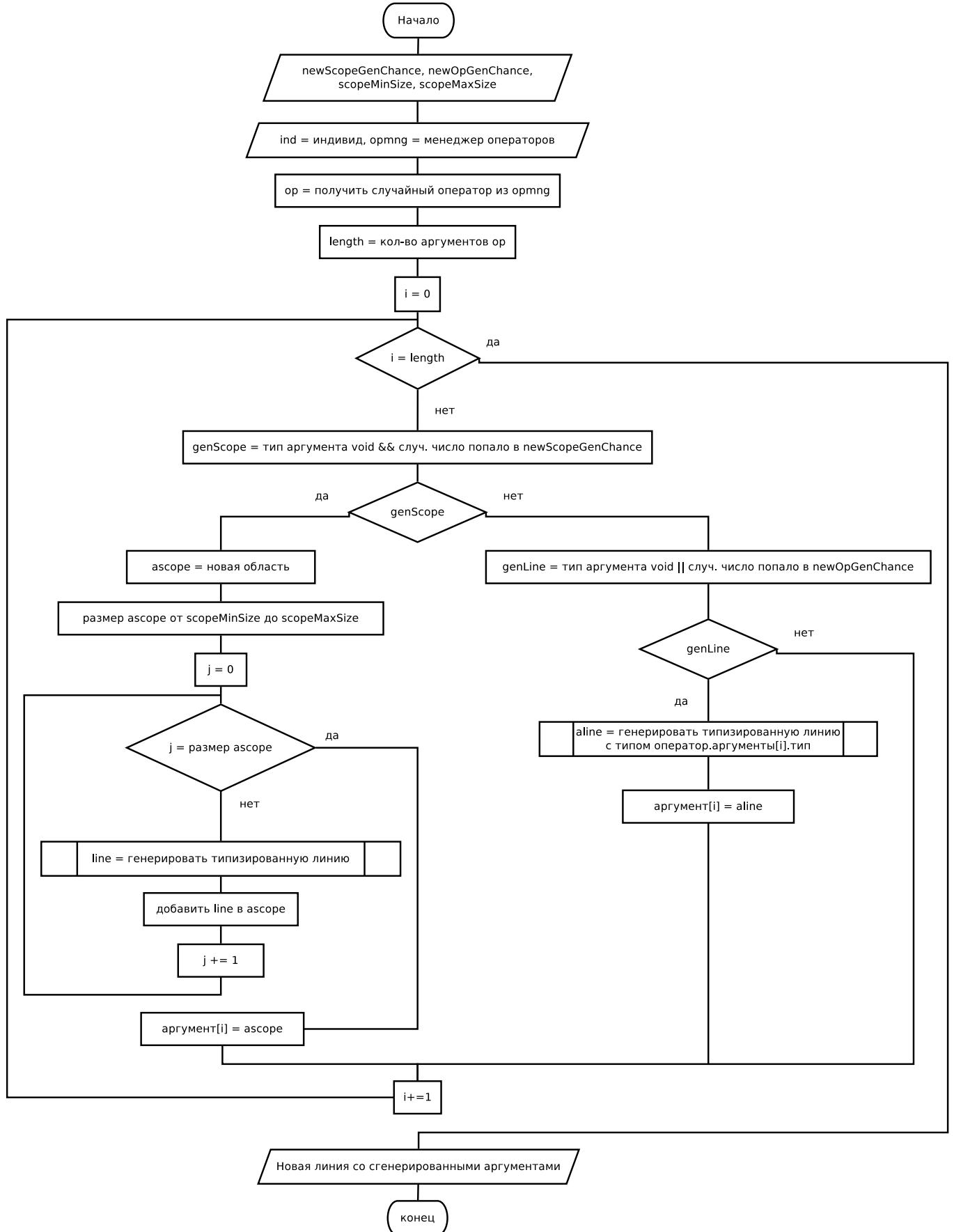
Алгоритм состоит из двух больших частей:

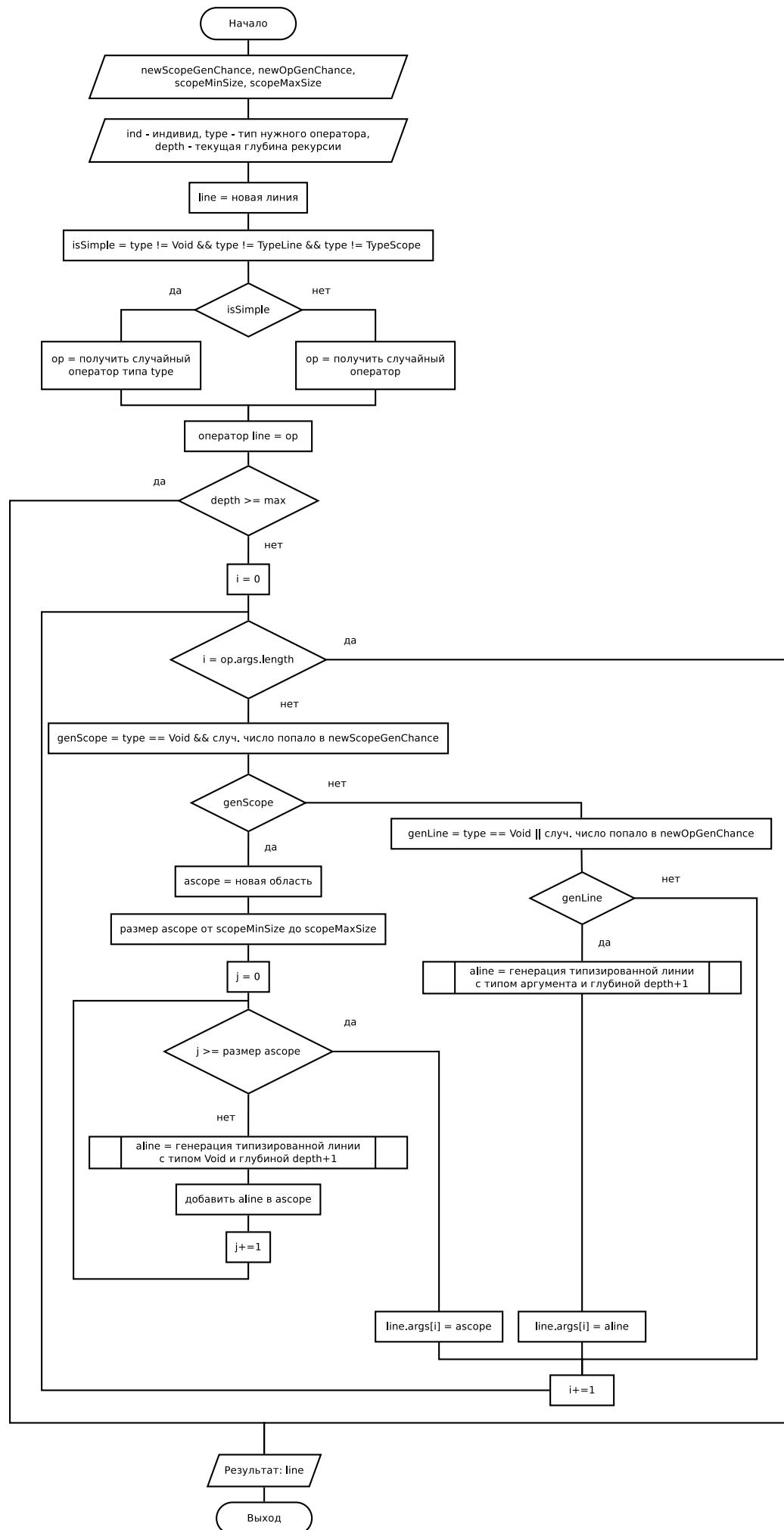
- генерация нетипизированной линии для корневой области
- генерация типизированной линии для соблюдения строгой типизации

При генерации аргумента с пустым типом возможны два сценария:

- генерация линии, тип оператора которой определяется аргументом, для которого проводится генерация
- генерация области с размером от scopeMinSize до scopeMaxSize (задается пользователем)

Алгоритм является рекурсивным и останавливается либо самостоятельно из-за генерации констант, либо из-за генерации деревьев максимальной глубины. Специальное ограничение на глубину необходимо, так как нельзя заранее сказать какой глубины будет сгенерировано дерево (может закончиться доступная память), тем более параметры данной генерации задаются пользователем.



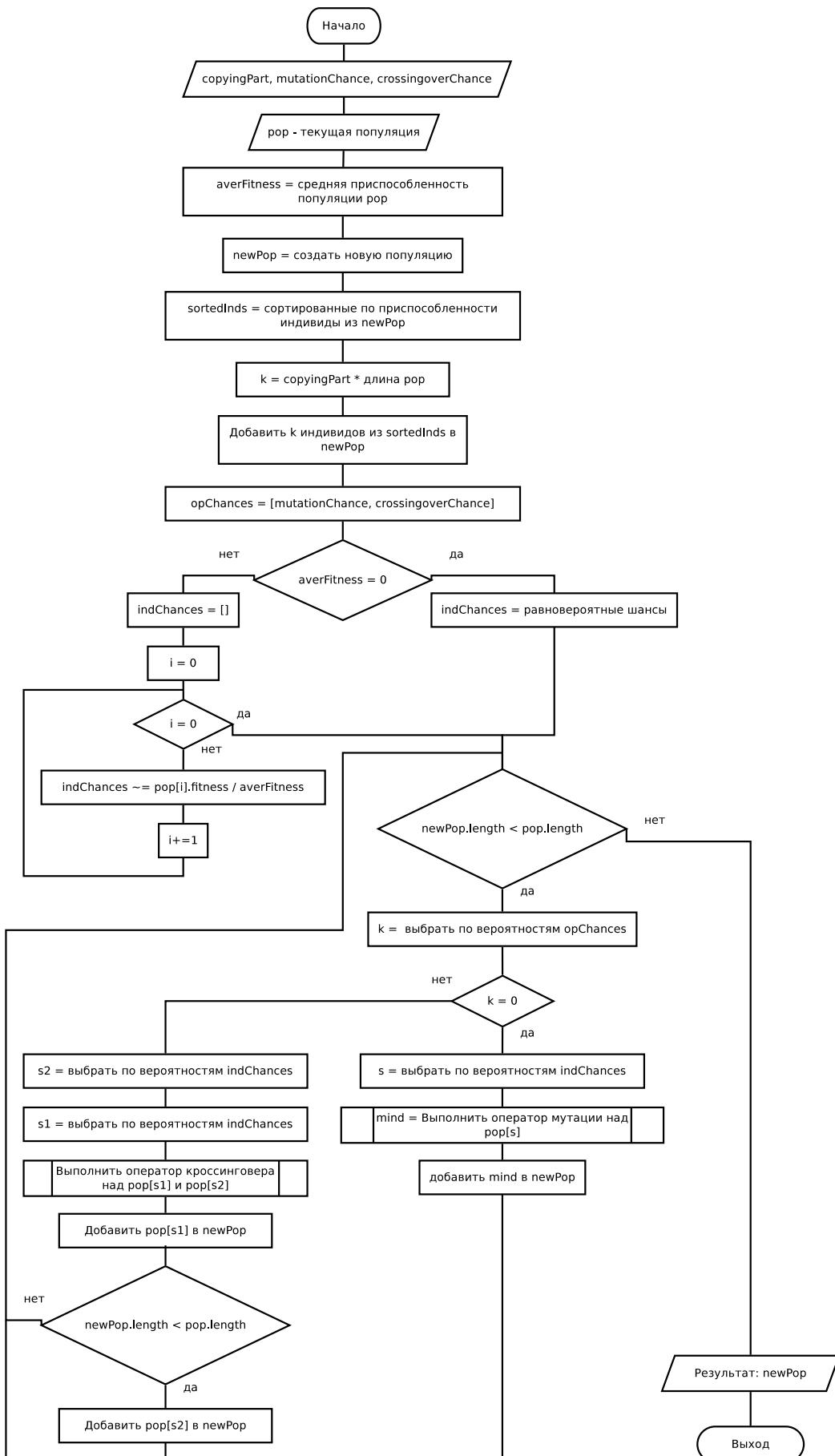


4.2.14 Получение новой популяции

В АСОИ используется практически классический подход при генерации новой популяции. Сначала индивиды сортируются по убыванию функции приспособленности и в новую популяцию попадает часть лучших (`copyingPart`, задается пользователем), этот прием копирования "элитных" геномов позволяет избежать деградации популяции. Далее рассчитывается среднее значение функции приспособленности, на основе которого рассчитываются вероятности каждого индивида для генетических операций (алгоритм нормализует массив вероятностей):

$$P_i = \frac{f_i}{f_{max}}$$

В соответствии с вероятностью каждого индивида и шансом операции выполняются либо операция мутации, либо кроссинговер. Обработанные индивиды заполняют новую популяцию. Нужно учитывать, что при нечетном размере популяции один из последних индивидов должен отбрасываться, чтобы избежать утечки памяти.



4.2.15 Выбор случайного узла дерева

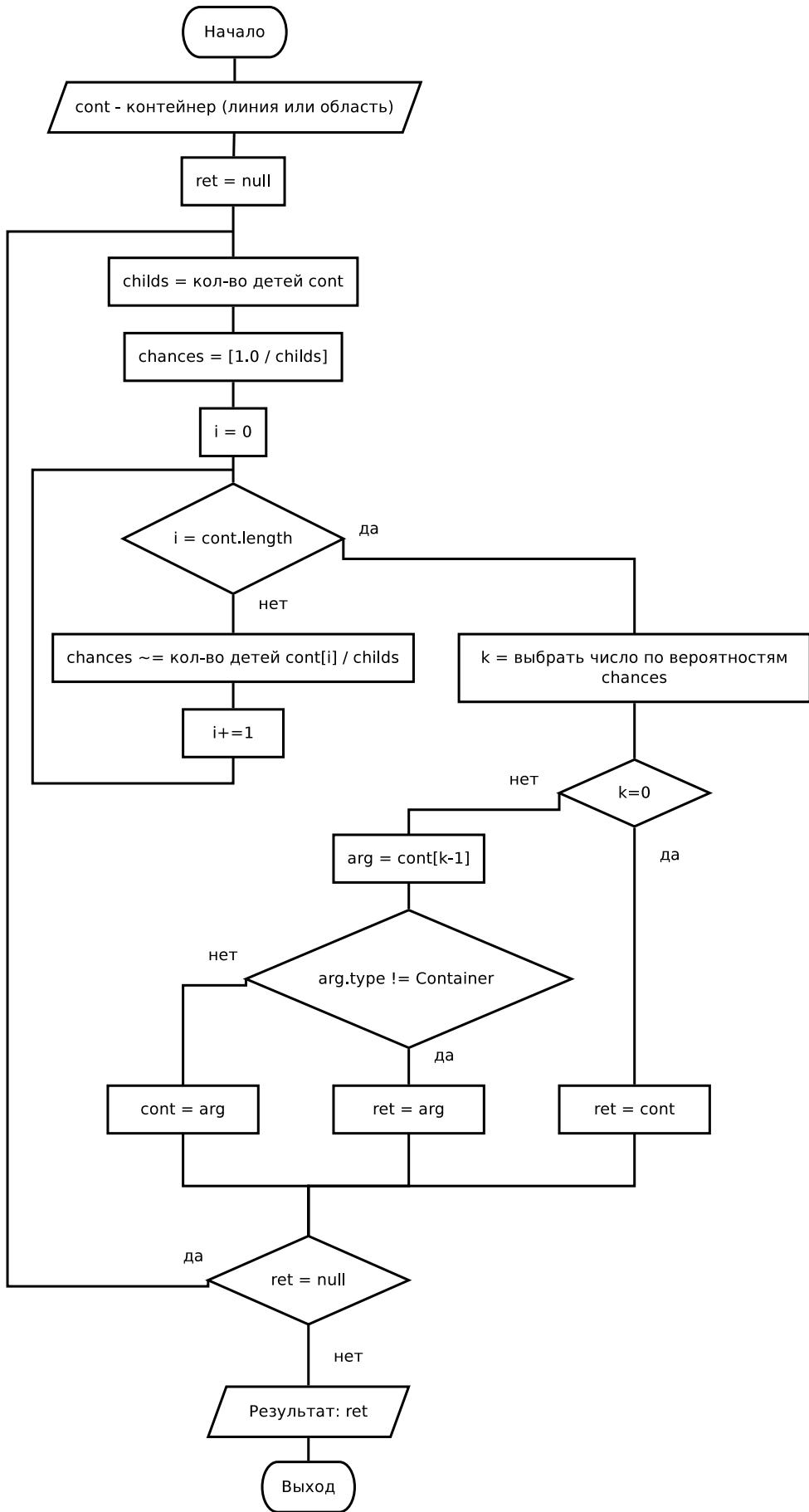
Данный алгоритм производит выбор узла дерева с равномерным распределением. Для того, чтобы реализовать требование равномерности, расчет вероятности выбора элемента и перехода на уровень вглубь рассчитываются в зависимости от количества потомков дерева. Вероятность выбрать данный узел:

$$p_0 = \frac{1}{tree.childs}$$

Вероятность выбрать i -ый аргумент:

$$p_i = \frac{arg_i.childs}{tree.childs}$$

Алгоритм останавливается, когда достигается лист или выбирается один из узлов.



5 Технологическая часть

5.1 Разработка интерфейса взаимодействия с пользователем

Целью данного этапа является разработка удобного и простого интерфейса, который позволяет пользователю работать с АСОИ с максимальной продуктивностью.

Графический интерфейс пользователя разрабатывался в конструкторе графических интерфейсов пользователя **Glade** для **GTK+**. Для удобства представления и ввода данных были разработы три экранных формы, каждая из которых отвечает за свою часть требований технического задания:

- Экранная форма настроек эволюционного процесса. Через нее осуществляется просмотр операторов проблемно-ориентированного языка и задание настроек эволюционного процесса. Предусматривается проверка входных данных на корректность.

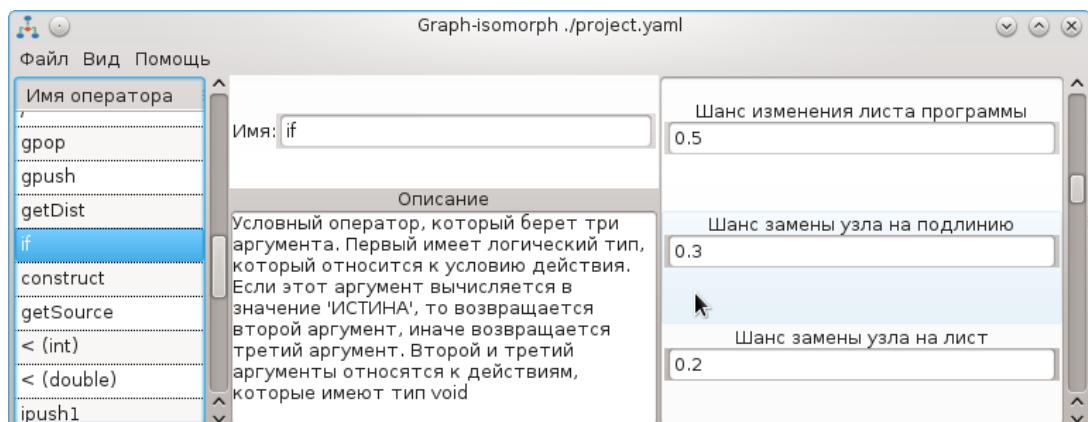


Рисунок 4 – Графическая форма настроек эволюционного процесса

- Форма управления процессом эволюции. На данной форме отображается статус обработки нового поколения и текущие изображения входных графов для тестируемых алгоритмов.

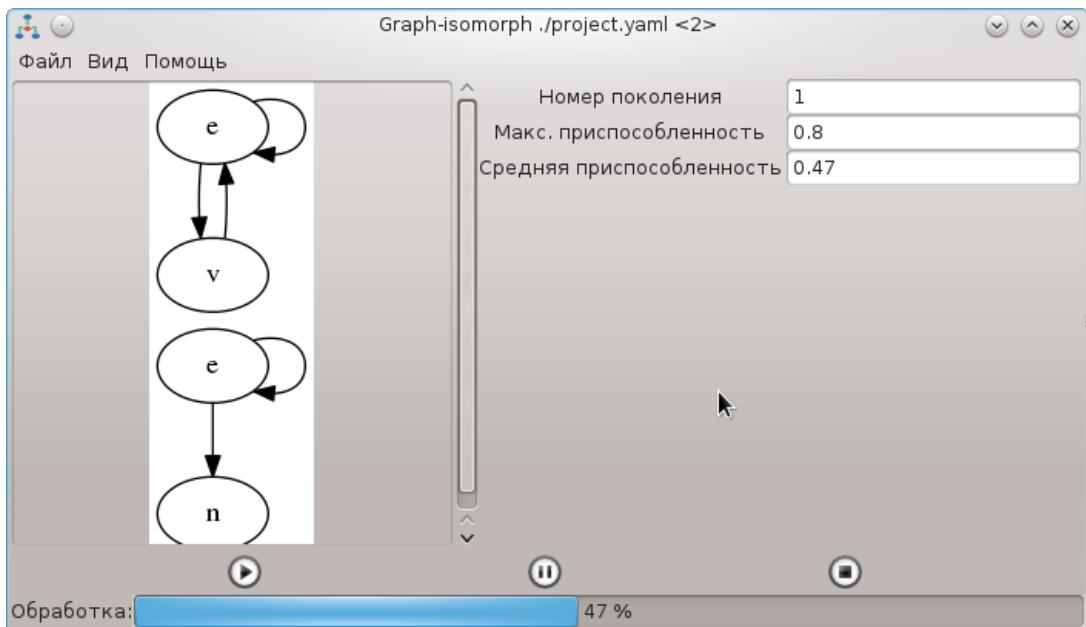


Рисунок 5 – Экранная форма управления процессом эволюции

- Форма просмотра результатов эволюции. На данной форме отображаются: состав текущей популяции, текстовое представление исходного кода алгоритма-индивидуа, графическое представление кода алгоритма-индида.

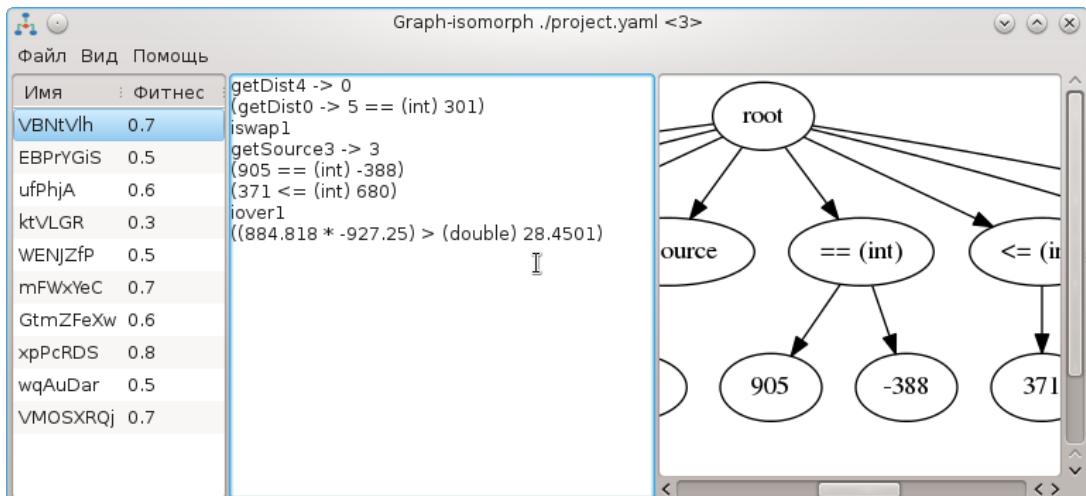


Рисунок 6 – Экранная форма управления процессом эволюции

Вывод изображений исходного кода и изображений входных графов осуществляется через систему визуализации графов **Graphviz**, перед этим алгоритмы проходят специальную обработку для представления их исходного кода в специальной нотации, понятной системе визуализации.

Нужно заметить, что интерфейс программы разрабатывался с учетом возможных изменений его внешнего вида без перекомпиляции программы. Для этого

описание интерфейса хранится в отдельном файле с расширением **.glade**, который загружается при старте приложения, и по информации из этого файла строится внешний вид графического интерфейса пользователя. В программе есть возможность использовать нестандартный путь до файла описания интерфейса, путем передачи своего имени файла при старте приложения через ключ **-gui**.

Полный граф диалога с пользователем изображен на рис. 7 и на листе 7 графического приложения.

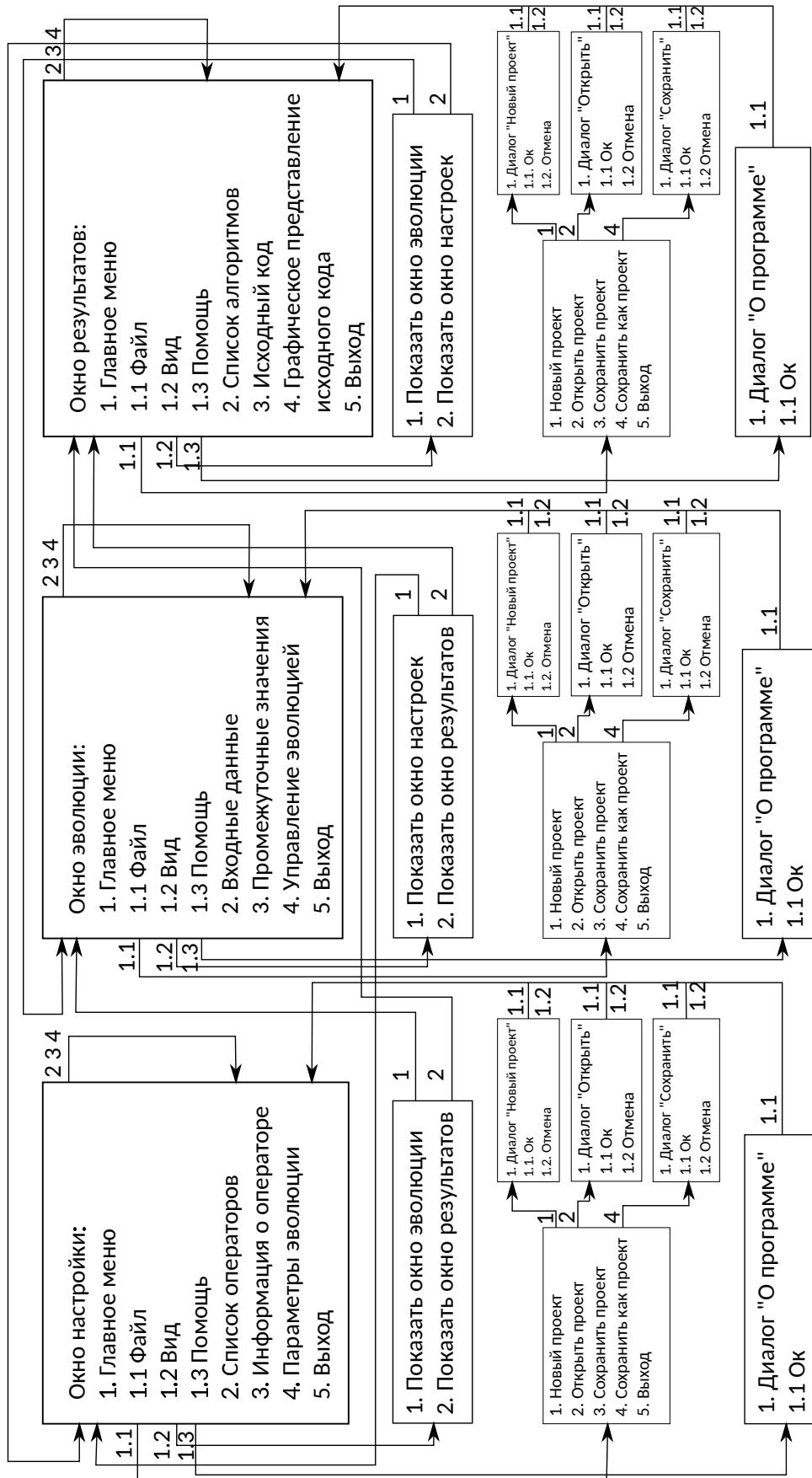
5.2 Разработка форматов входных и выходных данных программы

5.2.1 Разработка форм входных данных

Входные данные программы делятся на два вида:

- Входные данные, вводимые пользователем через экранные формы. Пользователь, используя графические элементы ввода, предоставляет детальные настройки процесса эволюции.
- Входные данные, загружаемые из проектного файла. Все введенные пользователем данные сохраняются в так называемом **проекте**, для возможности продолжить поиск с момента предыдущей остановки эволюционного процесса. Путь до файла проекта предоставляются АСОИ через диалоговое окно загрузки проекта или при старте программы в списке параметров (через ключ **-proj**).

Граф диалога с пользователем



5.2.2 Разработка форм выходных данных

Выходные данные программы отображаются только через графический интерфейс пользователя. Основные виды выходных данных:

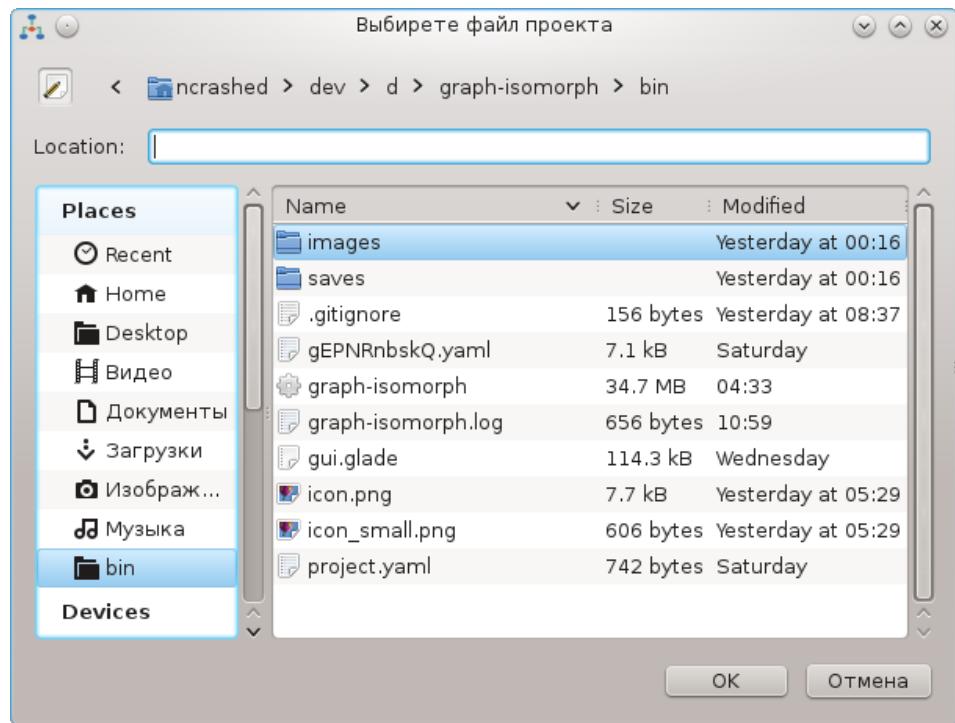


Рисунок 8 – Диалог открытия файла проекта

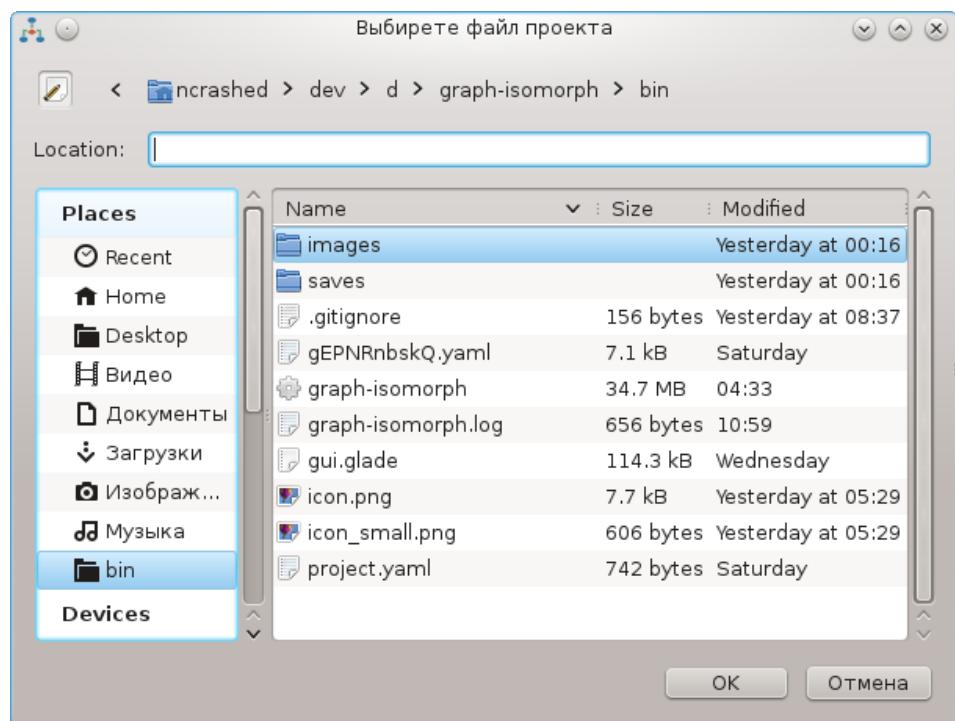


Рисунок 9 – Ввод параметров эволюционного процесса

- Названия и описания операторов проблемно-ориентированного языка.
- Значения параметров эволюционного процесса.
- Текстовая форма исходного кода найденных алгоритмов. Внутренние структуры проблемно-ориентированного языка преобразуются в человекочитаемый текст псеводязыка программирования с Си-подобным синтаксисом.

сисом.

- Графическая форма исходного кода найденных алгоритмов. Структуры проблемно-ориентированного языка преобразовываются в код на языке **dot** и визуализируются через систему визуализации графов **Graphviz**.

6 Исследовательская часть

6.1 Разработка проблемно-ориентированного языка

Метод генетического программирования требует тщательной подготовки внутреннего представления индивидов-программ. Традиционные методы формальных языков программирования не подходят для этого представления, так как на используемый язык накладываются нестандартные ограничения:

- Внутреннее представление должно быть удобным для работы в эволюционном процессе, то есть программа должна большую часть времени быть представлена в виде абстрактного синтаксического дерева. Преобразование в текст и обратно должно осуществляться только для демонстрации программы пользователю или для сохранения промежуточных результатов.
- В то же время язык для внутреннего представления должен быть удобным для введения новых операторов, расширения возможностей индивидов-программ. Такое эффект легко достигается, если используется концепция EDSL - встраиваемые проблемно-ориентированные языки, которые строятся поверх других языков общего назначения.

В рамках бакалаврского проекта были расширены и доработы предыдущие наработки автора в области генетического программирования. Данный EDSL получил название **D**Evol**** - D Evolution. Далее будут рассмотрены основные концепции данного проблемно-ориентированного языка.

Далее будут рассмотрены основные компоненты **D**Evol**** и представлено подмножество языка, которое доступно для расширения описанием предметной области решаемой задачи. Эти компоненты включают в себя:

- Операторы - описывают вычисления над типизированными значениями. Примерами операторов являются: сложение чисел, изменение скорости

вращения колеса робота, считывание информации с датчиков и т.д.

- Типы - описывают структуру значений, контролируют корректность программы. Примерами типов являются: целочисленные числа, логические значения, дуги графа, матрицы и пр.
- Аргументы - конкретные значения определенного типа. Описание аргумента также содержит информацию для генетических операторов: максимальное и минимальное значение, запрещенные значения, правила генерации случайных значений.

Определив множество операторов, типов и аргументов, полностью описывающих предметную область, пользователь получает готовую систему генетического программирования, которая в автоматизированном режиме решает поставленную задачу.

6.2 Представление генетического кода программ

Внутреннее представление **DEvol** базируется на следующих понятиях: **Дерево** - связный ациклический граф.

АСТ - абстрактное синтаксическое дерево, узлы которого помечены операторами языка, а листья с соответствующими operandами языка.

Лес - упорядоченное множество упорядоченных деревьев.

Исходный код любого индивида представляет из себя лес из специальных АСТ, называемых **линиями**. Схема исходного кода индивида представлена на рис. 10.

Программа индивида выполняется императивно, начиная с первой линии и заканчивая последней. Проблемно-ориентированный язык **DEvol** является императивным, статически типизированным языком. Изначально в языке не было операторов с побочными действиями, но для реализации решаемой в данной ра-



Рисунок 10 – Представление исходного кода индивида. Узел "Корень" введен для наглядности. Последовательность выполнения программы слева на право.

боте задачи, он был расширен от чистого функционального до императивного с наличием "грязных" (в терминах функциональных языков) операторов.

6.2.1 Линия

Линия - дерево, корнем которого является оператор, а листья являются его аргументами. Количество аргументов и их последовательность определяется арностью оператора и его спецификацией. Схема линии представлена на рис. 12.

Линия возвращает значение типа, определяемое ее оператором. Линии могут быть вложены в аргументы путем обрачивания в специальный системный тип. Такие аргументы вычисляются прежде чем вычисляется оператор, владеющий данным аргументом.

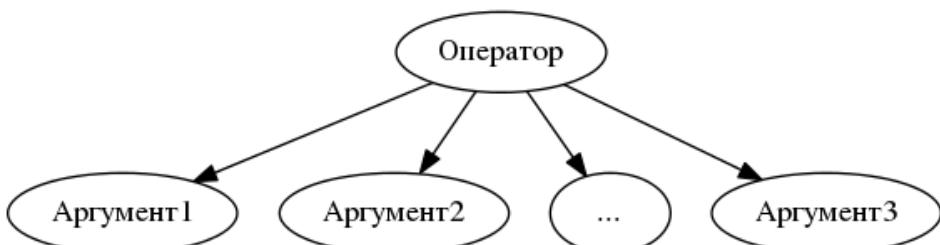


Рисунок 11 – Схема линии, состоящей из оператора и набора аргументов.

Линия является основным элементом, которым оперирует генетические операторы (мутация и кроссинговер). Каждая линия имеет свой тип, который определяется возвращаемым типом оператора. Также каждый аргумент принадлежит одному из типов, определенных в проблемно-ориентированном языке. На-

личие типизации значительно усложняет корректную реализацию генетических операторов, но позволяет описывать большее множество программ. Данная особенность является ключевой в **DEvol** по отношению к большинству реализаций метода генетического программирования.

6.2.2 Область

Область - лес, состоящий из линий. Данное понятие было введено в язык для реализации последовательности операторов аналогичной "блокам" из популярных языков программирования.



Рисунок 12 – Схема области, состоящей из набора линий.

Область может быть вложена в аргументы путем обворачивания в системный тип. Такие аргументы всегда имеют тип **voidtype**, так как внутри области могут быть операторы с различными возвращаемыми типами.

6.2.3 Оператор

Оператор - определенный пользователем алгоритм вычисления значения определенного типа и/или изменения состояния системы на основе значения вычисленных входных значений аргументов, принадлежащих определенному типу. Это основной объект, который должен быть определен в **DEvol** для решения поставленной задачи.

Каждый оператор состоит из:

- а) Уникальное название
- б) Описание, содержащее поясняющую информацию о назначении оператора
- в) Возвращаемый тип
- г) Количество и типы входных аргументов
- д) Дополнительные ограничения на входные аргументы
- е) Описание процедуры оператора, которой доступна информация об индивиде, его окружении и аргументы оператора.

Пользователь описывает оператор на языке **D** путем реализации абстрактных методов класса **Operator**, ниже представлен пример оператора сравнения двух чисел:

```
class PlusOperator : Operator
{
    TypePod!double doubletype;

    enum description = "Ariphmetic operation for adding real numbers";

    this()
    {
        doubletype = cast(TypePod!double)(TypeMng.getSingleton().getType
            ↪ ("Typedouble"));
        assert(doubletype, "We need double type!");
    }

    mRetType = doubletype;
    super("+", description, ArgsStyle.BINAR_STYLE);

    ArgInfo a1;
    a1.type = doubletype;
    a1.min = "-1000";
    a1.max = "+1000";
```

```

    args ~= a1;
    args ~= a1;
}

override Argument apply(IndAbstract ind, Line line, WorldAbstract
    ↪ world)
{
    auto ret = doubletype.getNewArg();

    auto a1 = cast(ArgPod!double)(line[0]);
    auto a2 = cast(ArgPod!double)(line[1]);

    ret = a1.val + a2.val;
    return ret;
}
}

```

После реализации класса нового оператора необходимо его зарегистрировать в менеджере операторов, чтобы системе стало известно о доступности нового оператора.

Значения аргументов по-умолчанию вычисляются заранее, до вызова метода **apply**, но для некоторых управляющих операторов, таких как: операторы циклов, условные операторы - такое поведение не является желательным. Для настройки процесса вычисления значений аргументов был введен механизм отложенных вычислений (так называемая **lazy evaluation**). Для этого в ограничения типов необходимо поставить значение **false** в поле **eval** структуры описания ограничений аргумента **ArgInfo**.

6.2.4 Тип и Аргумент

Каждое значение в **DEvol** имеет тип, который описывает множество допустимых значений. Введение типизации в генетическое программирование позволяет решить проблему корректности генерируемых программ. Разработанные алгоритмы первичной генерации, мутации и кроссинговера всегда выдают корректные с точки зрения типов программы. Таким образом отсекается целое множество программ, которые принципиально не являются решениями задачи.

В **DEvol** типы делятся на две категории:

- Системные типы - типы, без которых не может оперировать система генерации программ и их выполнения. Исключение этих типов из проблемно-ориентированного языка приводит к полной неработоспособности системы, так как генетические операторы и алгоритм вычисления графа работают с использованием этих типов. Перечисление этих типов и их описание дано в таблице 5.
- Пользовательские типы - к ним относятся все типы, необходимые для описания предметной области рассматриваемой задачи. Исключение этих типов из языка не приводит к неработоспособности системы.

Каждому типу соответствует описание аргумента, содержащее информацию для генетических операторов: максимальное и минимальное значение, запрещенные значения, правила генерации случайных значений.

Пользовательские типы реализуется через реализацию абстрактных методов класса **Type** и реализацией абстрактных методов класса **Argument** для соответствующего типа аргументов. Далее новый тип регистрируется в менеджере типов, чтобы система узнала о существовании типа.

Таблица 5 – Системные типы

Название	Описание
voidtype	Пустой тип, множество его значений содержит единственное значение void . Используется как тип оператора, для которого главной целью является изменение состояния окружения индивида.
typeline	Позволяет вкладывать линии в аргументы других линий. При вычислении аргумента с линией заменяется на тип, возвращаемый оператором линии.
typescope	Позволяет вкладывать области в аргументы других линий. При вычислении аргумента всегда заменяются на void .

6.3 Обеспечение тьюринг-полноты DSL

При выборе базиса проблемно-ориентированного языка следует соблюдать осторожность. Если на предоставленных операторах будет невозможно алгоритмически реализовать решение проблемы, то процесс эволюции никогда не закончится, так как решение не попадет в пространство поиска.

Самый простой способ обеспечить попадание нужного решения в пространство поиска - обеспечить тьюринг-полноту полученного языка. В теории вычислимости язык называется **тьюринг-полным**, если на нём можно реализовать любую вычислимую функцию. Другими словами, для каждой вычислимой функции существует вычисляющая её программа на этом языке.

Формальное доказательство полноты языка по Тьюрингу является очень затратным и объемным, поэтому для большинства языков программирования используется другой подход. Достаточно свести проверяемый язык к языку, для которого уже доказана тьюринг-полнота.

6.3.1 Выбор базисного языка

Существует множество тьюринг-полных языков программирования, к которым можно свести множество операторов и типов проблемно-ориентированного языка. Поэтому проведем выбор базисного языка, при реализации операторов и типов которого, автоматом получаем тьюринг-полный проблемно-ориентированный язык.

Для выбора базисного языка будем использовать следующие критерии:

- a) Простота освоения и читаемость – характеризует субъективную оценку сложности разбора программ на данном языке, этот критерий важен при анализе найденных решений.
- б) Модель работы с памятью – различные языки используют разные модели памяти:
 - Непрерывная память – вся память является последовательностью байтов, адресация производится через указатели.
 - Стековая память – все операции работы с памятью модифицирует стек (или несколько стеков). **Стек** – структура данных, представляющая собой список элементов, организованных по принципу «последним пришёл – первым вышел».
 - Неявная память – вся память сосредоточена в аргументах функций и в функциях. В отличие от остальных моделей, не требует реализации дополнительных операторов для работы с памятью.
- в) Сложность реализации побочных эффектов – многие функциональные языки имеют сложности с реализацией побочных эффектов, так как они плохо вписываются в теоретическую базу языка, поэтому при использовании такого языка необходимо также реализовывать механизмы

обеспечения побочных эффектов. Побочный эффект – возможность в процессе выполнения своих вычислений: читать и модифицировать значения глобальных переменных, осуществлять операции ввода-вывода, реагировать на исключительные ситуации, вызывать их обработчики.

- г) Независимость операций – многие операторы не являются корректными или приводят к исключительным ситуациям при неправильном сочетании с другими операторами, такой зависимости следует избегать, так как при генерации программ и генетических операциях.
- д) Объем необходимых операций – определяется минимальным количеством операций, которые составляют ядро языка, т.е. тех операций, на которых может быть реализована любая программа на этом языке. Этот критерий напрямую влияет на количество операторов в итоговом проблемно-ориентированном языке.

C Один из самых распространенных системных языков программирования. Основывается на императивной парадигме, использует непрерывную модель памяти. При неправильной последовательности объявления и использования переменных можно получить семантически некорректную программу. При активном использовании указателей читаемость программ резко снижается.

Forth Язык на парадигме стековой машины, соответственно использует стековую модель памяти. Главным достоинством языка является простота базовых операций языка вместе с достаточной независимостью этих операций. Без подготовки сложность разбора программ средняя.

Core Чистый функциональный язык с неявной моделью памяти. Легко читается и понимается, включая программы, которые были сгенерированы случайным образом. Самая высокая степень независимости команд, но требует сложной реализации побочных эффектов.

Ассемблер Семейство языков ассемблеров основывается на императивной парадигме, имеет самое малое количество операций, но программы практически нечитаемы при случайной генерации. Однако на ассемблере большая часть программ приводят к повреждению памяти машины и к исключительным ситуациям, что значительно усложняет алгоритмы генерации корректных программ.

Распределим ранги рассмотренным языкам программирования (таблица 6). Предпочтение отдается легкости реализации всех подсистем АСОИ, если выбирать за основу данных язык программирования.

Таблица 6 – Таблица рангов для выбора базисного языка программирования

Критерий	C	Forth	Core	Ассемблер
Читаемость	3	2	1	4
Модель памяти	3.5	1	2	3.5
Независимость операций	3	2	1	4
Побочные эффекты	2.5	2.5	4	1
Σ	12	7.5	8	12.5

Итого по таблице 6 необходимо выбрать язык **Forth** как базу для реализации операторов и типов проблемно-ориентированного языка.

6.4 Описание DSL для решения задачи

На основе языка **Forth**, который был выбран в предыдущей части, был разработан набор операторов (таблица 7) и типов (таблица 8). Для входных графов используется по одному стеку, которые хранят ребра графа. Также предусмотрен стек общего назначения для действительных чисел. Программа может сообщить свой ответ через специальный оператор. Присутствуют специальные операторы для преобразования типов и для работы с ребрами графа.

Таблица 7 – Операторы проблемно-ориентированного языка

Класс оператора	Название	Описание
-----------------	----------	----------

Логические операции	Логическое «ИЛИ»	Берет два значения, вычисляет логическое «ИЛИ».
	Логическое «И»	Берет два значения, вычисляет логическое «И».
	Логическое «НЕТ»	Берет значение, вычисляет логическое отрицание.
Операции со стеком	gswap	Меняет местами два последних значения в стеке общего назначения.
	grot	Перемещает третий элемент с головы стека на вершину. Стек общего назначения.
	gpop	Взятие головы со стека общего назначения.
	gpush	Сохраняет действительное число в стек общего назначения.
	gdup	Дублирует голову стека общего назначения.
	gover	Дублирует значение под головой стека на вершину. Стек общего назначения.
Операции со стеками входных данных	iswap1, iswap2	Меняет местами два последних значения в стеке входного графа.
	irot1, irot2	Перемещает третий элемент с головы стека на вершину. Стеки входных графов.
	ipop1, ipop2	Взятие головы со стека входных графов.

	ipush1, ipush2	Сохраняет действительное число в стек входного графа.
	idup1, idup2	Дублирует голову стека входного графа.
	iover1, iover2	Дублирует значение под головой стека на вершину. Стеки входных графов.
Операторы отношения	==	Сравнивает два значения на равенство и возвращает логическое значение.
	<	Возвращает «ИСТИНА» тогда и только тогда, когда второе значение больше первого.
	<=	Возвращает «ИСТИНА» тогда и только тогда, когда второе значение больше или равно первому.
	>	Возвращает «ИСТИНА» тогда и только тогда, когда второе значение меньше первого.
	>=	Возвращает «ИСТИНА» тогда и только тогда, когда второе значение меньше или равно первому.
Управляющие операторы	while	Вычисляет второй аргумент пока первый аргумент вычисляется в значение «ИСТИНА».

	if	Вычисляет первый аргумент, если первый аргумент вычисляется в значение «ИСТИНА», иначе вычисляется второй аргумент.
Вывод ответа	answer	Записывает ответ программы. «ИСТИНА» если входные графы изоморфные, «ЛОЖЬ» если не изоморфные.
Арифметические операторы	+	Складывает два действительных числа.
	*	Перемножает два действительных числа.
	/	Делит два действительных числа.
Преобразование типов	cast	Преобразует целочисленное в действительное число.
	round	Преобразует действительное в целочисленное число с помощью математического округления.
Работа с графиками	construct	Создает новое ребро графа из двух индексов: начала и конца.
	getSource	Возвращает индекс вершины, из которой выходит ребро графа.
	getDist	Возвращает индекс вершины, в которую приходит ребро графа.

Таблица 8 – Типы проблемно-ориентированного языка

Название	Описание	Пример
Double	Действительные десятичные числа	10.0, 0.42
Integer	Целочисленные десятичные числа	42, -4, 1
Boolean	Логические значения	true, false
Edge	Ребро графа	1 -> 2, 4 -> 5

7 Организационно-экономическая часть

7.1 Обоснование сметы затрат

Расчет затрат на разработку данного программного продукта осуществлялся на основе цен за период март-июнь 2014 года.

7.1.1 Расчет затрат на расходные материалы

Во время разработки дипломного проекта были произведены следующие затраты на расходные материалы:

- а) Бумага формата А4 с плотностью $80 \frac{\text{г}}{\text{м}^2}$ 500 листов – 150 рубля
- б) Печать графической части на формате А1 (11 штук) – $11 * 50 = 550$ рубля
- в) Катридж CE285A для лазерного принтера LaserJet M1132 MFP – 2670 рубля
- г) Записываемый компакт диск CD-R – 22 рубля

Итоговые затраты на расходные материалы:

$$C_m = 3392 \text{ руб.}$$

7.1.2 Расчет затрат на оборудование

В данном разделе приведены расчеты затрат, связанных с использованием вычислительной техники во время разработки, включая затраты на ремонт данной техники.

Программный продукт разрабатывался на ноутбуке Lenovo IdeaPad V570C 59319588, стоимость которого составляет 14900 рублей. Конфигурация данного ноутбука является достаточной для разработки и отладки данной системы. Для подготовки документации использовался многофункциональный лазерный принтер LaserJet M1132 MFP стоимостью 5632 рубля. Итого:

$$C_{\text{об}} = 20532 \text{ руб.}$$

Длительность использования оборудования: 6 месяцев.

Затраты на вычислительную технику высчитываются по формуле с использованием ускоренных сроков амортизации:

$$C_{\text{ЭВМ}} = k \sum_i S_i \frac{T_i}{12}$$

Где:

- k - коэффициент амортизации на год (для ускоренной амортизации $k = 0.15$)
- $\sum_i S_i$ - суммарная стоимость оборудования
- T_i - период использования оборудования в месяцах

$$C_{\text{ЭВМ}} = 0.15 * 20532 * 0.5 = 1539.90 \text{ руб.}$$

Таким образом затраты на вычислительную технику при ускоренных сроках амортизации составляют 1539.90 рублей.

Затраты на ремонт вычислительной техники составляют 10% от ее стоимости:

$$C_{\text{рем}} = 0.1 \sum_i S_i = 0.1 * 20532 = 2043.20 \text{ руб.}$$

Итоговые затраты на оборудование с учетом ремонта составляют:

$$C_{\text{ЭВМ}} + C_{\text{рем}} = 1539.90 + 2043.20 = 3583.10 \text{ руб.}$$

7.1.3 Расчет затрат на услуги сторонних организаций

В данном пункте учитываются затраты на выполнение сторонними организациями работ, непосредственно связанных с разработкой данного дипломного проекта. Затраты на услуги сторонних организаций:

- Вывод графической части на плоттере с учетом печати черновых листов
 - 1100 рублей
- Переплет программной документации – 300 рублей

Итоговая стоимость услуг сторонних организаций:

$$C_{\text{yco}} = 1400 \text{ руб.}$$

7.1.4 Расчет заработной платы

В данном пункте рассчитывается заработка плата исполнителей, непосредственно связанных с разработкой проекта, с учетом их должностного оклада и времени участия в разработке.

Затраты на выплату исполнителям заработной платы состоят из двух компонент:

$$C_{\text{зарп}} = C_{\text{з.осн}} + C_{\text{з.доп}}$$

Где:

- $C_{\text{з.осн}}$ - основная заработка плата
- $C_{\text{з.доп}}$ - дополнительная заработка плата

Расчет предварительной заработной платы $C_{\text{зп}}$ производится по следующей формуле:

$$C_{\text{зп}} = \sum_{i=1}^n C_{\text{мин3п}} K_i t_i^{\text{ПАЗР}}$$

Где:

- а) n - количество работников
- б) $C_{\min \text{ЗП}}$ - минимальная заработка плата, равная 12600 рублей для жителей Москвы
- в) K_i - коэффициент, соответствующий разряду работника, в данном случае равный 1
- г) $t_i^{\text{РАЗР}}$ - время разработки, 6 месяцев

Возможен расчет исходя из трудового договора, где прописан размер заработной платы, который в общем случае может отличаться от нормативной.

$$C_{\text{зп}} = 75600 \text{ руб.}$$

В соответствии с главой 23 НКРФ доходы физических лиц за вычетом некоторых льгот подлежат обязательному налогообложению (налог на доходы физических лиц). Для компенсации выплат размер месячного оклада увеличивается, что отражено в формуле:

$$C_{\text{з.осн}} = C_{\text{зп}}(1 + H_{\text{дфл}}) = 85428 \text{ руб.}$$

Где:

- $C_{\text{зп}}$ – сумма к выплате, которая была оговорена с работником
- $H_{\text{дфл}}$ – налог на доходы с физических лиц, 13%

7.1.5 Расходы на дополнительную заработную плату

Расходы на дополнительную заработанную плату учитывают все выплаченные непосредственно исполнителям за время, не проработанное на производстве,

но предусмотренное законодательством, в том числе: оплата очередных отпусков, компенсация за недоиспользованный отпуск, и др. Величина этих выплат составляет 20% от размера основной заработной платы:

$$C_{з.доп} = 0.2C_{з.осн} = 17085.60 \text{ руб.}$$

В результате получаем:

$$C_{зарп} = C_{з.осн} + C_{з.доп} = 102513.60 \text{ руб.}$$

7.1.6 Расчет отчислений на социальные нужды

В данном пункте учитываются отчисления на социальные нужды, производимые в фонды социального страхования, обязательного медицинского страхования и пенсионный фонд. Расчет производится в соответствии с главой 24 Налогового Кодекса РФ «Единый социальный налог (взнос)».

Ставки единого социального налога определяются в зависимости от величины налогооблагаемой базы. Например, для налоговой базы, рассчитанной для каждого отдельного работника нарастающим итогом с начала года до 624 000 руб., величина единого социального налога рассчитывается по формуле:

$$C_{есн} = K_{есн}C_{зарп}$$

Где:

- $K_{есн}$ – коэффициент отчисления на соц. нужды
- $C_{зарп}$ – заработка плата в рублях

Коэффициент отчислений на социальные нужды без учета льгот складывается из следующих отчислений:

- ФСС РФ – отчисление в фонд социального страхования составляет 2.9% от заработанной платы
- ПФР – отчисление в пенсионный фонд составляет 22%
- ФФОМС – отчисление в фонд обязательного медицинского страхования составляет 5.1%

Исходя из приведенной информации:

$$K_{\text{есн}} = 0.3$$

Отсюда:

$$C_{\text{есн}} = 30754.08 \text{ руб.}$$

Затраты с учетом отчислений на социальные нужды за все время разработки:

$$C_{\text{есн}} + C_{\text{зарп}} = 133267.68 \text{ руб.}$$

7.1.7 Расчет накладных расходов

В данном пункте учитываются затраты на общехозяйственные расходы, внепроизводственные расходы и расходы на управление.

Накладные расходы составляют 10% от суммы остальных расходов.

$$C_{\text{нр}} = 0.1(C_{\text{м}} + C_{\text{о6}} + C_{\text{уко}} + C_{\text{зарп}} + C_{\text{есн}})$$

$$C_{\text{нр}} = 0.1 * (3392 + 20532 + 1400 + 102513.60 + 30754.08) = 15859.17 \text{ руб.}$$

7.1.8 Расчет прочих расходов

В данном пункте представлены налоги на имущество и налоги на транспортные средства.

Налог на имущество в данном случае не платится, поскольку все имеющиеся в наличии имущество, включаемое в налогооблагаемую базу в соответствии с инструкцией «О порядке исчисления и уплаты в бюджет налога на имущество предприятий», используется на нужды образования, и, следовательно, налогом на имущество не облагается.

Налог на владельцев транспортных средств не платится, в связи с отсутствием транспортных средств.

7.1.9 Расчет себестоимости

Себестоимость рассчитывается как сумма по всем выше перечисленным статьям затрат и составляет:

$$S = C_{\text{нр}} + C_{\text{м}} + C_{\text{об}} + C_{\text{yco}} + C_{\text{проч}} + C_{\text{зарп}} + C_{\text{есн}} = 174450.85 \text{ руб.}$$

7.1.10 Расчет прибыли

Расчет прибыли ведется с учетом отчислений в местный бюджет и налога на прибыль. Отчисления в местный бюджет составляют 4.5% от себестоимости и равны 6978.03 рублей. Чистая прибыль составляет 10% от себестоимости и равна 17445.09 рублей. Налог на прибыль составляет 35% и равен 61057.80 рублей.

Итого:

$$P = 6978.03 + 17445.09 + 61057.80 = 85480.92 \text{ руб}$$

7.1.11 Цена (без НДС)

Цена программного продукта определяется путем суммирования прибыли и себестоимости и составляет:

$$\Pi = P + S = 174450.85 + 85480.92 = 259931.77 \text{ руб}$$

7.1.12 Цена (с НДС)

Ввиду того, что реализация товаров и услуг на территории РФ осуществляется с налогом на добавленную стоимость (18% от цены), в цену изделия включается НДС. Тогда окончательная цена разработанной системы с учетом НДС составляет:

$$\Pi_{\text{нДС}} = \Pi(1 + 0.18) = 259931.77 * 1.18 = 306719.49 \text{ руб}$$

7.2 Эргономика рабочего места и организация рабочего пространства

Говоря об эргономике в компьютерной области, можно сказать, что это достаточно молодая сфера. Бурное развитие она приобрела за последние десять лет. И по мере компьютеризации человечества она становится все более актуальной. Ведь согласитесь, что сейчас пользователи проводят за компьютерами намного больше времени, чем когда-либо. А незнание и невыполнение правил работы с ним часто оборачивается не только плохим самочувствием, но и потерей здоровья.

Трудовая активность человека во многом определяется условиями, в которых он работает. К ним прежде всего относятся рабочее пространство и рабочее место. Та часть рабочего пространства, где располагается производственное оборудование, с которым взаимодействует человек в рабочей среде, называется рабочим местом [?].

Как показали научные исследования, однообразные движения, совершающиеся в течение длительного времени, в сочетании с плохой организацией труда и рабочего места вызывают физические неудобства и наносят вред здоровью. Чаще всего возникают воспалительные заболевания сухожилий. Неправильная организация рабочего места может вызвать ненужную нагрузку на мышцы. Исследования показали, что примерно 20% нарушений, связанных с работой за компьютером, вызваны неправильной организацией рабочего места.

Хорошая организация рабочего пространства очень важна для сохранения здоровья, поэтому необходимо ответить на несколько вопросов, ответы на которые должны помочь организовать его.

7.2.1 Как будет использоваться ЭВМ

Кто будет работать за ним? Если за компьютером работает только один человек, то рабочее пространство заранее можно оптимально организовать под этого человека. И, например регулировка стула по высоте может не являться необходимостью. При работе нескольких человек за одним компьютером рабочее место должно подстраиваться под каждого человека, и чем больше различия между людьми, тем более широкий диапазон регулировки рабочего места необходим. Для того, чтобы обеспечить максимально комфортные условия каждому.

Как долго предполагается использовать компьютер в течение дня? Если компьютер используется несколько минут в день (до 30 минут), то вопросы эргономичной организации пространства не являются первостепенными. Если

компьютер используется более 1 часа, то следует уделить достаточное внимание организации рабочего места. И если компьютер используется больше 4 часов, следует максимально обдуманно организовать рабочее место.

Необходимо определить, какого типа программы будут работать на компьютере чаще всего. В зависимости от этого следует перед собой расположить, то устройства ввода, с которым приходится работать чаще всего.

Текстовые редакторы Удобное расположение клавиатуры с мышью наиболее важно. Современные клавиатуры имеют с правой стороны цифровую панель, поэтому при печатании текста, алфавитный набор клавиш нужно расположить перед собой по центру, для этого клавиатуру нужно сдвинуть немного вправо, так, чтобы клавиша с латинской буквой В попадала на центральную линию тела. Последние исследования в области эргономики показали, что идеальное положение при наборе текста, когда клавиатура находится под наклоном, такое положения может обеспечить специальный регулируемый держатель для клавиатуры.

7.2.2 Помещение и освещение

В помещении, предназначенном для работы на компьютере, должно иметься как естественное, так и искусственное освещение. Лучше всего, если окна в комнате выходят на север или северо-восток. Помещения необходимо оборудовать не только отопительными приборами, но и системами кондиционирования воздуха или эффективной вентиляцией. Стены и потолки следует окрашивать матовой краской: блестящие и тем более, зеркальные поверхности утомляют зрение и отвлекают от работы. В помещениях ежедневно должна проводиться влажная уборка.

Желательно, чтобы площадь рабочего места составляла не менее 6 квадратных метров, а объем - 20 кубических метров. Стол следует поставить сбоку от окна так, чтобы свет падал слева. Наилучшее освещение для работы с компьюте-

ром - рассеянный непрямой свет, который не дает бликов на экране. В поле зрения пользователя не должно быть резких перепадов яркости, поэтому окна желательно закрывать шторами либо жалюзи. Искусственное же освещение должно быть общим и равномерным, в то же время использование одних только настольных ламп недопустимо.

7.2.3 Организация рабочего стола

На рабочем столе должны свободно помещаться монитор, клавиатура, мышь и другое компьютерное оборудование, а также документы, книги, бумаги - все необходимые для работы вещи. Если вы хотите разместить в ряд несколько столов с мониторами, то следует поставить их таким образом, чтобы расстояние в ряду составляло не менее 2 метров, а между рядами - 1,2 метра. Врачи полагают, что при выполнении творческой работы, требующей значительного умственного напряжения или высокой концентрации внимания, рабочие места желательно изолировать друг от друга перегородками высотой 1,5-2 метра.

Помимо вышесказанного, строгие требования должны предъявляться к стулу, который просто необходим для поддержки правильной позы с учетом особенностей фигуры и изменения ее для снижения статического напряжения мышц шейно-плечевой области и спины. Желательно, чтобы стул регулировался по высоте, углам наклона сиденья и спинки, а также по расстоянию спинки от переднего края сиденья. Поверхности сиденья, спинки и подлокотников должны быть полумягкими, с покрытием, которое не скользит, не электризуется и пропускает воздух.

К сожалению, часто при работе очень мало внимания уделяется этому аспекту.

7.2.4 Правильная высота

Чтобы определить наиболее подходящую высоту стула, сядьте на него и положите руки на клавиатуру: ноги должны полностью касаться пола, бедра - находиться немного выше колен, спина - чувствовать упор, а предплечья - быть параллельными полу. Монитор следует размещать на столе прямо перед собой примерно на расстоянии вытянутой руки так, чтобы верхняя граница монитора находилась на уровне глаз или ниже не более чем на 15 сантиметров.

Правильное положение рук при работе с клавиатурой и мышью: локти располагаются параллельно поверхности стола и под прямым углом к плечу. Запястья не должны быть согнутыми, иначе возможно их повреждение. Желательно, чтобы во время работы запястья на что-нибудь опирались. Конструкция современных клавиатур и мышей предусматривает для них опору (дизайн клавиатуры и специальные коврики). Однако вы легко можете сами изготовить ее, например, взяв узкую полоску пенопласта и положив ее перед клавиатурой или мышью (однако следует учитывать, чтобы материал не вызывал чрезмерного раздражения рецепторов кожи (аллергические реакции), что может привести к возникновению заболеваний кожи). Клавиатура должна располагаться в 10-15 сантиметрах от края стола.

7.2.5 Эргономичная мебель

Разработанные медиками санитарно-гигиенические нормы должны учитываться при конструировании компьютерной и офисной мебели, а также при проектировании помещений офисов. В последнее время изделия, изготовленные с учетом требований гигиены и комфорта, часто называют эргономичными. Эргономика - наука о взаимодействии человека и машины. Сегодня одна из главных ее задач - снизить нагрузки на организм человека, связанные с работой на ком-

пьютере. Так, например, "эргономичная мышь" сконструирована таким образом, чтобы поддерживать запястье в нужном положении. Очевидно, одно из главных требований к современной компьютерной мебели - ее эргономичность.

Что представляет собой компьютерная мебель сегодня? Это чаще всего так называемая "универсальная стойка для компьютерного оборудования". Она, как правило, представляет собой подставку для монитора, "скворечник" для процессорного блока и полочку для принтера. Основные достоинства такой стойки - низкая цена и компактность, что немаловажно для небольших квартир.

В офисах компьютеры часто размещают на больших столах с выдвижной доской для клавиатуры. Монитор обычно ставят на угол, и во время работы все время приходится смотреть вправо или влево. Можно создать более удобную рабочую обстановку - соорудить Г-образный стол, то получите более удобный доступ к материалам.

Отдельной критики заслуживают выдвижные полки для клавиатуры. Как показали исследования, профессиональная болезнь машинисток (синдром запястного канала) зачастую вызвана именно этим приспособлением. Это неудивительно: высота офисного стола рассчитана на письменные работы, и клавиатура на выдвижной подставке оказывается заведомо ниже нормы.

7.2.6 Вентиляция

Рабочее место должно быть с хорошей вентиляцией. С одной стороны это важно для охлаждения разных частей компьютера, который выделяют тепло в процессе работы (системный блок, монитор, принтер и т.п.), а с другой стороны приток свежего воздуха в достаточной мере снабжает организм кислородом.

Если Вы курите, ни в коем случае не курите за компьютером, курение за компьютером только дополнительно дает нагрузку на Ваш организм. В результате курения в крови накапливается вредныйmonoоксид углерода (CO), а это снижает способность организма обеспечивать кровоснабжение мышц. Курение также

снижает прочность соединительной ткани в мышцах, увеличивая вероятность их травмирования.

7.2.7 Шум

Шум на рабочем месте может быть причиной стресса и вызывать лишнее напряжение мышц, что в свою очередь повышает утомляемость организма и снижает работоспособность. Поэтому необходимо выбирать по возможности тихое место. Используйте негромкое музыкальное сопровождение в качестве фона, для того чтобы замаскировать шум вентиляторов, винчестеров, принтера и т.п.

7.2.8 Рабочее кресло

Какой стул следует принимать на рабочем месте? Всем известно, что продолжительная сидячая работа вредна человеку, поэтому удобное рабочее кресло - это и наше здоровье, и настроение, и работоспособность, и производительность. Как говорит "всезнающая" статистика: работа на эргономически правильно сконструированных стульях по сравнению с обычными стульями:

- уменьшает число ошибок в два раза
- повышает концентрацию внимания (+ 7%)
- сохраняет активность (+ 9%)
- сохраняет позитивное самочувствие (+ 15%)
- способствует хорошему настроению (+ 10%)

Необходимо, чтобы рабочий стул свободно вращался относительно основания, регулировался по высоте и, кроме того, допускал возможность изменять угол наклона спинки (хорошо, если и сиденья тоже), а также устанавливать нужное

расстояние от спинки до переднего края сиденья. Обивка кресла должна быть не только практичной, стойкой к длительным физическим воздействиям, но и гигиеничной, т. е. выполненной из материалов, безвредных для здоровья и обеспечивающих удобство и комфорт в работе.

Идеальная высота сиденья - когда ступни ног полностью касаются пола, а угол сгиба коленей при этом составляет примерно 90 градусов. Очень важно, чтобы край сиденья имел мягкую скругленную вниз форму. Это позволяет избежать давления на кровеносные сосуды и не нарушать циркуляцию крови.

Позвоночник здорового человека напоминает знак интеграла. А, следовательно, спинке кресла необходимо иметь соответствующую форму, чтобы помочь сохранять это положение. Это очень важный момент. Если приходится сидеть на обычном стуле без выпуклости под поясницу, рекомендуется применять небольшую мягкую подушку для этих целей. Угол между спинкой кресла и сидением должен составлять чуть более 90 градусов. Иногда стулья снабжаются специальным механизмом, позволяющим одновременно менять угол наклона спинки и сиденья так, что положение позвоночника остается правильным в любой момент времени.

Хорошо, если спинка стула поддерживает лишь нижнюю половину спины, но при этом не является жестко закрепленной, чтобы не препятствовать движениям в процессе работы.

7.2.9 Как следует сидеть за ЭВМ

Осанка - это положение, которое принимает ваше тело, когда вы сидите за компьютером.

Правильная осанка необходима для профилактики заболеваний шеи, рук, ног и спины. Необходимо так организовать свое рабочее место, чтобы осанка была оптимальной.

Правильная осанка Страйтесь сидеть за компьютером на 2,5 см выше, чем обычно.

Расположите монитор прямо перед собой. Верхняя треть экрана - на уровне глаз, чтобы при работе угол наклона шеи был естественным.

Настройте высоту спинки стула таким образом, чтобы она соприкасалась с местом наибольшего изгиба спины.

Уши должны располагаться точно в плоскости плеч.

Плечи должны располагаться точно над бедрами.

Когда вы смотрите вниз, голова должна находиться точно над шеей, а не наклоняться вперед.

Опирайтесь обеими ступнями о пол или подставку для ног. Под столом должно быть достаточно просторно, чтобы Вы свободно могли вытягивать ноги.

Руки должны удобно располагаться по сторонам.

Локти согнуты и находятся примерно в 3 см от корпуса.

Запястья должны принять нейтральное положение (ни подняты, ни опущены).

Сядьте так, чтобы край стула не давил под колени.

Правильная осанка во время работы максимально разгружает мышцы и позволяет работать дольше, меньше уставая.

Но даже абсолютно правильная осанка не поможет, если весь день сидеть в одной позе. Неподвижное положение, даже абсолютно правильное, приведет к мышечной усталости.

Правильная осанка предусматривает изменение позы примерно дважды в час.

Длительное пребывание в одной и той же позе заставляет мышцы работать непрерывно без отдыха. Из-за отсутствия достаточного отдыха или изменения позы в мышцах накапливаются продукты распада, вызывающие болезненные ощущения. Для удаления продуктов распада и питания мышц необходимо адекватное кровоснабжение. Даже незначительное изменение положения тела каждые полчаса смешает нагрузку на другие мышцы, что позволяет мышцам отдыхать

и запасаться питательными веществами.

Типичные нарушения осанки во время работы **Вытягивание шеи** вперед приводит к напряжению мышц шеи, что может в свою очередь привести к сдавливанию уходящих в голову нервов, и это вызовет боль. Эта поза увеличивает нагрузку на мышцы шеи в три раза. Мышцы шеи, поддерживающие голову, удлиняются, что может приводить к сдавливанию нерва.

Эта вредная привычка (вытягивать шею) может быть обусловлена тем, что монитор отодвинут слишком далеко. В результате, работающий на компьютере вытягивает шею вперед, чтобы лучше увидеть экран [?].

Сутулые плечи вызывают сдавливание сухожилий передней поверхности плеча, что приводит к боли в области руки и плеча.

Для того чтобы в течение дня поддерживать правильную осанку, мышцам, которые за это отвечают, необходим отдых. Это - мышцы спины, шеи и живота. Они должны получать кровоснабжение, достаточное для того, чтобы обеспечивать вертикальное положение головы и прямую спину в течение дня. Сильные мышцы помогают сохранять правильную осанку в течение более длительных периодов времени и повышают продуктивность работы.

О труде и отдыхе Но и правильная осанка, и хорошая мебель не помогут [?], если весь день сидеть в одной позе. Длительное неподвижное положение приведет к мышечной усталости. Если вам приходится весь день сидеть, вставайте время от времени либо слегка изменяйте высоту кресла или крышки стола, чтобы изменить общее положение тела. Согласно требованиям, разработанным Госсанэпиднадзором, суммарное время непосредственной работы с персональным компьютером не должно превышать шести часов за смену. На протяжении рабочего дня следует устраивать перерывы продолжительностью 10 - 20 минут. Работа без перерыва за монитором компьютера не должна превышать 45 - 60 минут. Во время перерывов рекомендуется выполнять необходимые комплексы физических упражнений.

8 Заключение

В результате выполнения задания на курсовое проектирование была получен инструмент для решения актуальной задачи. Результаты работы данной АСОИ могут быть применены в различных областях, но самым интересным практическим применением является сопоставление знаний в двух разных семантических сетях. Возникновение необходимости решать задачи такого рода являлось предпосылкой к созданию данного инструментарного средства.

Хотя данный программный продукт и производит поиск алгоритмов решения задачи проверки изоморфизма ориентированных графов, система может и найти решение за приемлемые временные рамки, так как метод генетического программирования не гарантирует нахождение оптимального решения за конечное количество обработанных поколений. Также решение может быть не найдено, так как исходная задача может принадлежать к классу NP-полных задач, однако это не доказано. Главное преимущество метода генетического программирования - возможность поиска по всему пространству возможных решений, не привлекая при этом человеческие интеллектуальные ресурсы.

В ходе выполнения курсового проектирования была изучена предметная область, изучен метод генетического программирования сложных алгоритмов, разобраны нюансы разработки проблемно-ориентированных языков и интерпретаторов, углублены знания в проектировании крупных программных продуктов, создании технической документации, создании схем UML. Были использованы современные технологии такие, как: язык программирования D, операционная система семейства GNU/Linux, графический интерфейс на основе GTK+, распределенные системы контроля версий Git и система компьютерной верстки TeX.

АСОИ поиска алгоритмов распознавания изоморфизма графов с помощью генетического программирования

Актуальность темы: задача проверки отношения изоморфизма двух графов является одной из двух нерешенных вычислительных задач, для которых не найдены алгоритмы решения за полиминальное время и которые не являются при этом NP-полными задачами.

Нахождение полиминального алгоритма значительно ускорит решение таких практических задач как:

- Проверка отношения эквивалентности знаний, хранимых в семантических сетях.
- Проверка соответствия электрических схем заданному шаблону
- Оптимизация программ при компиляции и оптимизация планов выполнения SQL запросов

Перечень задач, решенных в процессе проектирования:

- Исследование предметной области проектирования
- Определение функциональных задач
- Изучение метода Генетическое программирование
- Разработка проблемно-ориентированного языка для внутреннего представления алгоритмов
- Выбор и обоснование критериев качества оценки работы АСОИ и найденных алгоритмов

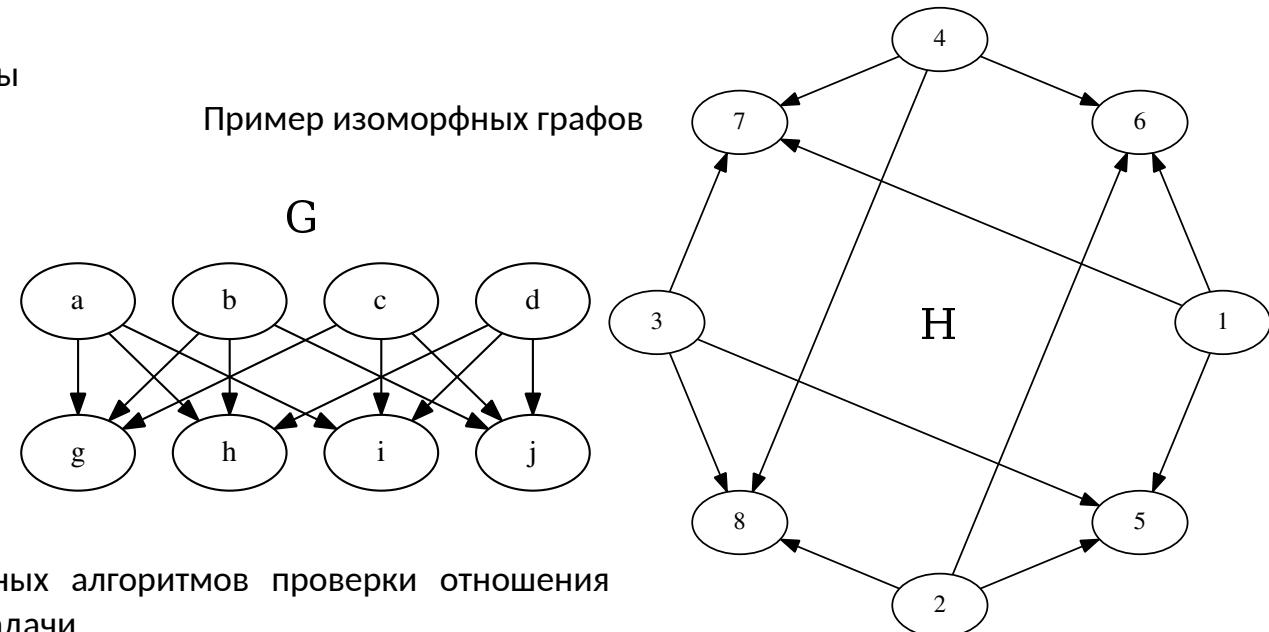
- Разработка схемы данных
- Разработка алгоритмов программы
- Разработка программы
- Отладка программы
- Разработка графического интерфейса пользователя
- Тестирование программы
- Разработка конструкторской и эксплуатационной документации

Цель: Осуществлять автоматический поиск эффективных алгоритмов проверки отношения изоморфизма среди всех возможных решений данной задачи

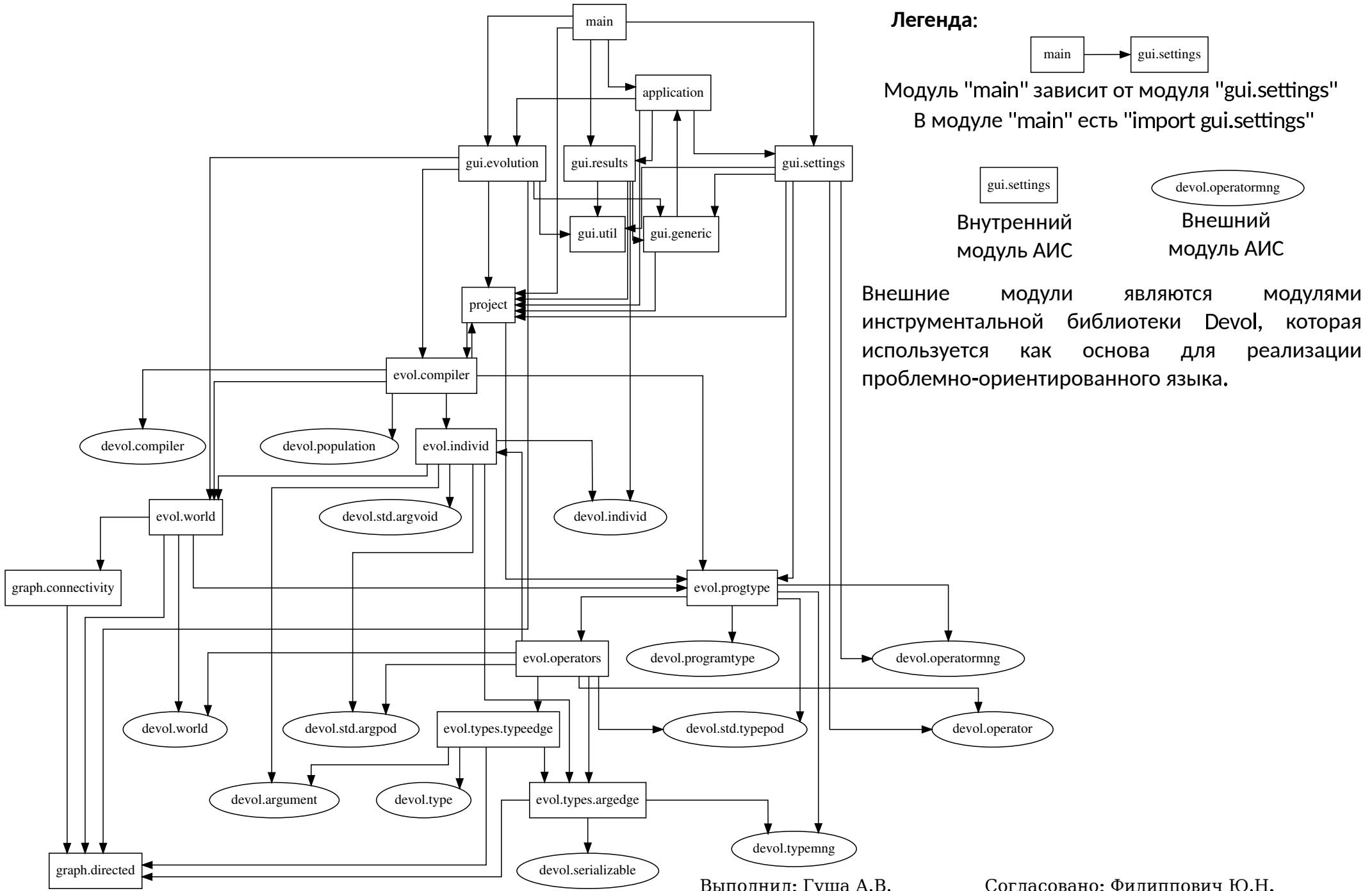
Изоморфизм - биекция между вершинами особого вида

$$f : G \leftrightarrow H \quad \begin{pmatrix} a & b & c & d & g & h & i & j \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{pmatrix}$$

Пример изоморфных графов

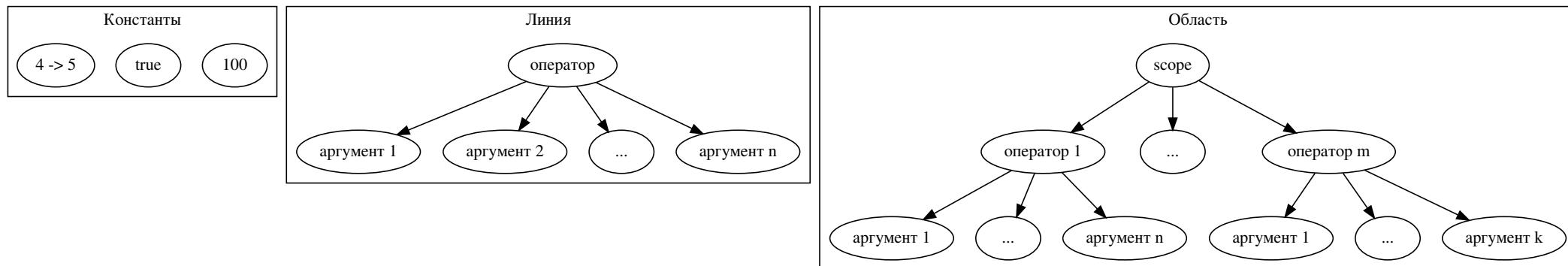


Структура модулей АСОИ



Описание проблемно-ориентированного языка генетического программирования

Программы строятся из операторов, каждый из которых имеет несколько слотов-аргументов определенного типа. Каждый аргумент может быть константой, линией (line) или областью (scope). Линии в области вычисляются последовательно.



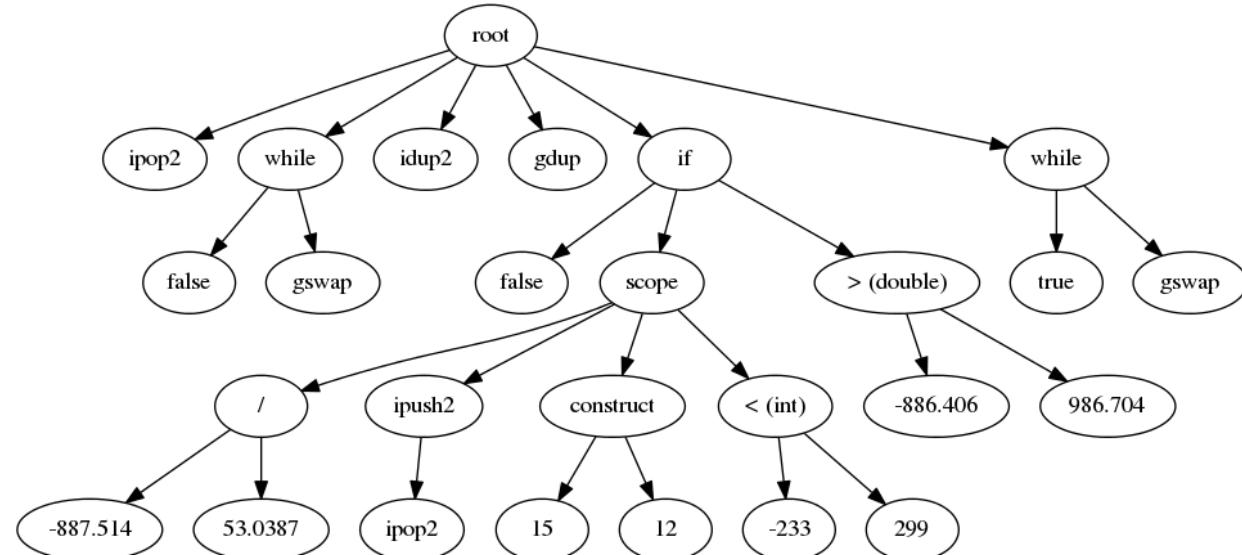
Данный язык является императивным с побочными эффектами, интерпретируется во время выполнения программы. Язык имеет строгую статическую типизацию, что необходимо для генерации изначально корректных программ.

Каждый оператор считается оператором с побочными эффектами

В данном языке определены следующие типы:

- Целочисленные числа (integer)
- Действительные числа (double)
- Логические значения (boolean)
- Пустой тип (void)
- Ребро графа вида `integer -> integer`

Пример типовой программы:

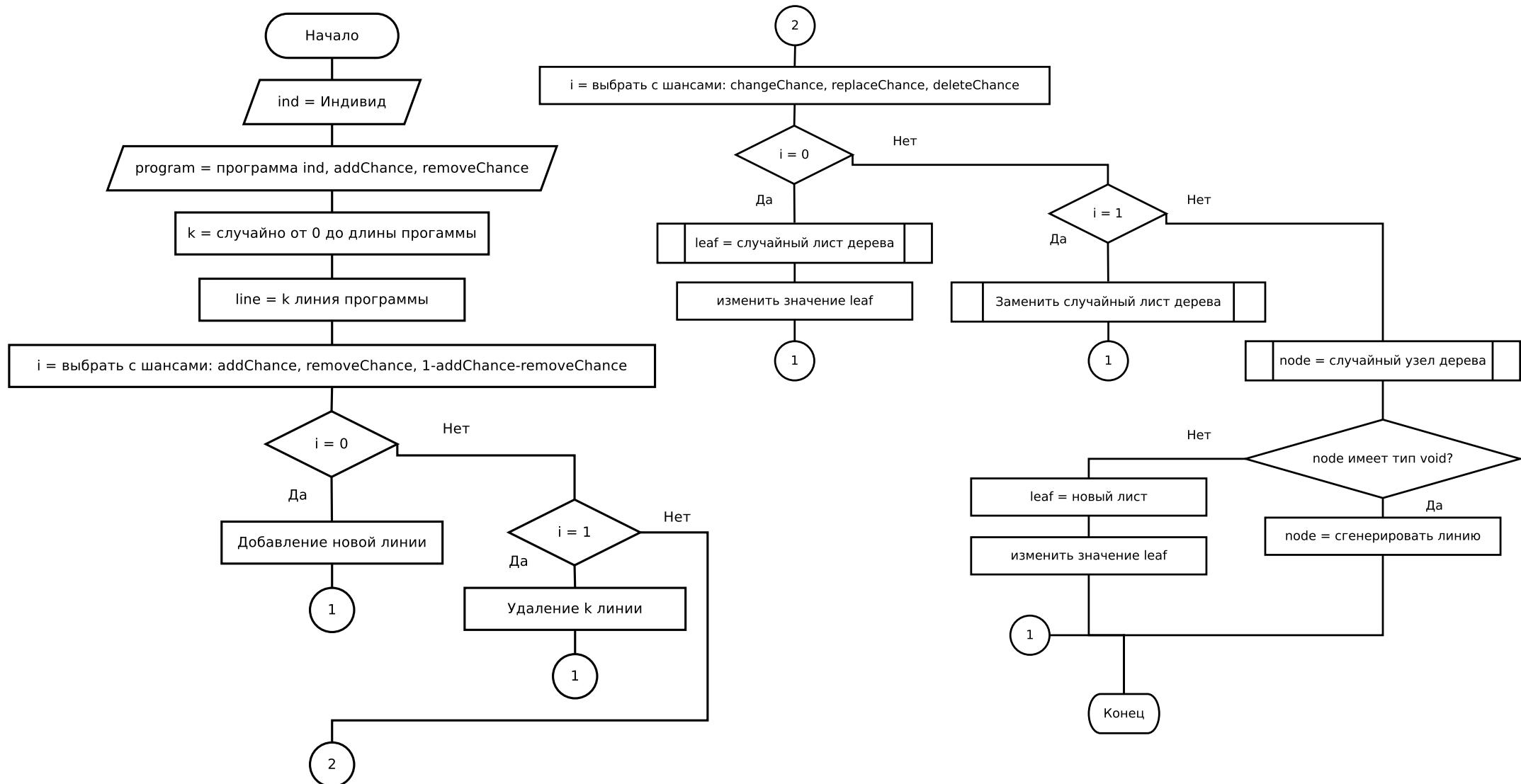


Программам доступны следующие операции:

- Условный оператор (`if`)
- Оператор цикла (`while`)
- Операторы работы со стеком общего назначения
- Операторы для работы с двумя стеками, хранящими входные графы
- Арифметические операции (`+, *, /`)
- Логические операции (`&&`, `||`, `!`)
- Операции преобразования типов
- Операции работы с ребрами графов
- Операции для записи ответа

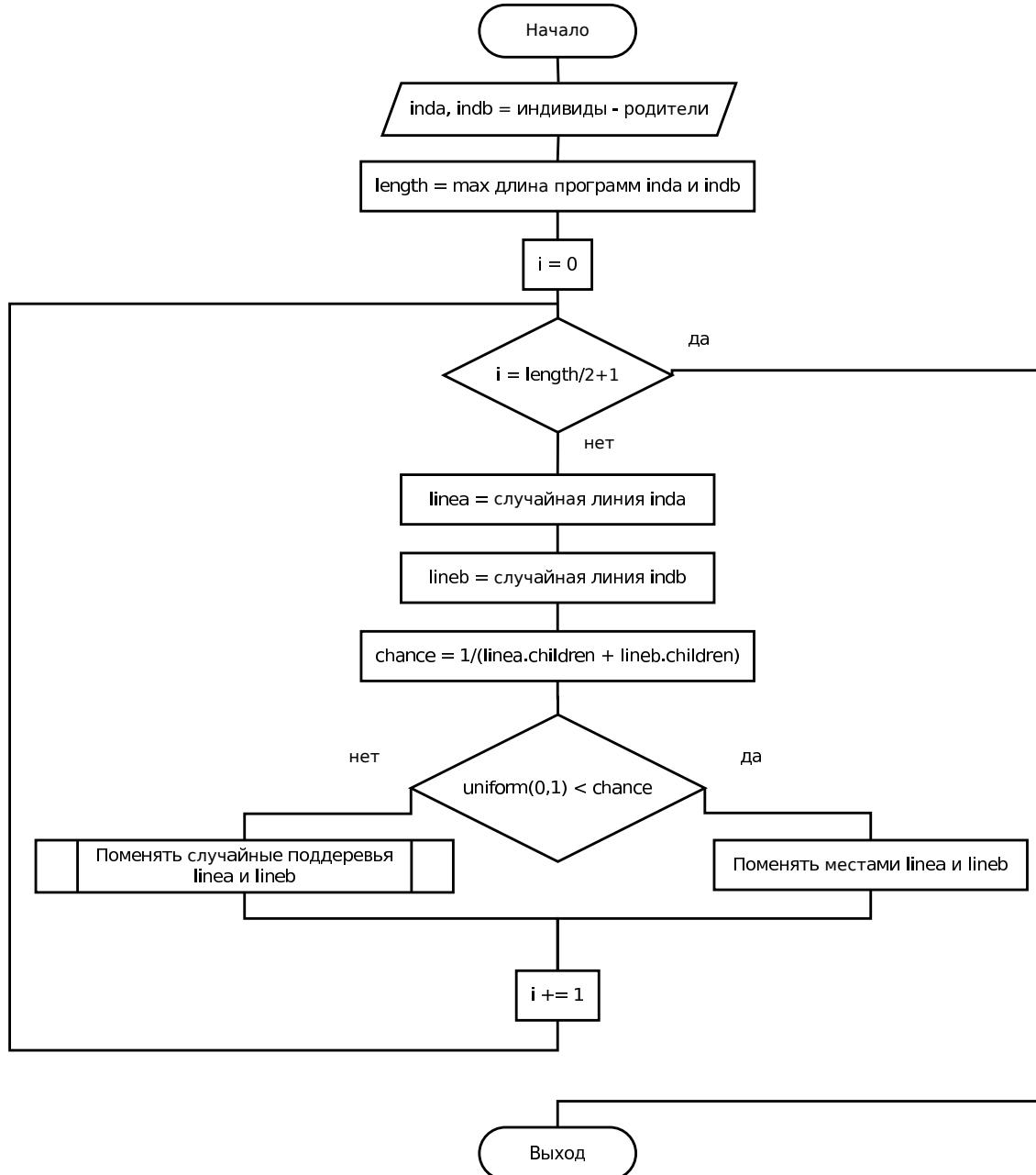
Основные алгоритмы АСОИ

Мутация



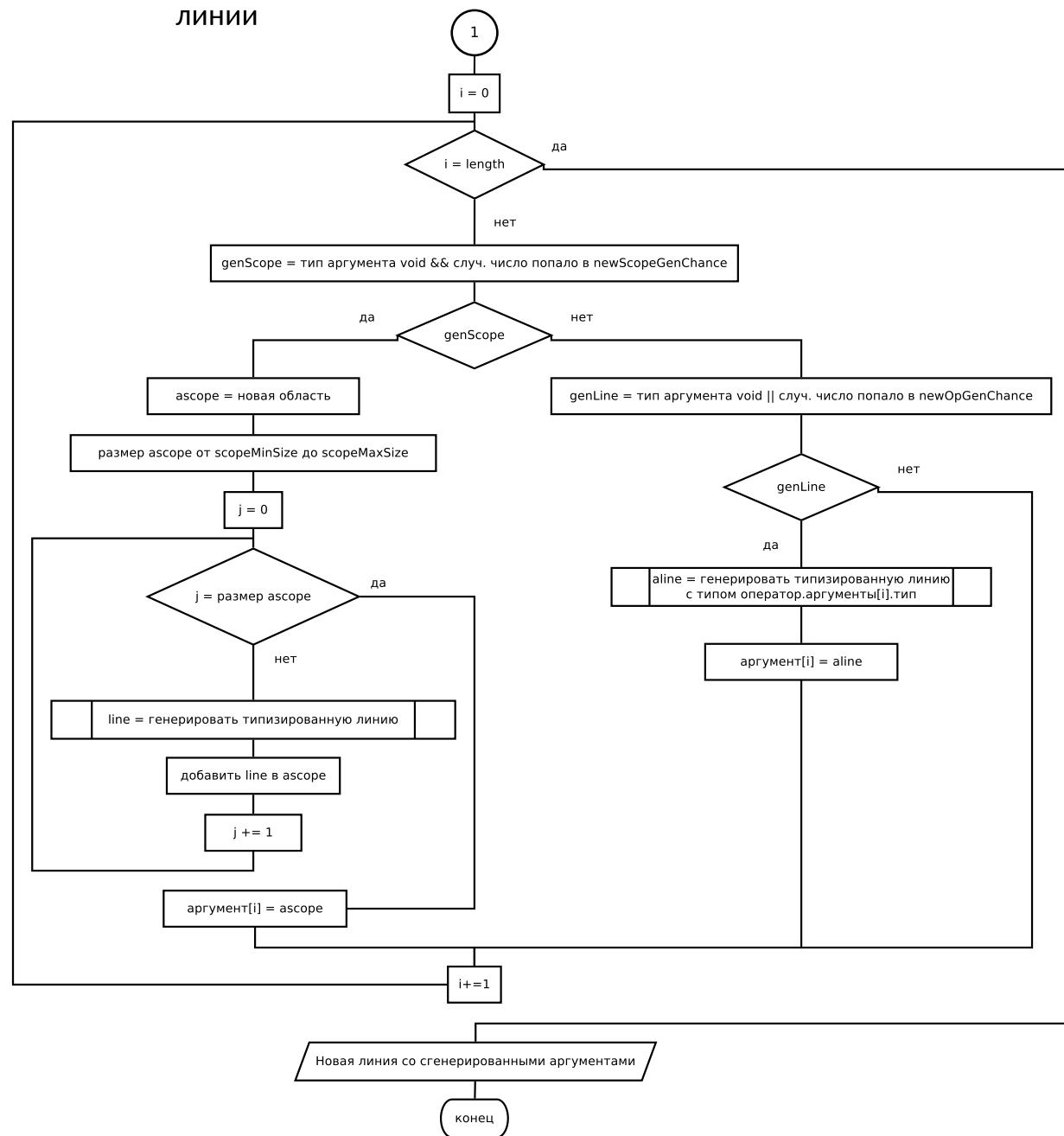
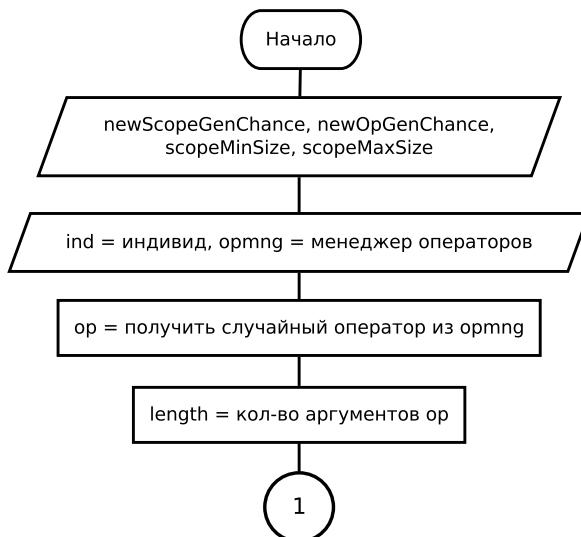
Основные алгоритмы АСОИ

Кроссинговер



Основные алгоритмы АСОИ

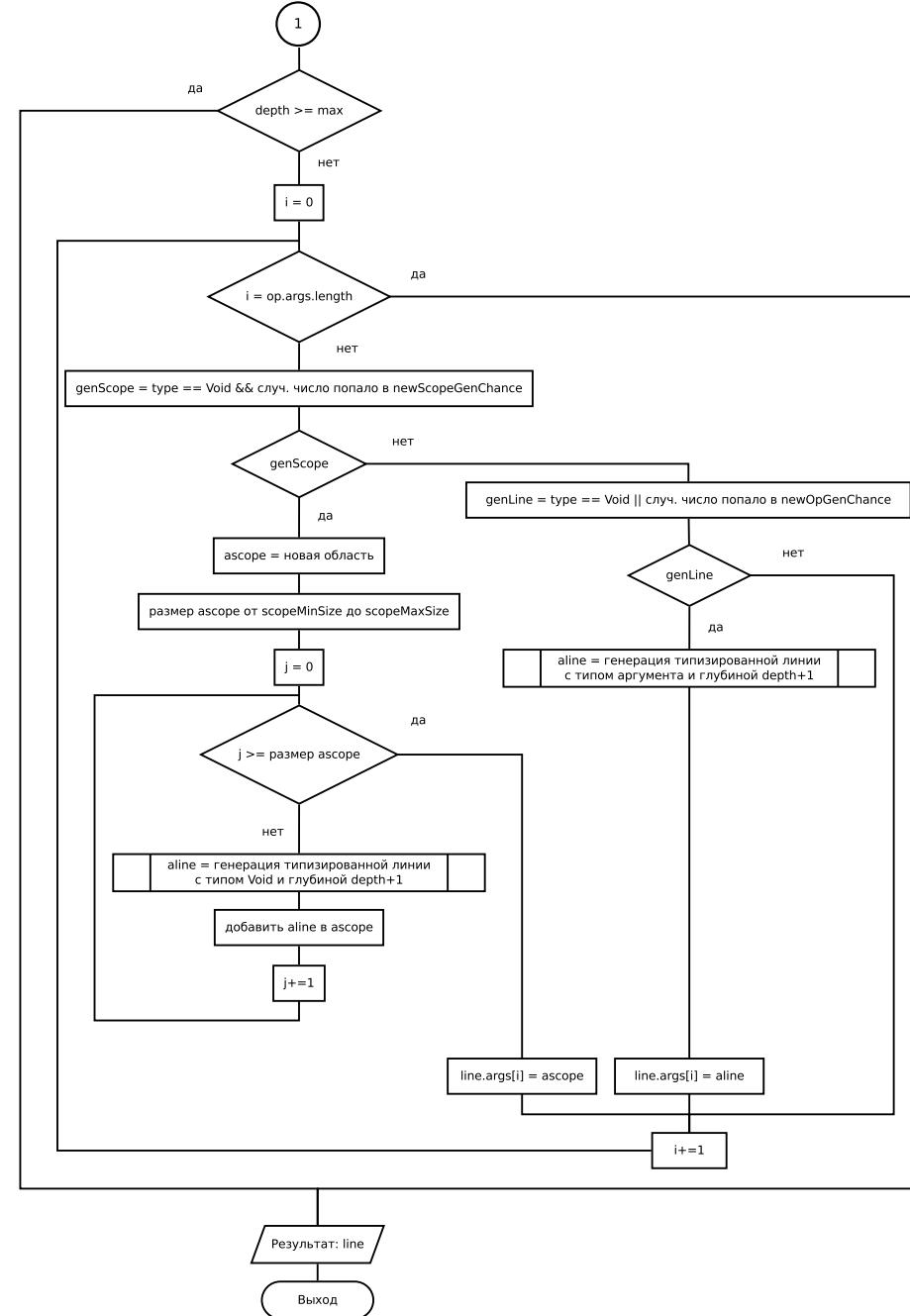
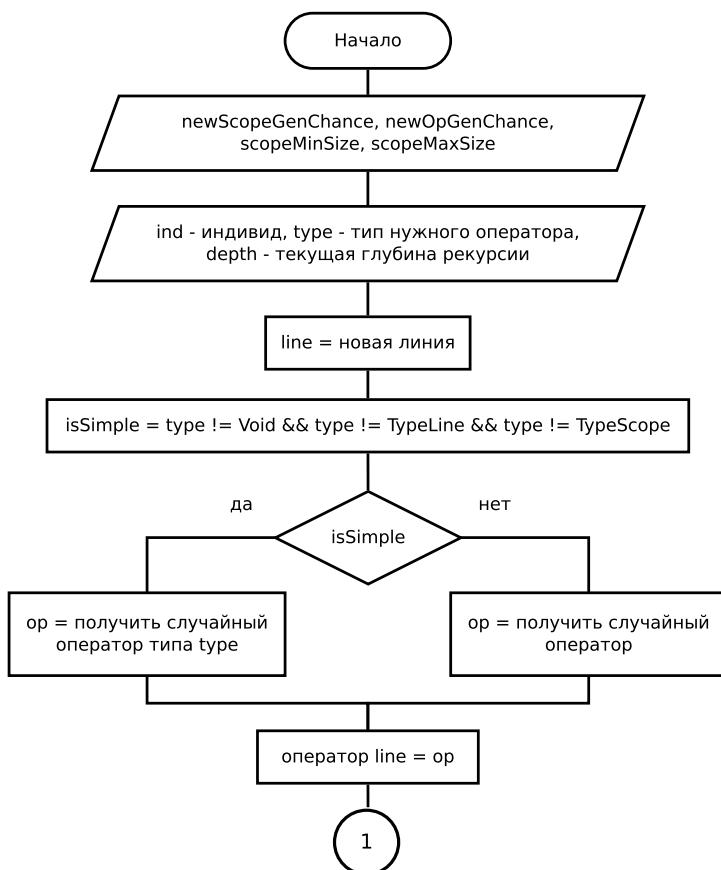
Генерация нетипизированной линии



Выполнил: Гуща А.В. _____ Согласовано: Филиппович Ю.Н. _____

Основные алгоритмы АСОИ

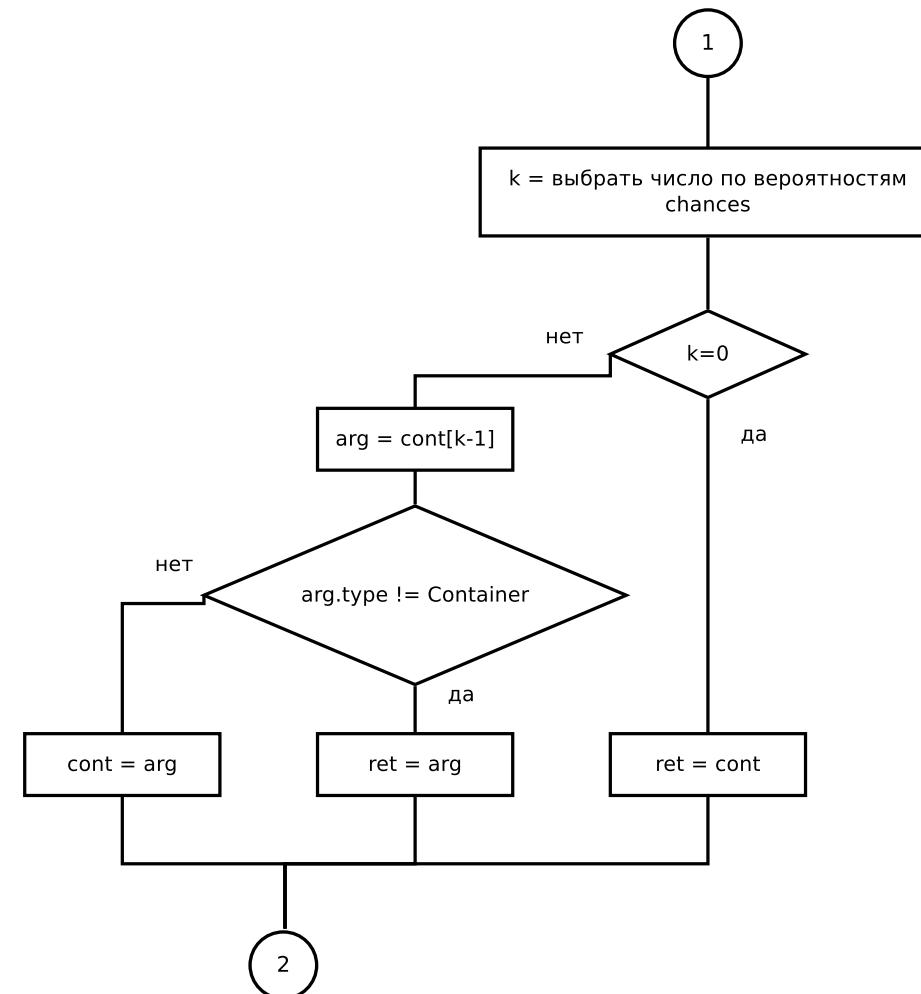
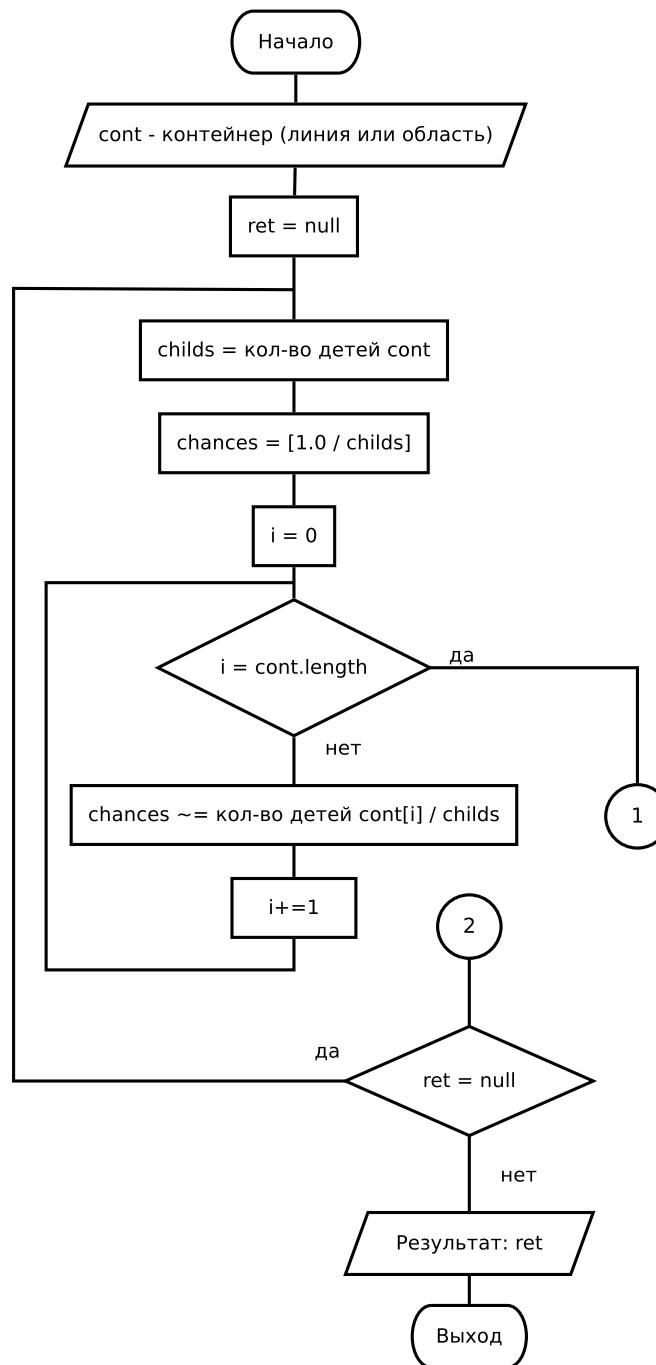
Генерация типизированной линии



Выполнил: Гуща А.В. _____ Согласовано: Филиппович Ю.Н. _____

Основные алгоритмы АСОИ

Выбор случайного узла дерева



Основные алгоритмы АСОИ

Формирование новой популяции

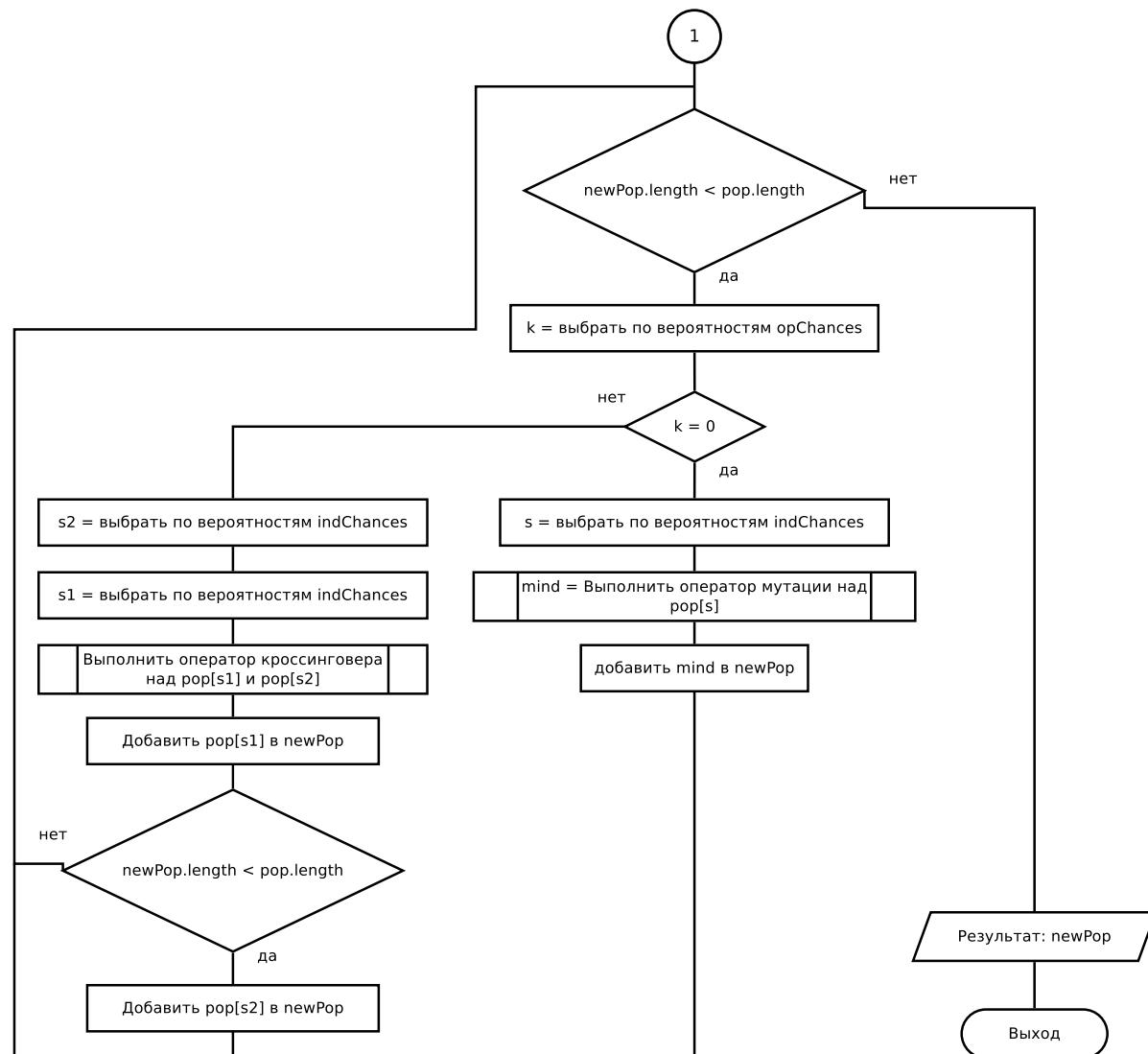
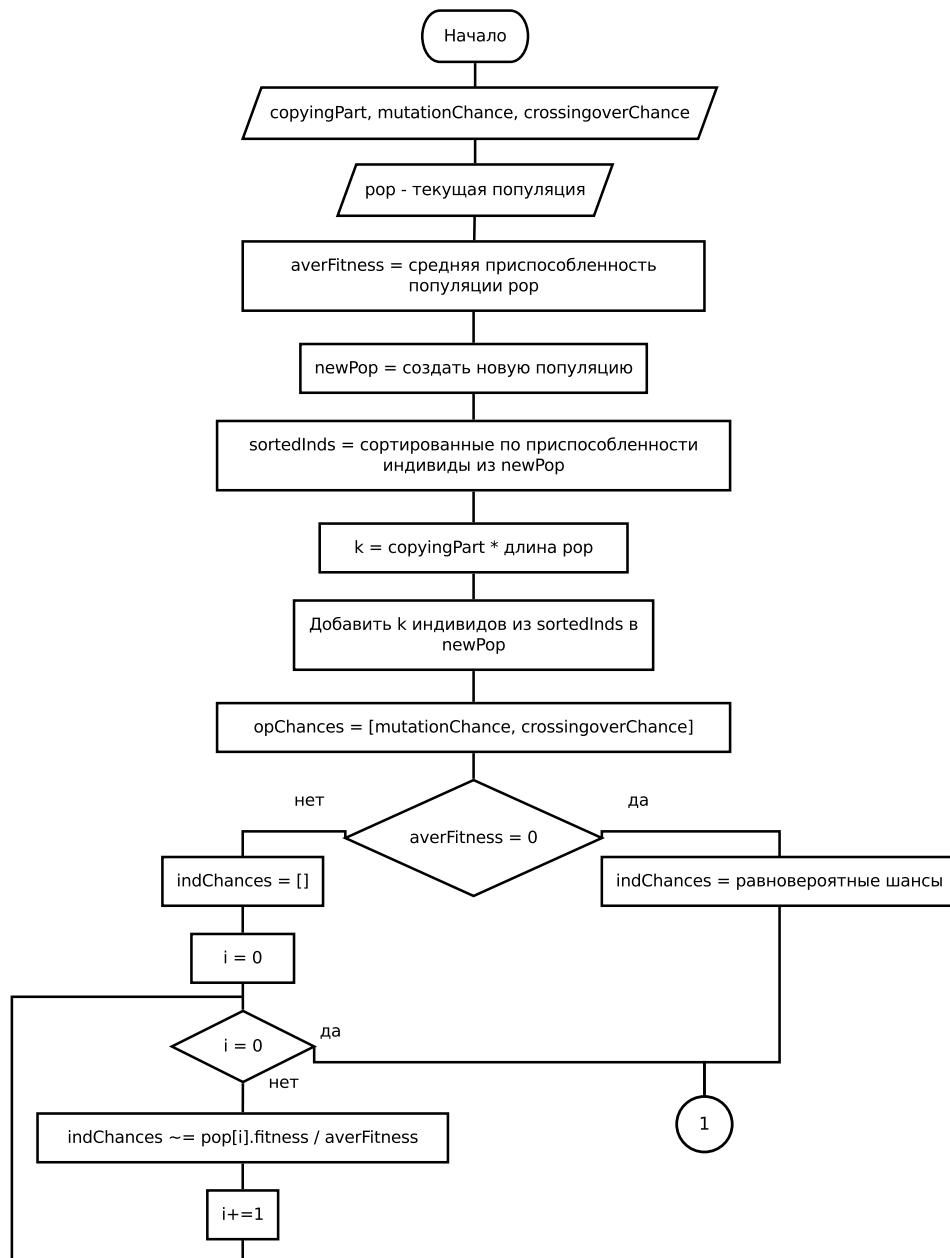
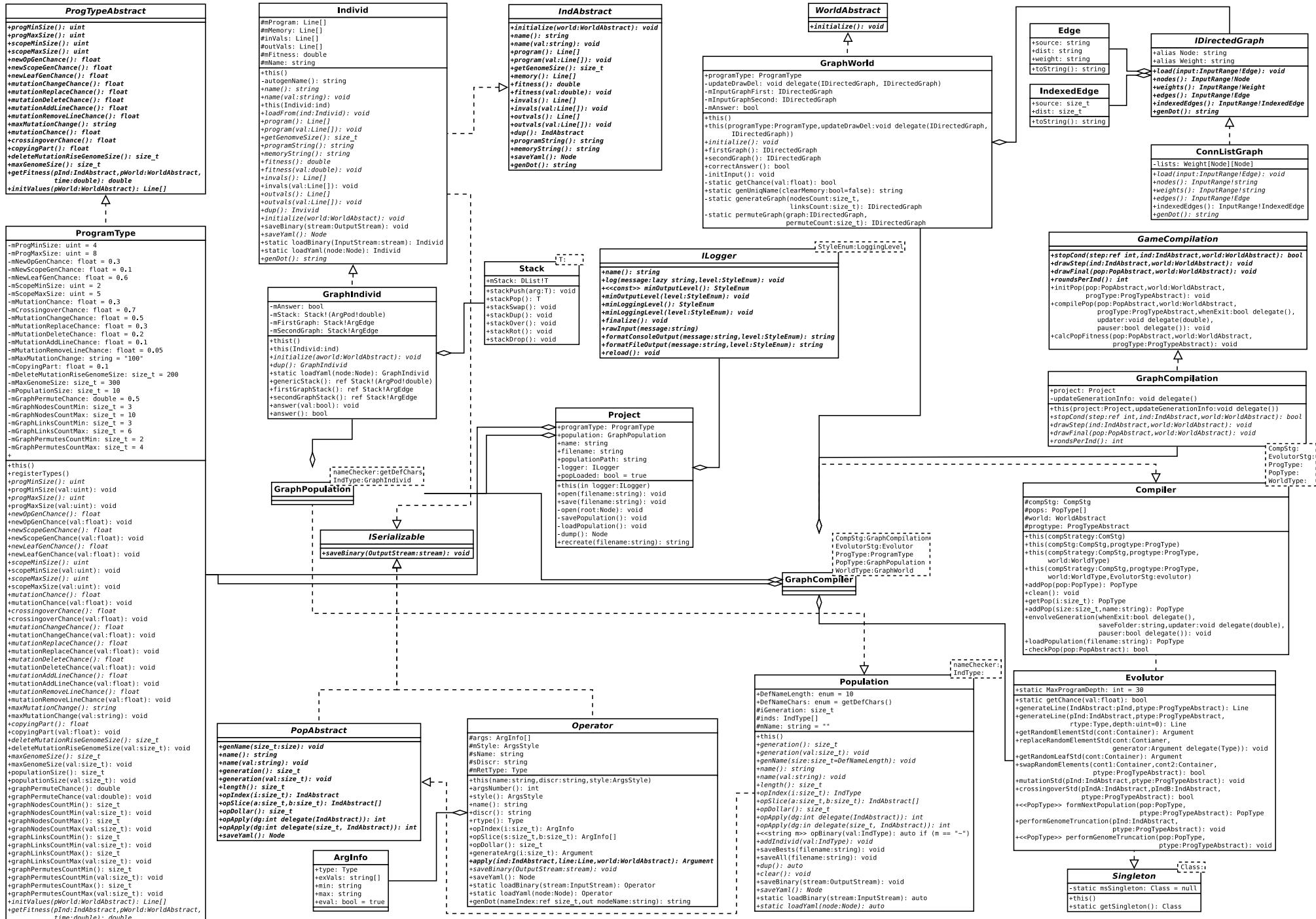


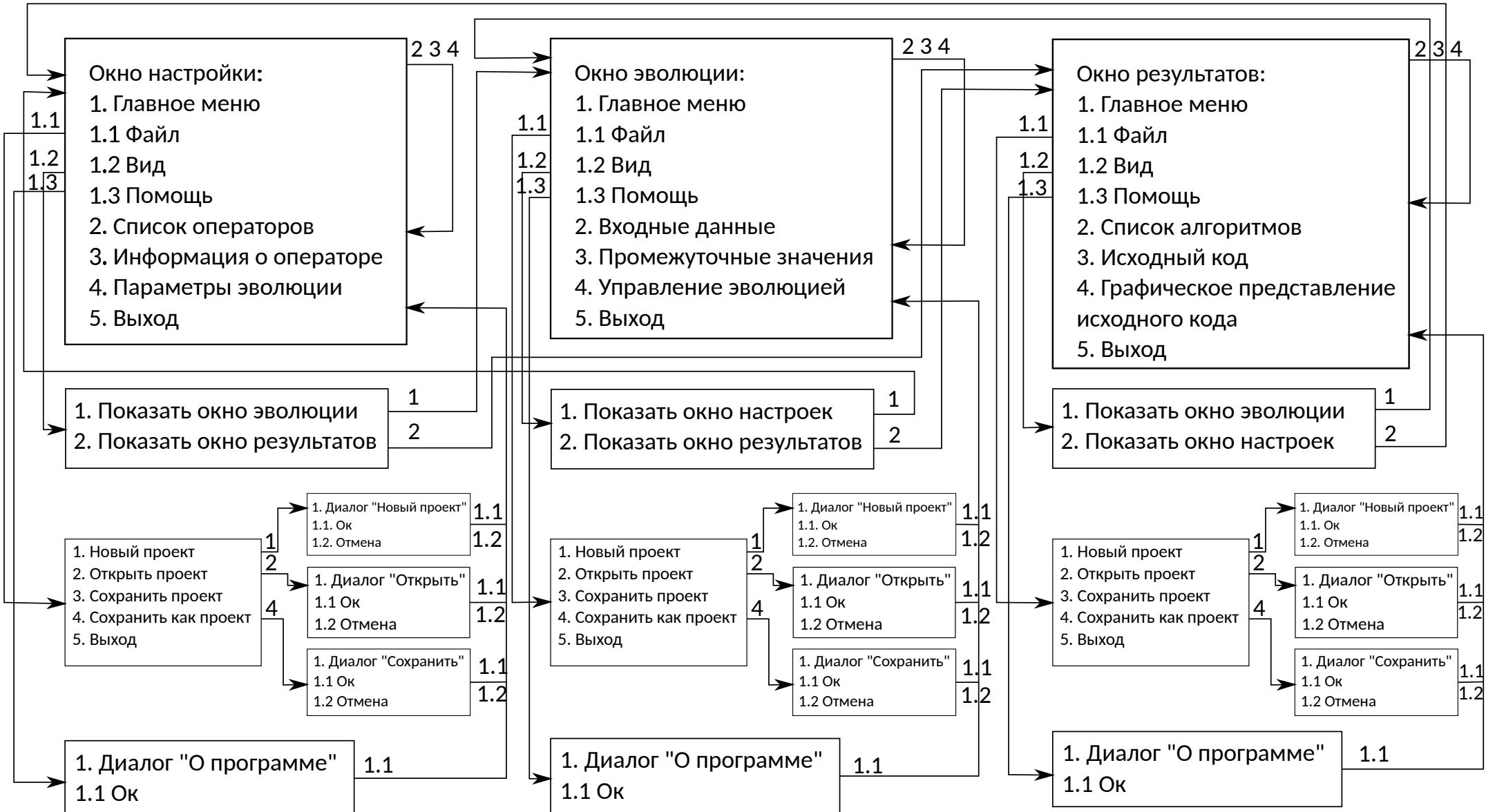
Диаграмма основных классов



Выполнил: Гуща А.В.

Согласовано: Филиппович Ю.Н.

Граф диалога с пользователем



**Федеральное государственное бюджетное образовательное
учреждение высшего профессионального образования
Московский государственный технический университет им.
Н. Э. Баумана**

23 0102

**АСОИ поиска алгоритмов распознавания изоморфизма
графов с помощью генетического программирования
Техническое задание**

Студент группы ИУ5-82

Гуща А. В

“___” _____

Содержание

1	Наименование	3
2	Основание для разработки	3
3	Исполнитель	3
4	Назначение и цель разработки	3
5	Требования к программе	4
5.1	Задачи, подлежащие решению	4
5.2	Требования к программному изделию	4
5.3	Требования к архитектуре программного изделия	5
5.4	Требования к входным и выходным данным	5
5.5	Требования к надежности	6
5.6	Требования к составу технических средств	6
6	Этапы разработки	7
7	Техническая документация, предъявляемая по окончании работы	7
8	Порядок приема работы	8
9	Дополнительные условия	8

1 Наименование

Автоматизированная информационная система поиска алгоритмов распознавания изоморфизма графов с помощью генетического программирования. Далее используется сокращение: программа.

2 Основание для разработки

Основанием для разработки является задание на курсовой проект, подписанное руководителем курсового проекта.

3 Исполнитель

Студент группы ИУ5-82 Гуща Антон Валерьевич

4 Назначение и цель разработки

Назначением разработки является предоставление пользователю инструмента поиска и анализа алгоритмов определения отношения изоморфизма ориентированных графов.

Целью разработки является нахождение алгоритмов проверки отношения изоморфизма для ориентированных графов, отбор лучших алгоритмов и анализ полученного решения. Программа должна автоматически проводить поиск и отбор алгоритмов и предоставлять пользователю графическую и текстовую информацию о промежуточных результатах для детального анализа пользователем. Автоматические процедуры, производимые программой, должны быть настраиваемыми пользователем для достижения результатов в кратчайшие сроки с желаемым качеством.

5 Требования к программе

5.1 Задачи, подлежащие решению

- а) Исследование предметной области проектирования
- б) Определение функциональных задач
- в) Изучение метода «Генетическое программирование»
- г) Разработка проблемно-ориентированного языка для внутреннего представления программ
- д) Выбор и обоснование критериев качества программы и оценки работы найденных алгоритмов
- е) Разработка схемы данных
- ж) Разработка алгоритмов программы
- з) Разработка программы
- и) Отладка программы
- к) Разработка графического интерфейса пользователя
- л) Тестирование программы
- м) Разработка конструкторской и эксплуатационной документации

5.2 Требования к программному изделию

- а) Просмотр используемого проблемно-ориентированного языка
- б) Просмотр и редактирование параметров автоматических процессов эволюции
- в) Сохранение и загрузка параметров эволюции и промежуточных результатов
- г) Управление процессом эволюции: запуск, постановка на паузу, возобновление после паузы, остановка

- д) Просмотр входных данных алгоритмов во время эволюционного процесса
- е) Просмотр промежуточных параметров эволюционного процесса
- ж) Просмотр статуса работы эволюционного процесса
- з) Просмотр промежуточных результатов после каждого эволюционного цикла:
 - Просмотр текущих алгоритмов, найденных программой
 - Просмотр значения оценки качества алгоритма
 - Просмотр исходного кода алгоритма
 - Просмотр графического изображения исходного кода алгоритма для проведения ручного анализа

5.3 Требования к архитектуре программного изделия

Программа должна работать в окружении операционной системы GNU/Linux и поддерживать вывод через графическую систему X Window System.

5.4 Требования к входным и выходным данным

- а) Все входные данные вводятся через графический интерфейс пользователя. К ним относятся:
 - 1) Параметры процесса эволюции
 - 2) Сохраненные параметры и популяции
- б) Все выходные данные должны предоставляться пользователю через графический интерфейс пользователя. К ним относятся:
 - 1) Промежуточные значения процесса эволюции: номер поколения, максимальная и средняя приспособленность.

- 2) Найденные алгоритмы, представленные в текстовой и графической форме.

5.5 Требования к надежности

Программа должна обеспечить поиск алгоритмов проверки отношения изоморфизма ориентированных графов, не должна выдавать ошибок, не предусмотренных работой программы.

5.6 Требования к составу технических средств

Программное обеспечение:

- Операционная система GNU/Linux с версией ядра не ниже 3.0
- Оконная система X Window System не ниже версии X11R7.3
- Библиотека элементов интерфейса GTK+ не ниже версии 3.10

Аппаратное обеспечение:

- Процессор, поддерживающий архитектуру x86_64 с тактовой частотой не менее 1.5 ГГц
- Оперативная память от 1 Гб
- Графический ускоритель и монитор, способные отображать графический интерфейс операционной системы
- Устройства ввода: мышь и клавиатура

6 Этапы разработки

№	Наименование этапа	Форма завершения	Срок
1	Изучение предметной области	Рабочие материалы	Январь 2014 г.
2	Разработка технического задания	Согласованное и утвержденное техническое задание	Февраль 2014 г.
3	Разработка DSL	Рабочие материалы	Март 2014 г.
4	Разработка программы	Рабочие материалы	Апрель 2014 г.
5	Тестирование программы		Апрель 2014 г.
6	Разработка документации	Техническая документация (в соответствии с п. 7 ТЗ)	Май 2014 г.
7	Сдача и приемка программы		Май 2014 г.

7 Техническая документация, предъявляемая по окончании работы

- а) Техническое задание
- б) Расчетно-пояснительная записка
- в) Текст программы
- г) Программа и методика испытаний
- д) Руководство пользователя
- е) Графические материалы (6 листов А1):
 - 1) Общие сведения о спроектированной АСОИ
 - 2) Структурная схема АСОИ

- 3) Описание проблемно-ориентированного языка генетического программирования
- 4) Основные алгоритмы обработки информации
- 5) Диаграмма основных классов АСОИ
- 6) Схема графа диалога

8 Порядок приема работы

Приемка работы осуществляется в соответствии с документом «Программа и методика испытаний».

9 Дополнительные условия

Данное Техническое Задание может дополняться и изменяться в установленном порядке.

*Федеральное государственное бюджетное образовательное
учреждение высшего профессионального образования
Московский государственный технический университет им.
Н. Э. Баумана*

23 0102

**АСОИ поиска алгоритмов распознавания изоморфизма
графов с помощью генетического программирования
Руководство пользователя**

Студент группы ИУ5-82

Гуща А. В

“___” _____

2016

Содержание

1 Назначение программы

Назначением разработки является предоставлению пользователю инструментов по автоматическому поиску алгоритмов проверки изоморфизма ориентированных графов и выдача графической информации для осуществления ручного анализа.

2 Условия выполнения программы

2.1 Требования к программным средствам

Для работы данного приложения необходимо, чтобы на компьютере были установлены следующие программные продукты:

- Операционная система семейства GNU/Linux с версией ядра не ниже 3.0
- Оконная система X Window System не ниже версии X11R7.3
- Библиотека элементов интерфейса GTK+ не ниже версии 3.10

2.2 Требования к составу технических средств

Данная программа должна работать на компьютере следующей конфигурации:

- Процессор, поддерживающий архитектуру x86_64 с тактовой частотой не менее 1.5 ГГц
- Оперативная память объемом не менее 1 Гб
- Графический ускоритель и монитор, способные отображать графический интерфейс операционной системы
- Устройства ввода: мышь и клавиатура

2.3 Требования к подготовке оператора

Для продуктивного использования данного программного продукта пользователь должен обладать следующими навыками и знаниями:

- Базовые знания английского языка, если операционная система имеет английский язык как основной
- Знания из теории графов: понятия графа, изоморфизма графов, деревья и др.
- Базовые знания информатики: алгоритм, программа, процесс интерпретации, проблемно-ориентированный язык программирования и др.
- Базовые знания эволюционных методов: функция приспособленности, популяция, индивиды, геном и др.

3 Выполнение программы

3.1 Инсталляция программного продукта

Для инсталляции программного продукта необходимо скопировать следующие файлы в инсталляционную папку:

- graph-isomorph
- gui.glade
- icon.png
- icon-small.png

3.2 Запуск программного продукт

Для запуска программы необходимо произвести двойное нажатие левой клавиши мышки на иконке файла «graph-isomorph» в графическом режиме или через терминал: перейти в инсталляционную папку и ввести:

```
./graph-isomorph
```

Возможен запуск с заранее определенным файлом проекта:

```
./graph-isomorph --proj=имя_файла_проекта
```

Для получения справки о входных параметрах приложения:

```
./graph-isomorph --help
```

Что отобразит в терминале следующее сообщение:

```
$ ./graph-isomorph --help
```

```
graph-isomorph [options]
```

```
options:  --gui=<path>  - path to glade file. Optional, default  
                  is 'gui.glade'.  
  --log=<path>   - path to log file. Optional, default  
                  is 'graph-isomorph.log'.  
  --proj=<path>  - path to project file. Optional, default  
                  is './project.yaml'.  
  --help          - display the message.
```

4 Сообщения оператору

4.1 Выход из приложения

Чтобы осуществить выход из приложения, необходимо:

- Вызвать контекстное меню «Файл» нажатием левой клавиши мыши
- Нажать на пункт «Выход» левой клавишей мыши

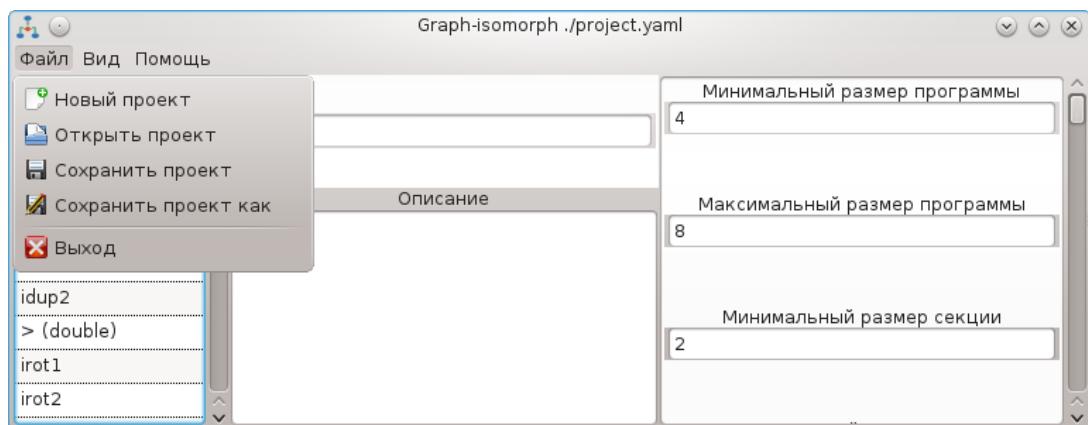


Рисунок 1 – Выход из приложения

4.2 Вызов справочной информации

Чтобы просмотреть справочную информацию, необходимо:

- Вызвать контекстное меню «Помощь» нажатием левой клавиши мыши
- Нажать на пункт «О программе» левой клавишей мыши

4.3 Операции с проектом

Вся информация, которая должна передаваться между запусками приложения, хранится в **проекте**, который сохраняется на жесткий диск в формате YAML. По умолчанию открывается проект с именем «project.yml».

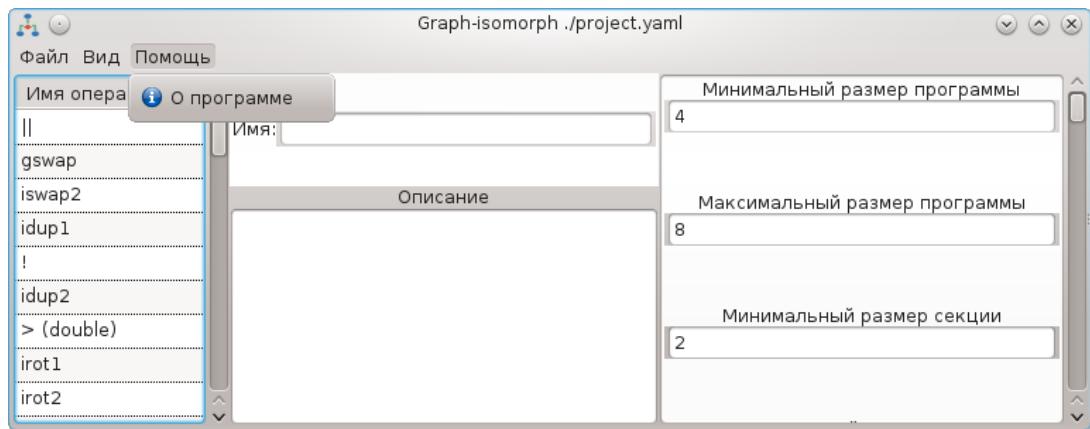


Рисунок 2 – Вызов справочной информации

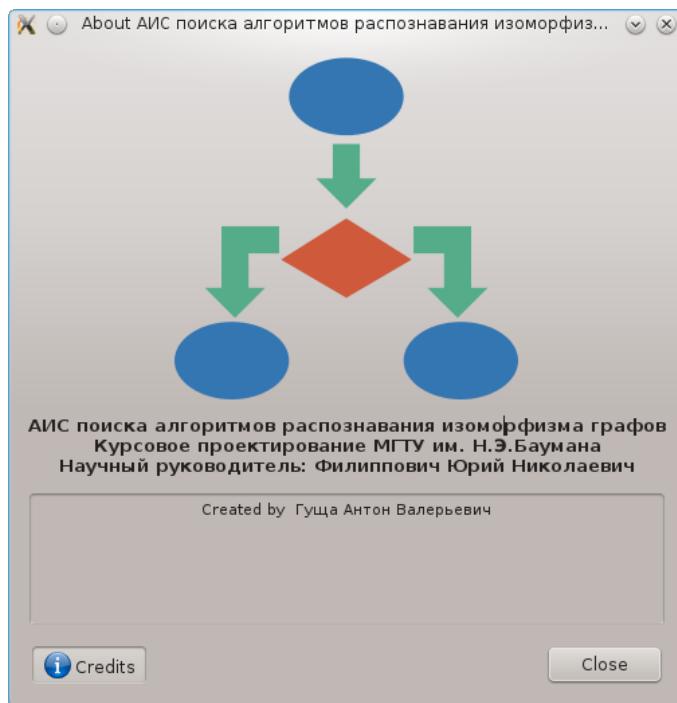


Рисунок 3 – Просмотр справочной информации

4.3.1 Создание нового проекта

Для создания нового проекта необходимо:

- Вызвать контекстное меню «Файл» нажатием левой клавиши мыши
- Нажать на пункт «Новый проект»
- В появившемся диалоге «Выберите файл нового проекта» осуществить выбор файла
- Нажать на кнопку «OK» диалога

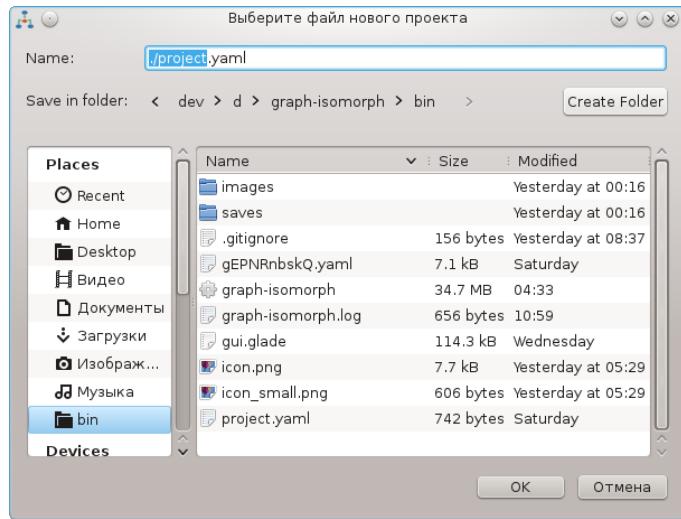


Рисунок 4 – Диалог создания нового проекта

4.3.2 Открытие проекта

Для открытия проекта необходимо:

- Вызвать контекстное меню «Файл» нажатием левой клавиши мыши
- Нажать на пункт «Открыть проект»
- В появившемся диалоге «Выберите файл проекта» осуществить выбор файла
- Нажать на кнопку «OK» диалога

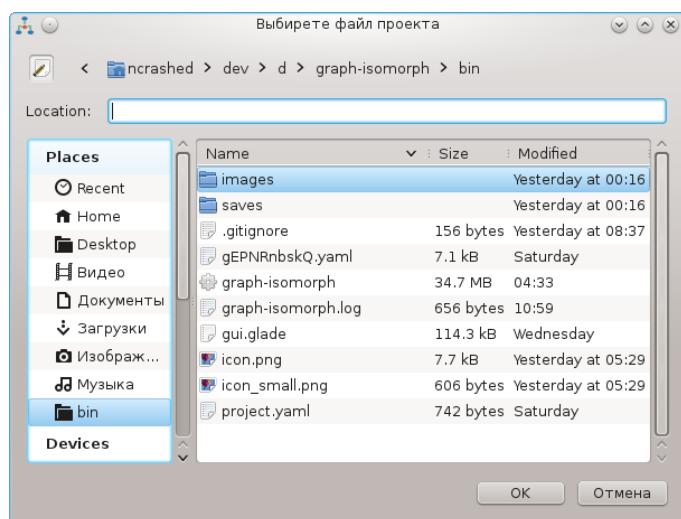


Рисунок 5 – Диалог открытия проекта

4.3.3 Сохранение проекта

Есть два способа сохранения проекта:

- Сохранение по текущему названию проекта:
 - Вызвать контекстное меню «Файл» нажатием левой клавиши мыши
 - Нажать на пункт «Сохранить проект»
- Сохранение с уточнением названия проекта:
 - Вызвать контекстное меню «Файл» нажатием левой клавиши мыши
 - Нажать на пункт «Сохранить проект как»
 - В появившемся диалоге «Выберите файл проекта» осуществить выбор файла
 - Нажать на кнопку «OK» диалога

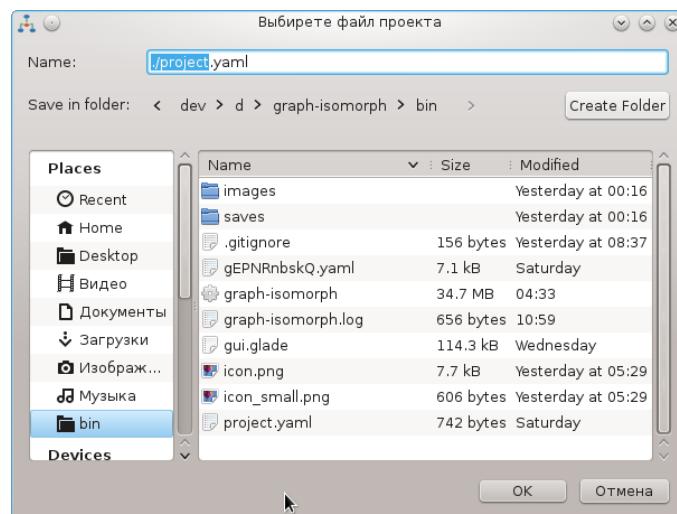


Рисунок 6 – Диалог сохранения проекта

4.4 Окно настроек

Окно настроек предназначено для:

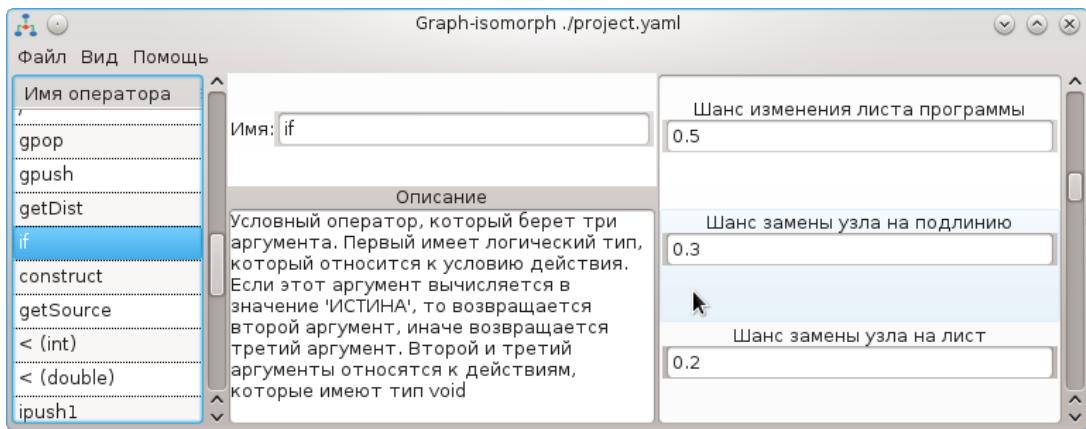


Рисунок 7 – Окно настроек

- Просмотр операторов проблемно-ориентированного языка, их названий и описания
- Просмотр и установка параметров эволюции, которые сохраняются в проекте

Данное окно видно при запуске приложения, но его возможно закрыть, тогда его можно снова активировать через другие окна путем нажатия на пункт «Показать окно настроек» меню «Вид».

4.4.1 Просмотр операторов проблемно-ориентированного языка

С левой стороны окна присутствует список операторов, зарегистрированных в проблемно-ориентированном языке представления алгоритмов проверки изоморфизма графов. При нажатии на имени оператора в центральной части появляется подробная информация о назначении оператора.

4.4.2 Просмотр и установка параметров эволюции

В правой части окна содержится список полей ввода для параметров процесса эволюции. Над каждым полем ввода имеется описание назначения параметра, в поле отображается текстовое представление параметра, которое использу-

ется АСОИ на данный момент. При некорректном вводе параметра появляется диалоговое окно с описанием проблемы и введенное значение отвергается.

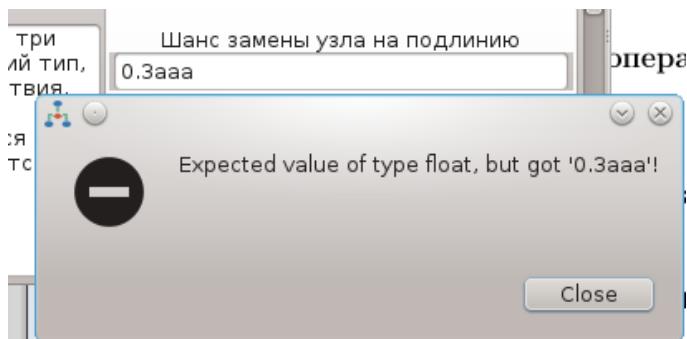


Рисунок 8 – Сообщение об ошибке распознавания входных данных

4.5 Окно эволюции

Окно эволюции предназначено для:

- Отображения процесса эволюции
- Управлением процессом эволюции
- Просмотр входных данных для алгоритмов определения изоморфизма

При запуске приложения окно эволюции не видно, для того, чтобы активировать его, необходимо нажать на пункт «Показать окно эволюции» меню «Вид».

4.5.1 Управление процессом эволюции

В нижней части окна находится три кнопки управления процессом эволюции:

- Первая кнопка с черным треугольником - запуск процесса эволюции или его возобновление
- Вторая кнопка с двумя вертикальными прямоугольниками - пауза процесса эволюции

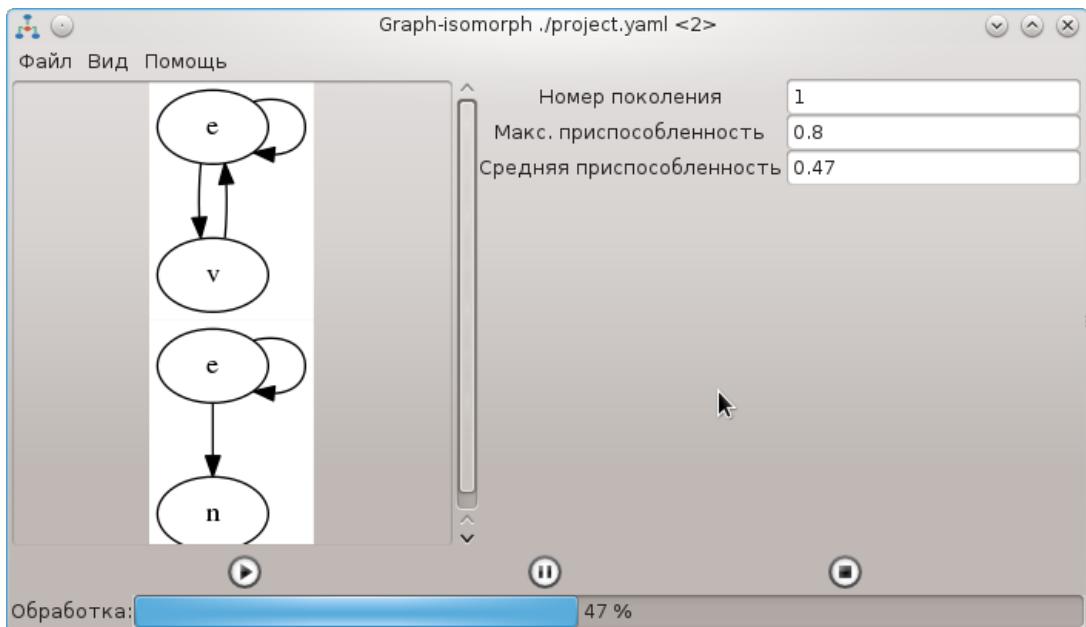


Рисунок 9 – Окно эволюции

- Третья кнопка с черным прямоугольником - остановка процесса эволюции, при нажатии на эту кнопку будет создана новая популяция и процесс эволюции начнется с самого начала.

Также предусмотрен горизонтальный индикатор процесса обработки одного поколения. Его заполненность характеризует завершенность процессов определения значений приспособленности алгоритмов и генерации следующего поколения.

4.5.2 Просмотр входных данных

В левой части окна находится область отображения входных графов, которые подаются на вход проверяемых алгоритмов. Первый график находится выше второго. Имеются полосы прокрутки для просмотра больших изображений графов.

4.5.3 Просмотр промежуточных значений

В правой части окна находятся поля ввода без возможности редактирования, в которых отображаются:

- Номер текущего поколения
- Максимальное значение функции приспособленности
- Среднее значение функции приспособленности

4.6 Окно результатов

Окно результатов предназначено для:

- Просмотра состава популяции
- Просмотра исходного кода алгоритмов
- Просмотра графической формы алгоритма

При запуске приложения окно результатов не видно, для того, чтобы активировать его, необходимо нажать на пункт «Показать окно результатов» меню «Вид».

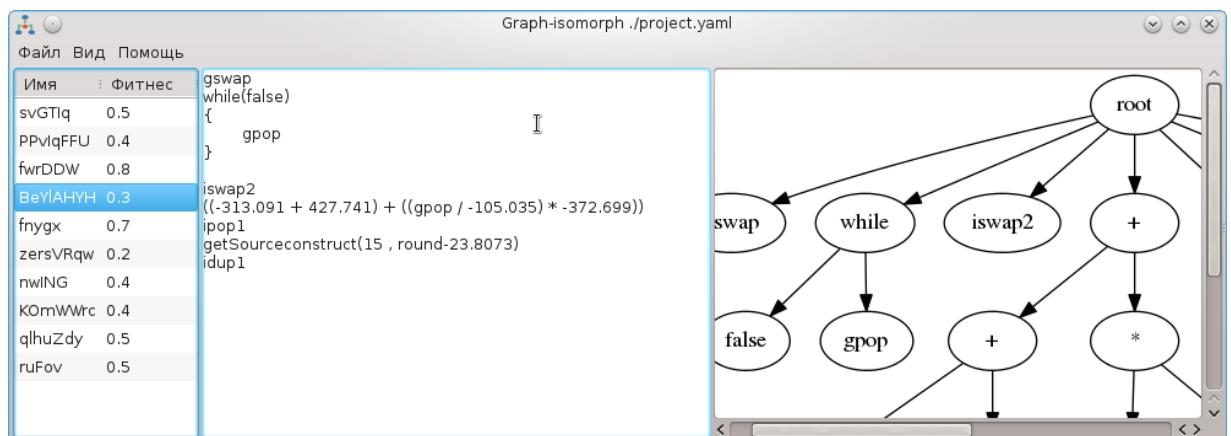


Рисунок 10 – Окно результатов

4.7 Просмотр состава популяции

В левой части окна присутствует список индивидов, которые составляют текущую популяцию. В элементах списка отображается имя индивида (случайно сгенерированная строка) и значение функции приспособленности.

4.7.1 Просмотр исходного кода алгоритмов

При нажатии на элемент списка индивидов в центральной области окна отображается текстовое представление генома выбранного индивида. Данный исходный код является псевдокодом с Си-подобным синтаксисом. Для длинных исходных кодов присутствуют полосы прокрутки.

4.7.2 Просмотр графической формы алгоритма

При нажатии на элемент списка индивидов в правой области окна отображается графическое представление генома выбранного индивида в виде дерева с переменной арностью. Для крупных изображений деревьев присутствуют полосы прокрутки.

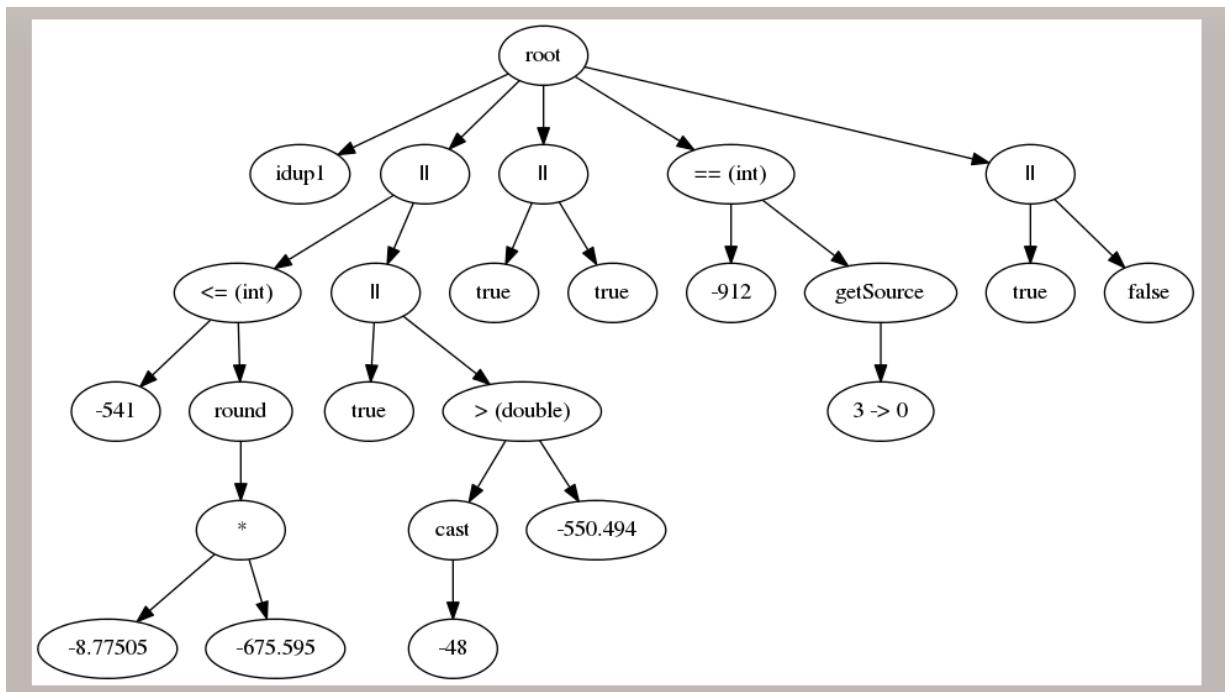


Рисунок 11 – Просмотр графического представления генома индивида

**Федеральное государственное бюджетное образовательное
учреждение высшего профессионального образования
Московский государственный технический университет им.
Н. Э. Баумана**

23 0102

**АСОИ поиска алгоритмов распознавания изоморфизма
графов с помощью генетического программирования
Программа и методика испытаний**

Студент группы ИУ5-82

Гуща А. В

“___” _____

2016

1 Аннотация

В данном документе описываются последовательность и методы проведения испытаний при тестировании программного изделия, состав и структура технических и программных средств, необходимых для проведения испытаний, а также приводятся требования к предъявляемой документации, характеристикам программы применительно к условиям эксплуатации и требования к информационной и программной совместимости.

Содержание

2 Объект испытаний

Объектом испытаний является АСОИ поиска алгоритмов распознавания изоморфизма графов с помощью генетического программирования. Сокращенное название: graph-isomorph, программа, АСОИ.

3 Цель испытаний

Цель испытаний состоит в проверке работоспособности АСОИ и соответствия выполняемых функций требованиям документа «Техническое задание».

4 Состав предъявляемой документации

На испытания программного продукта предъявляются следующие документы:

- Техническое задание
- Программа и методика испытаний
- Руководство пользователя

5 Технические требования

5.1 Требования к программной документации

Состав программной документации должен удовлетворять требованиям документа «Техническое задание», предъявляемого по окончании работы.

Программная документация должна быть оформлена в соответствии с ГОСТ и ЕСПД по составлению и оформлению документов на программное изделие.

5.2 Требования к техническим характеристикам

5.2.1 Требования к составу аппаратного обеспечения

Данная программа должна работать на компьютере следующей конфигурации:

- Процессор, поддерживающий архитектуру x86_64 с тактовой частотой не менее 1.5 ГГц
- Оперативная память объемом не менее 1 Гб
- Графический ускоритель и монитор, способные отображать графический интерфейс операционной системы
- Устройства ввода: мышь и клавиатура

5.2.2 Требования к составу программного обеспечения

Для работы данного приложения необходимо, чтобы на компьютере были установлены следующие программные продукты:

- Операционная система семейства GNU/Linux с версией ядра не ниже 3.0
- Оконная система X Window System не ниже версии X11R7.3
- Библиотека элементов интерфейса GTK+ не ниже версии 3.10

5.2.3 Требования к квалификации оператора

Для продуктивного использования данного программного продукта пользователь должен обладать следующими навыками и знаниями:

- Базовые знания английского языка, если операционная система имеет английский язык как основной

- Знания из теории графов: понятия графа, изоморфизма графов, деревья и др.
- Базовые знания информатики: алгоритм, программа, процесс интерпретации, проблемно-ориентированный язык программирования и др.
- Базовые знания эволюционных методов: функция приспособленности, популяция, индивиды, геном и др.

6 Состав и порядок испытаний

Испытания данного программного продукта должны проводиться в следующем порядке:

- Инсталляция АСОИ. Установка программного изделия осуществляется в соответствии с руководством пользователя (см. документ руководство пользователя, пункт 3.1)
- Запуск АСОИ. Запуск программного изделия осуществляется в соответствии с руководством пользователя (см. документ руководство пользователя, пункт 3.2)
- Тестирование базовых операций АСОИ

6.1 Требования к составу аппаратного обеспечения

Требования к составу аппаратного обеспечения учитываются согласно документу программа и методика испытаний, пункт 5.2.1.

6.2 Требования к составу программного обеспечения

Требования к составу программного обеспечения учитываются согласно документу программа и методика испытаний, пункт 5.2.2.

7 Методы испытаний

№	Пункт ТЗ	Выполняемые действия	Результат
1	5.2.а	Окно настроек: выбрать оператор в списке в левой части окна.	Отображение имени и описание оператора в центральной части окна (См приложение рис. ??).
2	5.2.б	1. Окно настроек: выбрать поле ввода параметра в правой части окна из списка. Просмотреть текущее значение параметра.	Подсвідка поля ввода (рис. ??)
		2. Ввести новое значение параметра. Перевести фокус на другое поле ввода.	В случае успеха значение поля изменится, в случае ошибки проверки входных данных будет отображен диалоговое сообщение с текстом ошибки (рис. ??).
3	5.2.в	Любое окно: 1. Активировать в главном меню «Файл» пункт «Сохранить проект как»	Откроется диалоговое окно сохранения проекта (рис.??).
		2. Выбрать название файла. Нажать «OK». Просмотреть папку с файлом.	По абсолютному пути сохранения проекта появился выбранный файл (рис. ??).
		3. Активировать в главном меню «Файл» пункт «Открыть проект»	Откроется диалоговое окно открытия проекта (рис. ??).

№	Пункт ТЗ	Выполняемые действия	Результат
		4. Выбрать сохраненный ранее файл проекта и нажать на «OK». Проверить загруженные данные.	Параметры эволюции и популяция обновятся в соответствии с загруженными данными.
4	5.2.г	1. Открыть окно эволюции: активировать меню «Вид» пункт «Показать окно эволюции»	На экране появится окно эволюции (рис. ??)
		2. Нажать на кнопку запуска эволюции (кнопка с черным треугольником).	Изображения входных графов начинают меняться, статус процесса в нижнем горизонтальном индикаторе выполнения увеличивается (см. рис. ??).
		3. Нажать на кнопку паузы эволюции (кнопка с двумя вертикальными прямоугольниками).	Изображения входных графов больше не меняются, статус процесса в нижнем горизонтальном индикаторе выполнения не увеличивается (см. рис. ??).
		4. Нажать на кнопку запуска эволюции.	Изображения входных графов начинают меняться, статус процесса в нижнем горизонтальном индикаторе выполнения увеличивается (см. рис. ??).
		5. Нажать на кнопку остановки эволюции (кнопка с черным квадратом).	Статус процесса сбрасывается в ноль процентов (см. рис. ??).
5	5.2.д	1. Открыть окно эволюции: активировать меню «Вид» пункт «Показать окно эволюции»	На экране появится окно эволюции (рис. ??)

№	Пункт ТЗ	Выполняемые действия	Результат
		<p>2. Нажать на кнопку запуска эволюции.</p> <p>3. Нажать на кнопку паузы эволюции. Проанализировать изображения входных графов.</p>	<p>Изображения входных графов начинают меняться (см. рис. ??).</p> <p>Изображения входных графов больше не меняются, можно проводить анализ входных данных (см. рис. ??).</p>
6	5.2.е	<p>1. Открыть окно эволюции: активировать меню «Вид» пункт «Показать окно эволюции»</p> <p>2. Нажать на кнопку запуска эволюции.</p> <p>3. Ждать, пока не обрабатывается одно поколение (индикатор прогресса дойдет до 100% и сбросится в 0%). Нажать на кнопку паузы эволюции. Проанализировать промежуточные результаты.</p>	<p>На экране появится окно эволюции (рис. ??)</p> <p>Изображения входных графов начинают меняться (см. рис. ??).</p> <p>Значения в полях промежуточных значений обновились (см. рис. ??).</p>
7	5.2.ж	<p>1. Открыть окно эволюции: активировать меню «Вид» пункт «Показать окно эволюции»</p> <p>2. Нажать на кнопку запуска эволюции.</p>	<p>На экране появится окно эволюции (рис. ??)</p> <p>Статус процесса в нижнем горизонтальном индикаторе выполнения увеличивается (см. рис. ??).</p>

№	Пункт ТЗ	Выполняемые действия	Результат
8	5.2.3	<p>1. Выполнить действия в пункте №5. Открыть окно результатов: активировать меню «Вид» пункт «Показать окно результатов»</p> <p>2. Просмотреть список алгоритмов-индивидов в левой части окна, выбрать одного индивида щелчком по его имени.</p>	<p>На экране появится окно результатов (рис. ??)</p> <p>Центральная и правая часть окна обновятся и будут отображать текстовое и графическое представление исходного кода соответственно (см. рис. ?? и рис. ??)</p>

8 Приложение

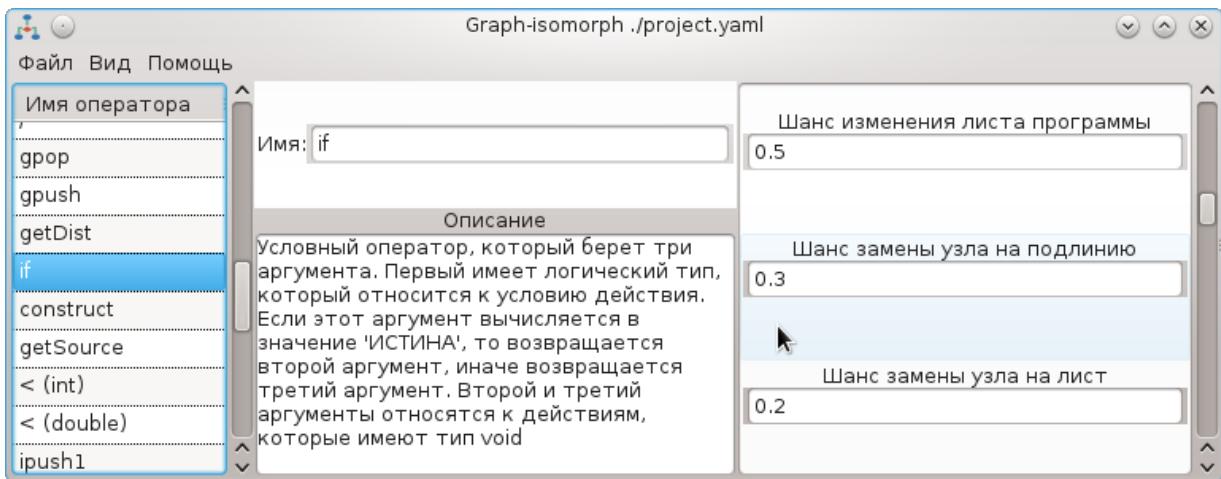


Рисунок 1 – Окно настроек

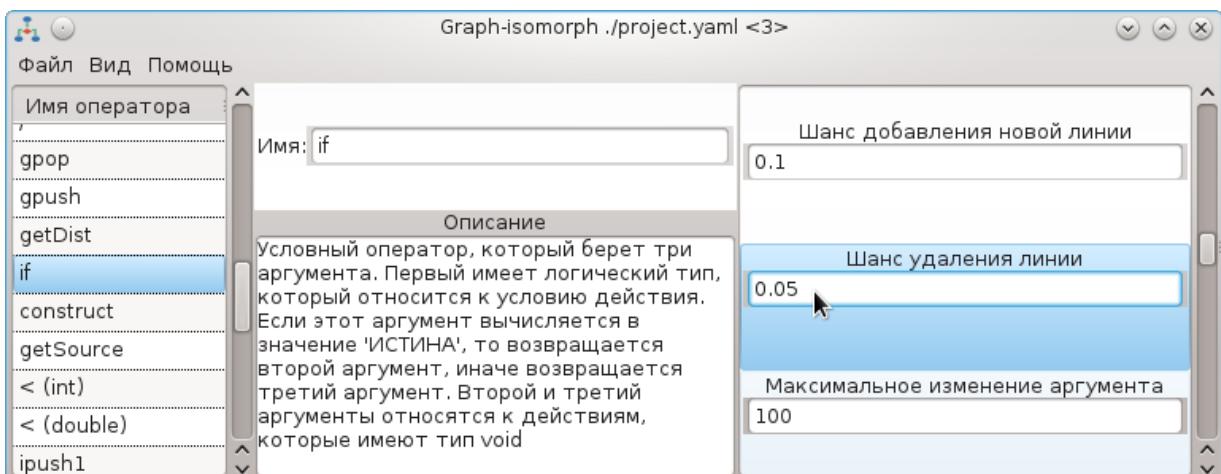


Рисунок 2 – Просмотр параметра эволюции

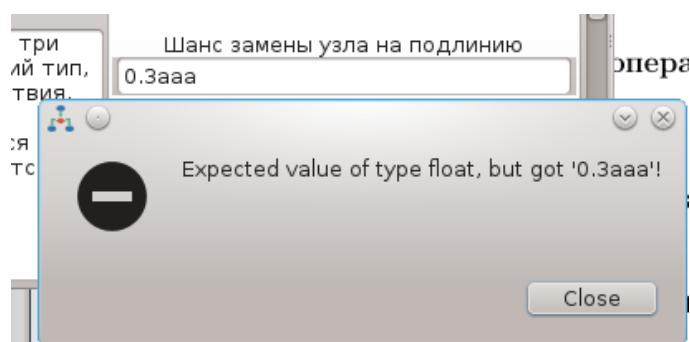


Рисунок 3 – Сообщение об ошибке распознавания входных данных

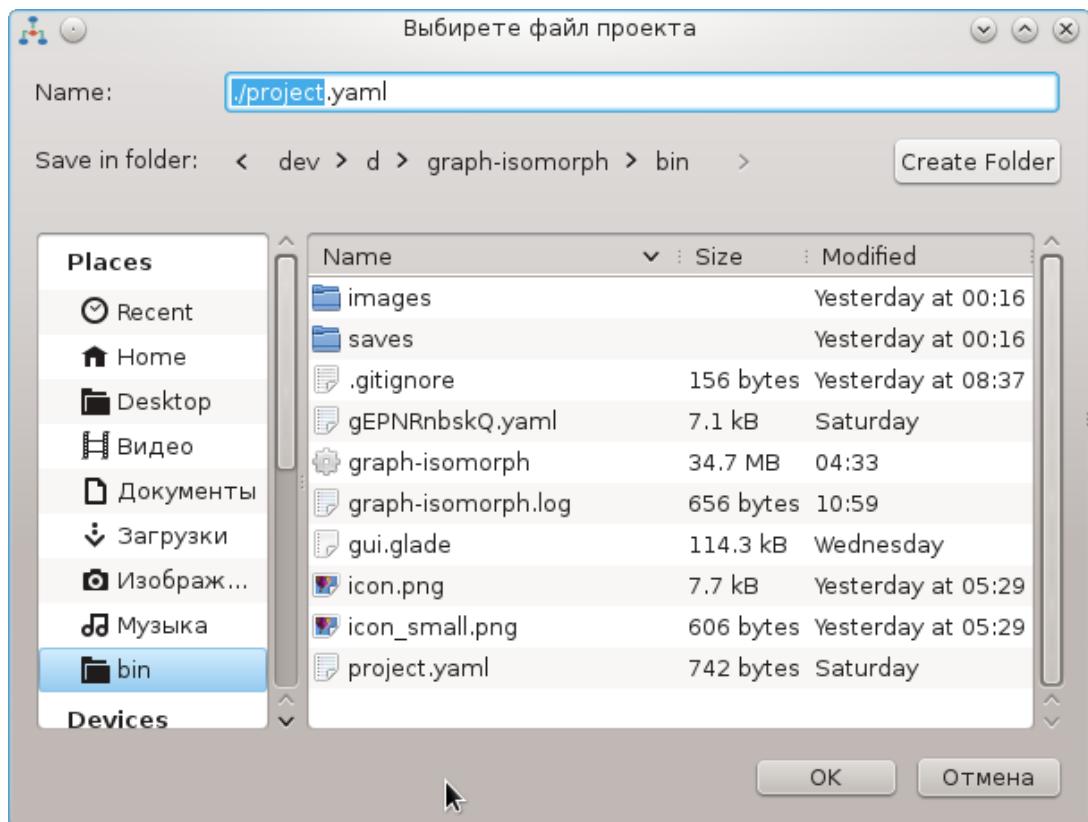


Рисунок 4 – Диалог сохранения проекта

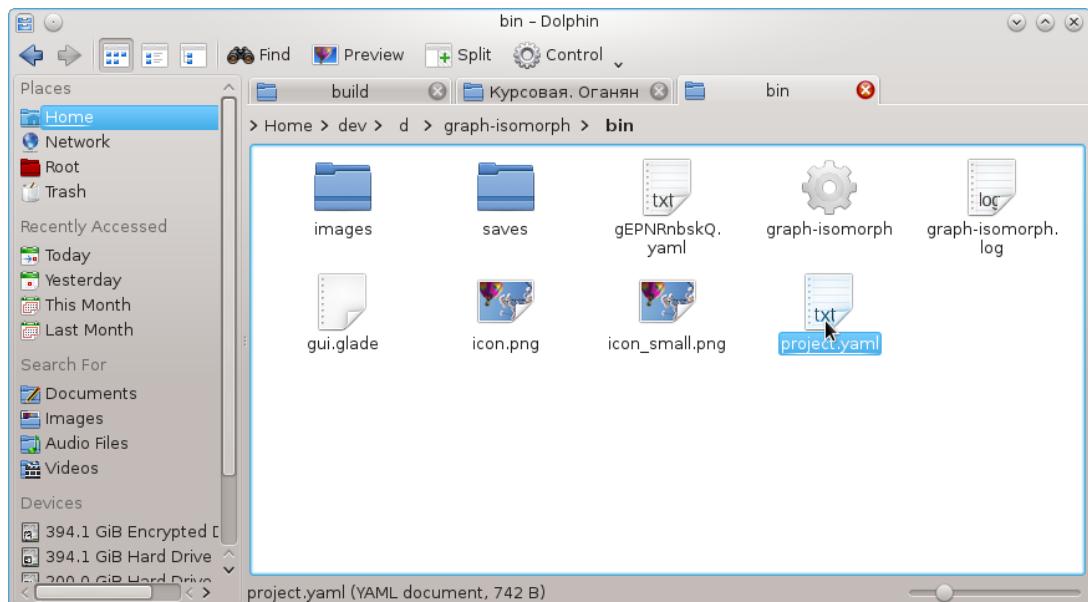


Рисунок 5 – Файл проекта в файловом менеджере

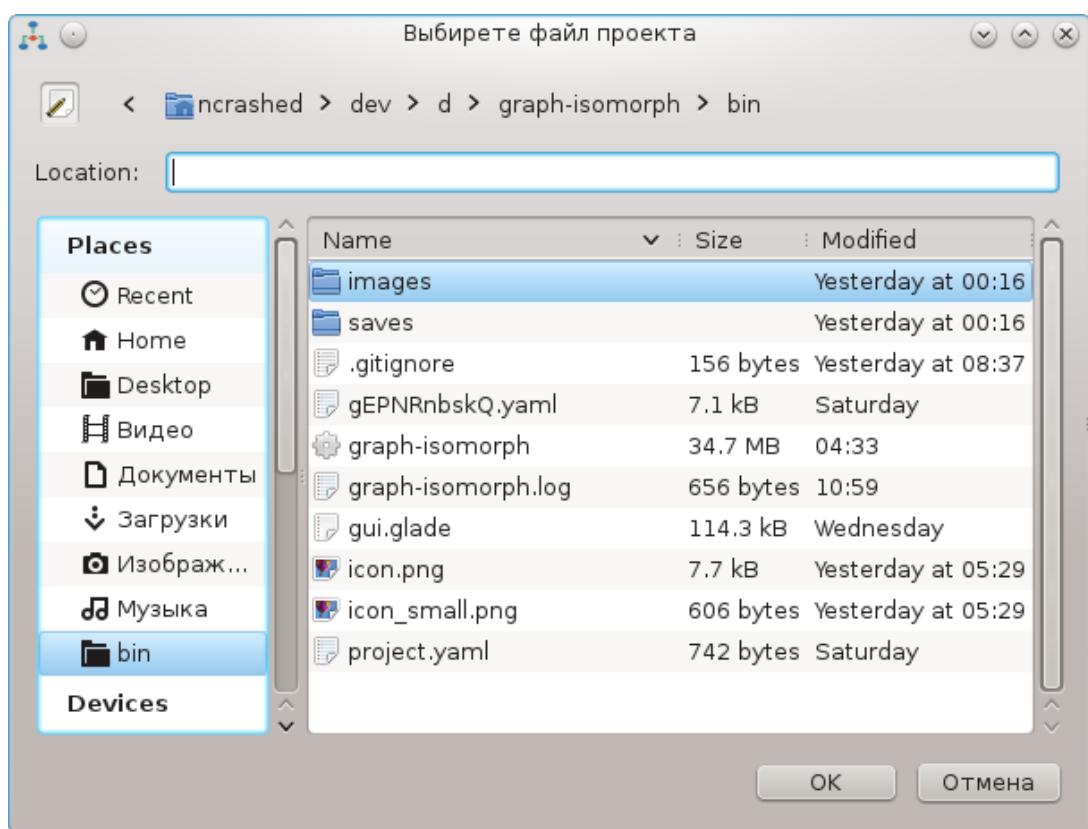


Рисунок 6 – Диалог открытия проекта

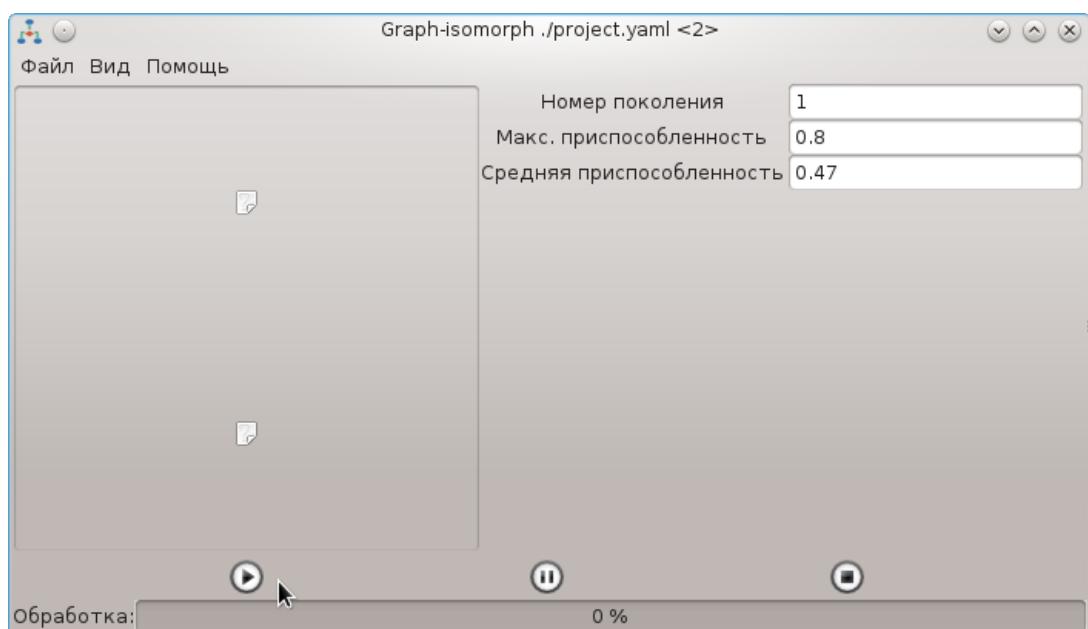


Рисунок 7 – Только что открытое окно эволюции

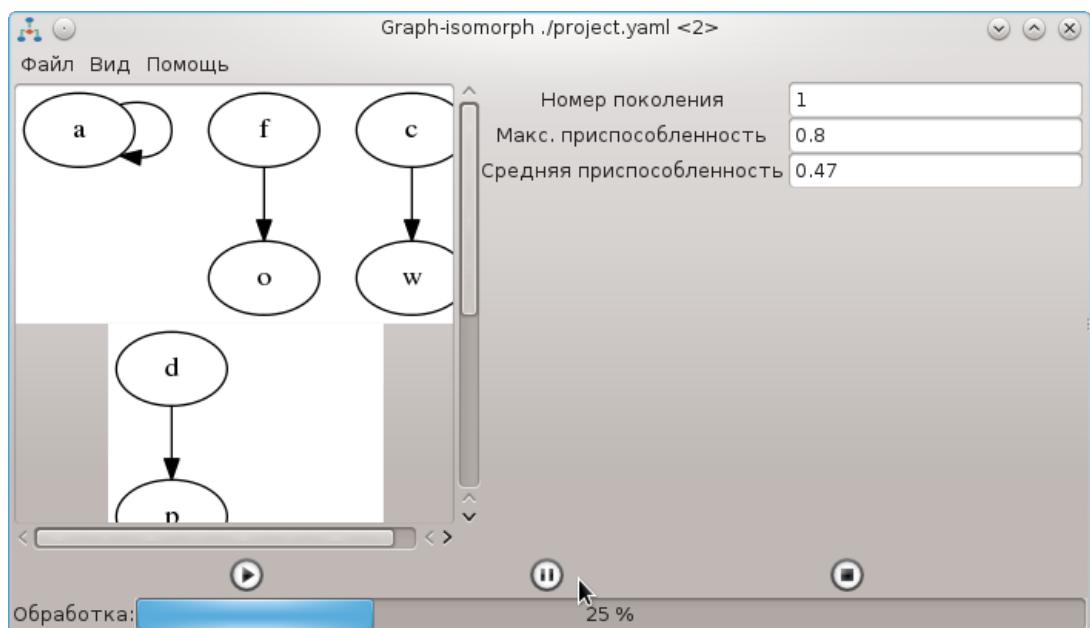


Рисунок 8 – Запуск эволюции и постановка на паузу

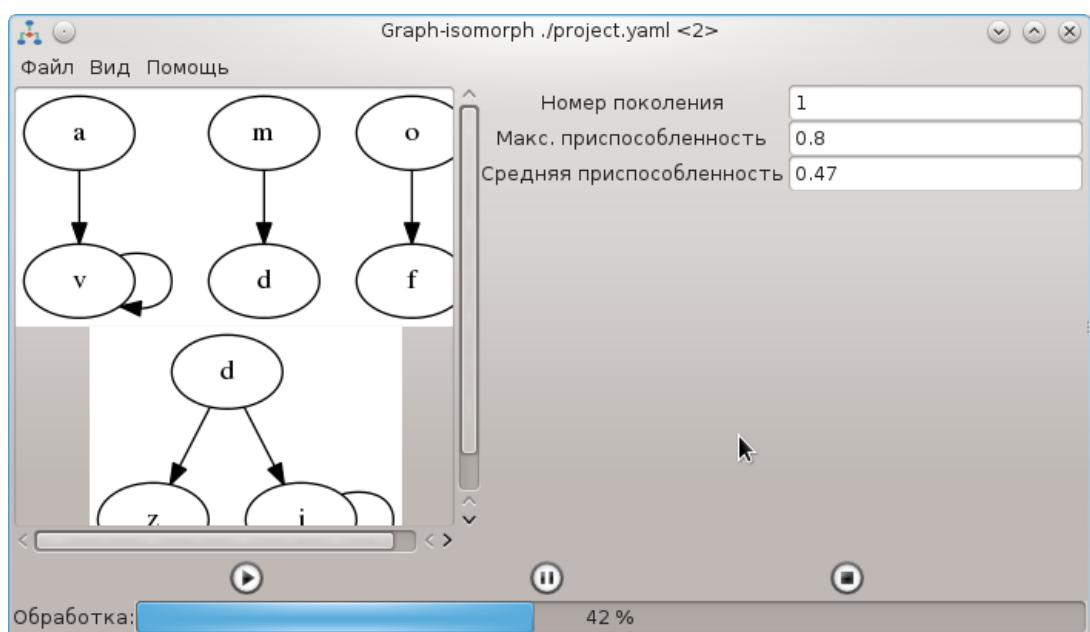


Рисунок 9 – Возобновление эволюции

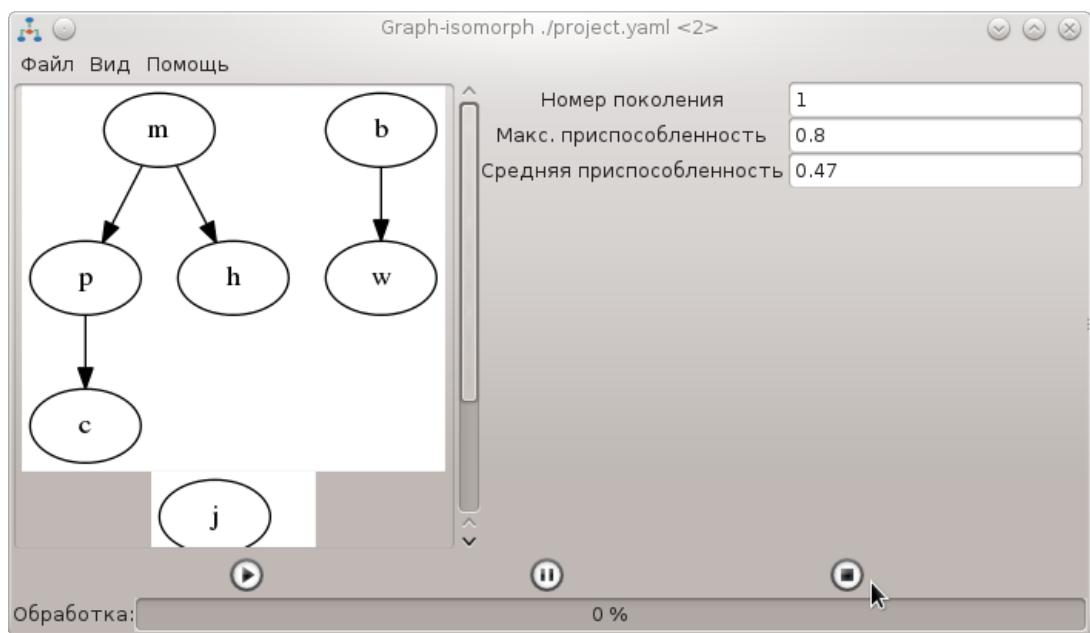


Рисунок 10 – Остановка эволюции

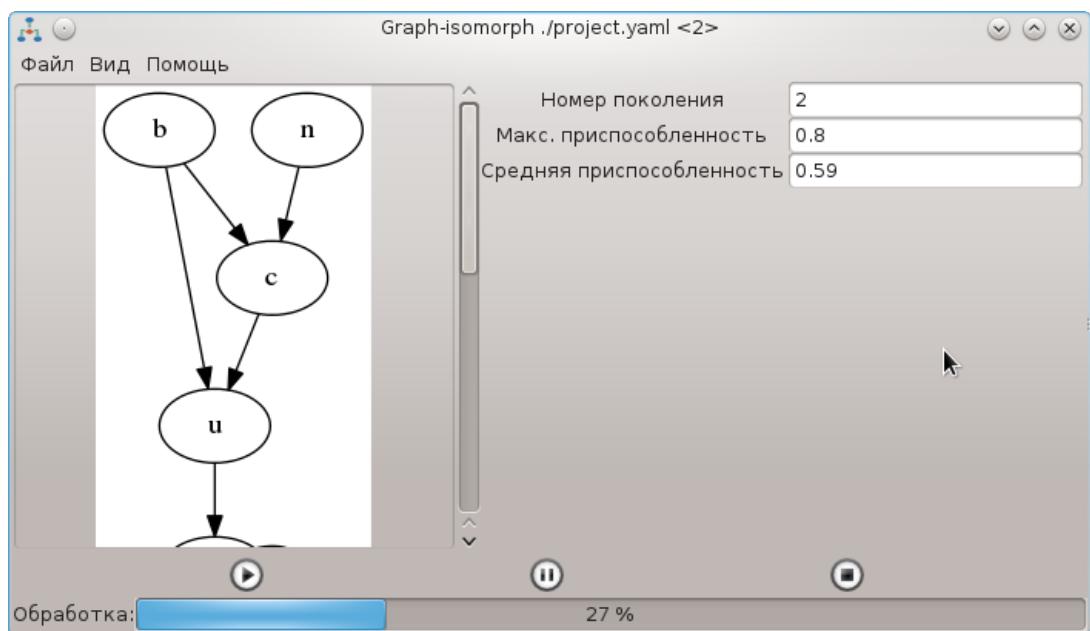


Рисунок 11 – Обновление промежуточных результатов

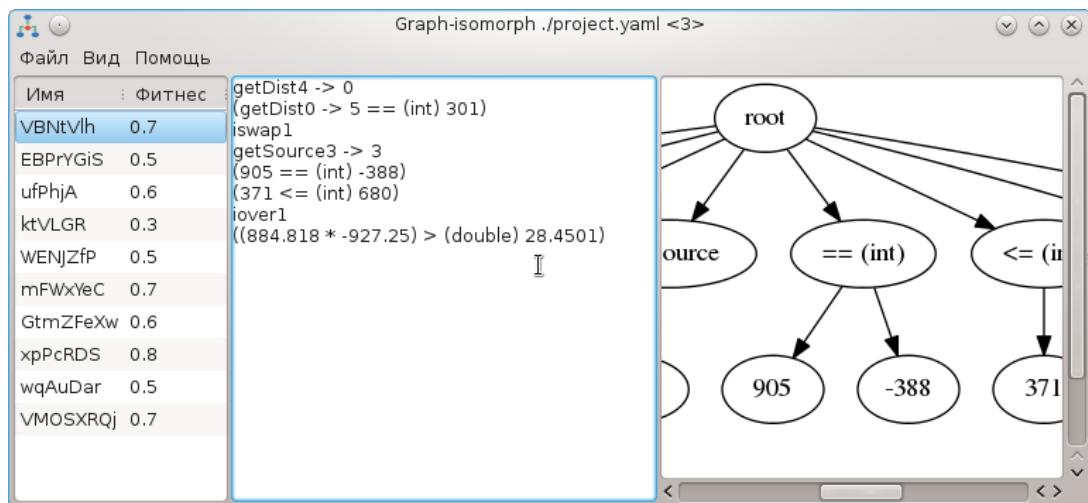


Рисунок 12 – Окно результатов

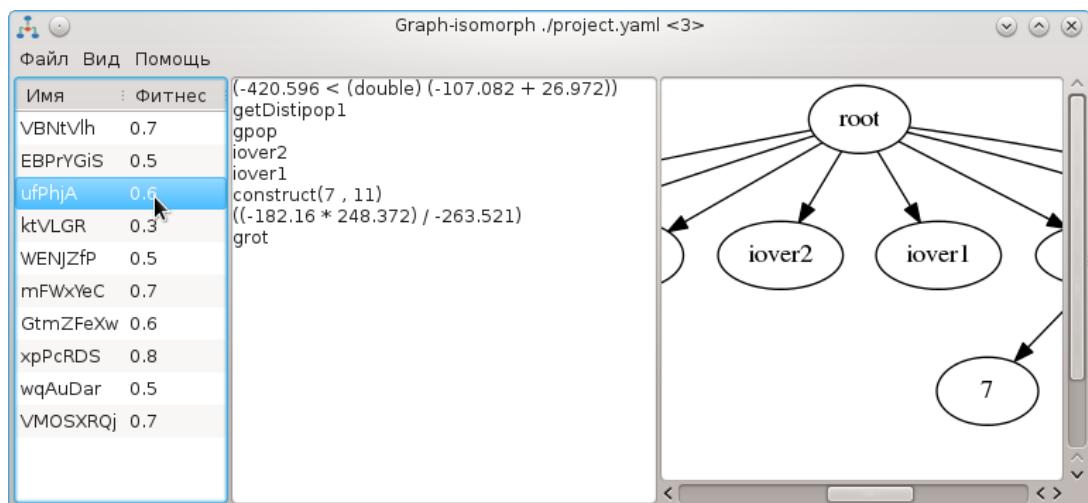


Рисунок 13 – Выбор одного из текущих алгоритмов и просмотр его значения функции приспособленности и его исходного кода

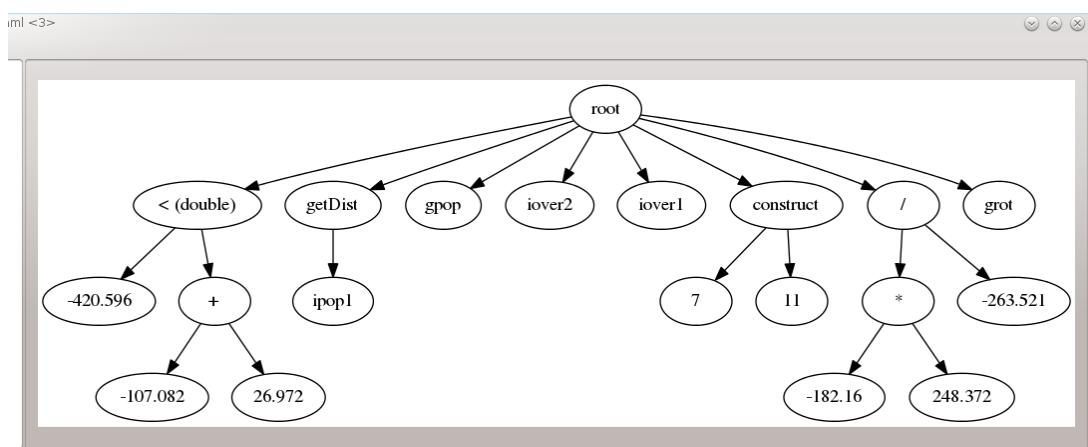


Рисунок 14 – Просмотр графической формы исходного кода индивида

*Федеральное государственное бюджетное образовательное
учреждение высшего профессионального образования
Московский государственный технический университет им.
Н. Э. Баумана*

23 0102

**АСОИ поиска алгоритмов распознавания изоморфизма
графов с помощью генетического программирования
Исходный код**

Студент группы ИУ5-82

Гуща А. В

“___” _____

2016

Содержание

1 Пакет APP(MAIN)

```
module app;

import gtk.Builder;
import gtk.Button;
import gtk.Main;
import gtk.Widget;
import gtk.ApplicationWindow;
import gtk.MenuItem;
import gobject.Type;

import std.stdio;
import std.getopt;
import std.c.process;
import std.file;

import dlogg.strict;

import gui.evolution;
import gui.results;
import gui.settings;

import project;
import application;

enum helpMsg =
"graph-isomorph [options]

options: --gui=<path> - path to glade file. Optional, default is 'gui.glade'.
          --log=<path> - path to log file. Optional, default is 'graph-isomorph.log'.
          --proj=<path> - path to project file. Optional, default is '"~Project.defaultProjectPath"'.
          --help           - display the message.";

void main(string[] args)
{
    string gladeFile = "./gui.glade";
    string logFile = "./graph-isomorph.log";
    string projFile = Project.defaultProjectPath;

    bool help = false;
    getopt(args,
           "gui", &gladeFile,
           "log", &logFile,
           "proj", &projFile,
           "help", &help
    );

    if(help)
    {
        writeln(helpMsg);
        return;
    }

    Main.initMultiThread(args);

    auto application = new Application(logFile, gladeFile, projFile);
    scope(exit) application.finalize();

    Main.run();
}
```

2 Пакет APPLICATION

```
module application;

import gtk.Builder;
import gtk.ApplicationWindow;

import gui.settings;
import gui.evolution;
import gui.results;

import dlogg.strict;

import project;
import std.file;
```

```

class Application
{
    shared ILogger logger;

    SettingsWindow settingsWindow;
    EvolutionWindow evolutionWindow;
    ResultsWindow resultsWindow;

    Project project;

    this(string logFile, string gladeFile, string projFile)
    {
        logger = new shared StrictLogger(logFile);

        logger.logInfo("Loading project file ...");
        project = new Project(logger);
        if(projFile.exists)
        {
            project.open(projFile);
        }
        else
        {
            logger.logInfo("Cannot find project file, creating new project");
        }

        Builder builder = new Builder();
        if( !builder.addFromFile(gladeFile) )
        {
            logger.logError(text("Failed to create gui from glade file ", gladeFile, "!'"));
            return;
        }

        logger.logInfo("Loading settings window");
        auto settingsWnd = cast(ApplicationWindow) builder.getObject("SettingsWindow");
        if(settingsWnd is null)
        {
            logger.logError("Failed to create settings window!");
            return;
        }

        logger.logInfo("Loading evolution window");
        auto evolutionWnd = cast(ApplicationWindow) builder.getObject("EvolutionWindow");
        if(evolutionWnd is null)
        {
            logger.logError("Failed to create evolution window!");
            return;
        }

        logger.logInfo("Loading results window");
        auto resultsWnd = cast(ApplicationWindow) builder.getObject("ResultsWindow");
        if(resultsWnd is null)
        {
            logger.logError("Failed to create results window!");
            return;
        }

        settingsWindow = new SettingsWindow(this, builder, logger, project, settingsWnd, evolutionWnd, resultsWnd,
                                         ↵ );
        evolutionWindow = new EvolutionWindow(this, builder, logger, project, settingsWnd, evolutionWnd, resultsWnd);
        resultsWindow = new ResultsWindow(this, builder, logger, project, settingsWnd, evolutionWnd, resultsWnd);

        updateAll();
    }

    void updateAll()
    {
        settingsWindow.updateContent();
        evolutionWindow.updateContent();
        resultsWindow.updateContent();
    }

    void finalize()
    {
        logger.finalize();
    }
}

```

3 Пакет PROJECT

```
module project;
```

```

import evol.progtype;
import evol.compiler;

import dyaml.all;
import std.path;
import std.file;
import std.stream;

import dlogg.log;

class Project
{
    ProgramType programType;
    GraphPopulation population;
    string name;
    string filename;
    string populationPath;

    bool popLoaded = true;

    private shared ILogger logger;

    enum defaultProjectPath = "./project.yaml";
    enum evolSettings = "evolutionSettings";
    enum projectName = "name";
    enum popPathKeyName = "population";

    this(shared ILogger logger)
    {
        this.logger = logger;
        programType = new ProgramType();
        filename = defaultProjectPath;
        name = defaultProjectPath.baseName;
    }

    private void open(Node root)
    {
        programType = new ProgramType();

        if(root.containsKey(projectName))
        {
            name = root[projectName].as!string;
        }

        if(root.containsKey(evolSettings))
        {
            Node node = root[evolSettings];

            void setValue(string field)()
            {
                mixin("alias T = typeof(programType." ~ field ~");");
                if(node.containsKey(field))
                {
                    mixin("programType." ~ field ~" = node[" ~ field ~"].as!`~T.stringof~;`");
                }
            }

            setValue!"progMinSize";
            setValue!"progMaxSize";
            setValue!"newOpGenChance";
            setValue!"newScopeGenChance";
            setValue!"newLeafGenChance";
            setValue!"scopeMinSize";
            setValue!"scopeMaxSize";
            setValue!"mutationChance";
            setValue!"crossingoverChance";
            setValue!"mutationChangeChance";
            setValue!"mutationReplaceChance";
            setValue!"mutationDeleteChance";
            setValue!"mutationAddLineChance";
            setValue!"mutationRemoveLineChance";
            setValue!"copyingPart";
            setValue!"deleteMutationRiseGenomeSize";
            setValue!"maxGenomeSize";
            setValue!"populationSize";
            setValue!"graphPermuteChance";
            setValue!"graphNodesCountMin";
            setValue!"graphNodesCountMax";
            setValue!"graphLinksCountMin";
            setValue!"graphLinksCountMax";
            setValue!"graphPermutatesCountMin";
            setValue!"graphPermutatesCountMax";
        }

        if(root.containsKey(popPathKeyName))
        {
    
```

```

        populationPath = root[popPathKeyName].as!string;
    }
}

void open(string filename)
{
    this.filename = filename;
    open(Loader(filename).load());

    loadPopulation();
}

void save(string filename)
{
    this.filename = filename;
    savePopulation();
    Dumper(filename).dump(dump);
}

private void savePopulation()
{
    if(population == null) return;

    if(populationPath == "")
    {
        populationPath = population.name~".yaml";
    }

    try
    {
        if(!populationPath.dirName.exists())
        {
            mkdirRecurse(populationPath.dirName);
        }

        Dumper(populationPath).dump(population.saveYaml);
    } catch(Exception e)
    {
        logger.logError(text("Failed to load population from '", populationPath, "'. Reason: ", e.msg));
        debug logger.logError(e.toString());
    }

    population = null;
    popLoaded = true;
}
}

private void loadPopulation()
{
    try
    {
        auto node = Loader(populationPath).load();
        population = GraphPopulation.loadYaml(node);
        popLoaded = true;
    } catch(Exception e)
    {
        logger.logError(text("Failed to load population from '", populationPath, "'. Reason: ", e.msg));
        debug logger.logError(e.toString());
    }

    population = null;
    popLoaded = true;
}
}

private Node dump()
{
    Node[string] emap;

    void setValue(string field)()
    {
        mixin("alias T = typeof(programType.^field^);");
        emap[field] = Node(mixin("programType.^field"));
    }

    setValue!"progMinSize";
    setValue!"progMaxSize";
    setValue!"newOpGenChance";
    setValue!"newScopeGenChance";
    setValue!"newLeafGenChance";
    setValue!"scopeMinSize";
    setValue!"scopeMaxSize";
    setValue!"mutationChance";
    setValue!"crossingoverChance";
    setValue!"mutationChangeChance";
    setValue!"mutationReplaceChance";
    setValue!"mutationDeleteChance";
    setValue!"mutationAddLineChance";
}

```

```

        setValue!"mutationRemoveLineChance";
        setValue!"copyingPart";
        setValue!"deleteMutationRiseGenomeSize";
        setValue!"maxGenomeSize";
        setValue!"populationSize";
        setValue!"graphPermuteChance";
        setValue!"graphNodesCountMin";
        setValue!"graphNodesCountMax";
        setValue!"graphLinksCountMin";
        setValue!"graphLinksCountMax";
        setValue!"graphPermutatesCountMin";
        setValue!"graphPermutatesCountMax";

    return Node([
        projectName : Node(name),
        evolSettings : Node(emap),
        popPathKeyName : Node(populationPath)
    ]);
}

void recreate(string filename)
{
    programType = new ProgramType();
    name = filename.baseName;
    this.population = null;
    this.filename = filename;
    save(filename);
}
}

```

4 Пакет GUI.EVOLUTION

```

module gui.evolution;

import gtk.Builder;
import gtk.MenuItem;
import gtk.ApplicationWindow;
import gtk.Image;
import gtk.ToolButton;
import gtk.ProgressBar;
import gdk.Threads;
import gtk.Entry;

import gui.util;
import gui.generic;
import dlogg.log;

import graph.directed;

import project;
import application;

import std.file;
import std.stdio;
import std.process;
import std.path;
import std.conv;
import std.concurrency;
import std.datetime;
import std.functional;

import core.thread;

import evol.compiler;
import evol.world;

class EvolutionWindow : GenericWindow
{
    this(Application app, Builder builder, shared ILogger logger
        , Project project
        , ApplicationWindow settingsWindow
        , ApplicationWindow evolutionWindow
        , ApplicationWindow resultsWindow)
    {
        super(app, builder, logger, project, evolutionWindow);

        evolutionWindow.hide();
        evolutionWindow.addOnHide( (w) => onWindowHideShow(AppWindow.Evolution, true) );
        evolutionWindow.addOnShow( (w) => onWindowHideShow(AppWindow.Evolution, false) );
        evolutionWindow.addonDelete( (e, w) { evolutionWindow.hide; return true; } );
    }

    auto showSettingsWndItem = cast(MenuItem)builder.getObject("ShowSettingsWndItem2");

```

```

    if(showSettingsWndItem == null)
    {
        logger.logError("EvolutionWnd: failed to get show settings wnd item!");
        assert(false);
    }
    showSettingsWndItem.addOnActivate( (w) => settingsWindow.showAll() );

    auto showResultsWndItem = cast(MenuItem) builder.getObject("ShowResultsWndItem2");
    if(showResultsWndItem == null)
    {
        logger.logError("EvolutionWnd: failed to get show results wnd item!");
        assert(false);
    }
    showResultsWndItem.addOnActivate( (w) => resultsWindow.showAll() );

    initProjectSaveLoad("2");
    initAboutDialog("2");

    initEvolution();
    initEvolutionControl();
}

void setInputImages(IDirectedGraph first, IDirectedGraph second)
{
    void setImage(IDirectedGraph graph, string wname)
    {
        auto image = cast(Image) builder.getObject(wname);
        assert(image != null);

        try
        {
            enum tempImageDir = "./images";
            if(!tempImageDir.exists)
            {
                mkdirRecurse(tempImageDir);
            }

            string dotfilename = buildPath(tempImageDir, wname + ".dot");
            string imagefilename = buildPath(tempImageDir, wname + ".png");

            auto file = File(dotfilename, "w");
            file.writeln(graph.genDot());
            file.close();

            shell(text("dot -Tpng ", dotfilename, " > ", imagefilename));

            image.setFromFile(imagefilename);
        }
        catch(Exception e)
        {
            logger.logError("Failed to load image from graph for " + wname);
            logger.logError(e.msg);
        }
    }

    setImage(first, "InputGraphImage1");
    setImage(second, "InputGraphImage2");
}

void setGenerationNumber(size_t i)
{
    auto entry = cast(Entry) builder.getObject("GenerationNumberEntry");
    assert(entry != null);

    entry.setText(toString(i+1));
}

void setMaxFitness(double fitness)
{
    auto entry = cast(Entry) builder.getObject("MaxFitnessEntry");
    assert(entry != null);

    entry.setText(toString(fitness));
}

void setAvarageFitness(double fitness)
{
    auto entry = cast(Entry) builder.getObject("AvarageFitnessEntry");
    assert(entry != null);

    entry.setText(toString(fitness));
}

void initEvolutionControl()
{
    auto startBtn = cast(ToolButton) builder.getObject("EvolutionStartButton");
}

```

```

assert(startBtn != null);

startBtn.setOnClickListener((b)
{
    try
    {
        startEvolution();
    } catch(Throwable th)
    {
        logger.logError(th.toString());
    }
});

auto pauseBtn = cast(ToolButton)builder.getObject("EvolutionPauseButton");
assert(pauseBtn != null);

pauseBtn.setOnClickListener((b)
{
    try
    {
        pauseEvolution();
    } catch(Throwable th)
    {
        logger.logError(th.toString());
    }
});

auto stopBtn = cast(ToolButton)builder.getObject("EvolutionStopButton");
assert(stopBtn != null);

stopBtn.setOnClickListener((b)
{
    try
    {
        stopEvolution();

        auto progressBar = cast(ProgressBar)builder.getObject("EvolutionProgressBar");
        assert(progressBar != null);
        progressBar.setFraction(0);

    } catch(Throwable th)
    {
        logger.logError(th.toString());
    }
});
}

void initEvolution()
{
    compiler = new GraphCompiler(
        new GraphCompilation(project, ()
        {
            threadsEnter();
            application.updateAll();
            threadsLeave();
        })
        , project.programType
        , new GraphWorld(
            project.programType
            ,(gr1, gr2)
            {
                threadsEnter();
                setInputImages(gr1, gr2);
                threadsLeave();
            }));
}

evolState = EvolutionState.Stoped;
}

void startEvolution()
{
    final switch(evolState)
    {
        case(EvolutionState.Running):
        {
            return;
        }
        case(EvolutionState.Paused):
        {
            evolutionTid.send(thisTid, EvolutionCommand.Resume);
            return;
        }
        case(EvolutionState.Stoped):
        {
            compiler.clean();
            if(project.popLoaded)

```

```

        {
            project.population = compiler.addPop(
                project.programType.populationSize);
        } else
        {
            project.popLoaded = false;
        }
    evolutionTid = spawn(&evolutionThread, cast(shared) this);
    evolutionTid.send(thisTid);
    return;
}
}

void stopEvolution()
{
    final switch(evolState)
    {
        case( EvolutionState.Running):
        {
            evolutionTid.send(thisTid, EvolutionCommand.Stop);
            return;
        }
        case( EvolutionState.Paused):
        {
            evolutionTid.send(thisTid, EvolutionCommand.Stop);
            return;
        }
        case( EvolutionState.Stoped):
        {
            return;
        }
    }
}

override void updateContent()
{
    super.updateContent();

    if(project.population != null)
    {
        setGenerationNumber(cast(size_t) project.population.generation);

        double val = 0.0;
        double maxFitness = 0.0;
        foreach(ind; project.population)
        {
            if(ind.fitness > maxFitness)
                maxFitness = ind.fitness;

            val += ind.fitness;
        }

        setMaxFitness(maxFitness);
        if(val != 0.0)
        {
            setAvarageFitness(val / cast(double) project.population.length);
        } else
        {
            setAvarageFitness(0.0);
        }
    }
}

void pauseEvolution()
{
    final switch(evolState)
    {
        case( EvolutionState.Running):
        {
            evolutionTid.send(thisTid, EvolutionCommand.Pause);
            return;
        }
        case( EvolutionState.Paused):
        {
            evolutionTid.send(thisTid, EvolutionCommand.Pause);
            return;
        }
        case( EvolutionState.Stoped):
        {
            return;
        }
    }
}

private

```

```

{
    enum EvolutionState
    {
        Stoped,
        Running,
        Paused
    }

    enum EvolutionCommand
    {
        Pause,
        Resume,
        Stop
    }

    __gshared EvolutionState evolState;
    __gshared GraphCompiler compiler;
    Tid evolutionTid;

    static void evolutionThread(shared EvolutionWindow wndShared)
    {
        Thread .getThis() .isDaemon(true);

        EvolutionWindow wnd = cast() wndShared;
        try
        {
            auto progressBar = cast(ProgressBar)wnd .builder .getObject("EvolutionProgressBar");
            assert(progressBar != null);

            evolState = EvolutionState.Running;
            scope(exit) evolState = EvolutionState.Stoped;

            wnd .project .programType .registerTypes();

            bool exit = false;
            bool paused = false;
            Tid parent = receiveOnly! Tid();

            void listener()
            {
                receiveTimeout(dur!"msecs"(1),
                    (Tid sender, EvolutionCommand command)
                {
                    final switch(command)
                    {
                        case(EvolutionCommand.Resume):
                        {
                            evolState = EvolutionState.Running;
                            paused = false;
                            break;
                        }
                        case(EvolutionCommand.Pause):
                        {
                            evolState = EvolutionState.Paused;
                            paused = true;
                            break;
                        }
                        case(EvolutionCommand.Stop):
                        {
                            exit = true;
                            paused = false;
                            evolState = EvolutionState.Paused;
                            break;
                        }
                    }
                });
            }
        }

        void updater(double percent)
        {
            listener();
            assert(progressBar != null);

            threadsEnter();
            progressBar.setFraction(percent);
            threadsLeave();
        }
        auto updaterDelegate = toDelegate(&updater);

        bool whenExit()
        {
            return exit;
        }
        auto whenExitDelegate = toDelegate(&whenExit);

        bool pauser()
    }
}

```

5 Пакет GUI.GENERIC

```
module gui.generic;

import gtk.Builder;
import gtk.ApplicationWindow;
import gtk.ImageMenuItem;
import gtk.FileChooserDialog;
import gtk.AboutDialog;
import gdk.Pixbuf;
import gtk.Main;
import dlogg.log;

import project;
import application;

import std.file;

abstract class GenericWindow
{
    enum defaultWindowTittle = "Graph-isomorph";

    this(Application app, Builder builder, shared ILogger logger, Project project
        , ApplicationWindow window)
    {
        this.app = app;
        mBuilder = builder;
        mLogger = logger;
        mProject = project;
        mWindow = window;

        window.setIconFromFile("icon_small.png");
        updateTittle();
    }

    Application application()
    {
        return app;
    }

    void updateTittle()
    {
        window.setTitle(defaultWindowTittle ~ " " ~ project.filename);
    }

    Builder builder()
    {
        return mBuilder;
    }

    shared(ILogger) logger()
    {
        return mLogger;
    }

    Project project()
    {
    }
}
```

```

    return mProject;
}

ApplicationWindow window()
{
    return mWindow;
}

void updateContent()
{
    updateTittle();
}

void initProjectSaveLoad(string distinct)
{
    logger.logInfo("New project button setup");
    auto newItem = cast(ImageMenuItem) builder.getObject("NewProjectMenuItem"~distinct);
    assert(newItem !is null);
    newItem.addOnActivate((i)
    {
        try
        {
            auto dlg = new FileChooserDialog("Choose new project file"
                , window
                , FileChooserAction.SAVE
                , ["OK", "Cancel"]
                , [ResponseType.OK, ResponseType.CANCEL]
            );

            dlg.setDoOverwriteConfirmation(true);
            dlg.setCurrentFolder(getcwd());
            dlg.setCurrentName(Project.defaultProjectPath);

            switch(dlg.run)
            {
                case(ResponseType.OK):
                {
                    string filename = dlg.getFilename;
                    if(filename !is null)
                    {
                        project.recreate(filename);
                        application.updateAll();
                    }
                    dlg.destroy;
                    return;
                }
                default:
                {
                    dlg.destroy;
                    return;
                }
            }
        }
        catch(Throwable e)
        {
            logger.LogError(e.toString);
        }
    });
}

logger.logInfo("Open project button setup");
auto openItem = cast(ImageMenuItem) builder.getObject("OpenProjectMenuItem"~distinct);
assert(openItem !is null);
openItem.addOnActivate((i)
{
    try
    {
        auto dlg = new FileChooserDialog("Выберите файл проекта"
            , window
            , FileChooserAction.OPEN
            , ["OK", "Отмена"]
            , [ResponseType.OK, ResponseType.CANCEL]
        );

        dlg.setCurrentFolder(getcwd());

        switch(dlg.run)
        {
            case(ResponseType.OK):
            {
                string filename = dlg.getFilename;
                if(filename !is null)
                {
                    project.open(filename);
                    app.updateAll();
                }
                dlg.destroy;
            }
        }
    }
});

```

```

        return;
    }
    default:
    {
        dlg.destroy();
        return;
    }
}
catch(Throwable e)
{
    logger.logError(e.toString());
}
});

logger.logInfo("Save as project button setup");
auto saveAsItem = cast(ImageMenuItem)builder.getObject("SaveAsMenuItem"~distinct);
assert(saveAsItem !is null);
saveAsItem.addOnActivate((i)
{
    try
    {
        auto dlg = new FileChooserDialogВыберите(" файлпроекта "
            , window
            , FileChooserAction.SAVE
            , ["OK", "Отмена"]
            , [ResponseType.OK, ResponseType.CANCEL]
        );
        dlg.setDoOverwriteConfirmation(true);
        dlg.setCurrentFolder(getcwd());
        dlg.setCurrentName(Project.defaultProjectPath);

        switch(dlg.run)
        {
            case(ResponseType.OK):
            {
                string filename = dlg.getFilename();
                if(filename !is null)
                {
                    project.save(filename);
                    app.updateAll();
                }
                dlg.destroy();
                return;
            }
            default:
            {
                dlg.destroy();
                return;
            }
        }
    }
    catch(Throwable e)
    {
        logger.logError(e.toString());
    }
});

logger.logInfo("Save project button setup");
auto saveItem = cast(ImageMenuItem)builder.getObject("SaveMenuItem"~distinct);
assert(saveItem !is null);
saveItem.addOnActivate((i)
{
    try
    {
        project.save(project.filename);
        app.updateAll();
    }
    catch(Throwable e)
    {
        logger.logError(e.toString());
    }
});

logger.logInfo("Exit application button setup");
auto exitItem = cast(ImageMenuItem)builder.getObject("ExitMenuItem"~distinct);
assert(exitItem !is null);
exitItem.addOnActivate((i)
{
    try
    {
        Main.quit();
    }
    catch(Throwable e)
    {

```

```

        logger.logError(e.toString());
    });
}
}

void initAboutDialog(string distinct)
{
    auto helpItem = cast(ImageMenuItem)builder.getObject("AboutMenuItem"~distinct);
    assert(helpItem);

    helpItem.addOnActivate((i)
    {
        auto dlg = new AboutDialog;
        dlg.setProgramNameACOI(" поиска алгоритмов распознавания изоморфизмов графов
            ↪ проектирование МГТУ им . НЭ Баумана.. \n" Научный" руководитель:
            ↪ Филиппович Орий Николаевич ");
        dlg.setAuthorsGusa([" Антон Валерьевич "]);
        dlg.setLicenseType(GtkLicense.MIT_X11);
        dlg.addOnResponse((r,d) => dlg.destroy);
        auto logo = new Pixbuf("icon.png");
        dlg.setLogo(logo);
        dlg.showAll;
    });
}

private
{
    Builder mBuilder;
    shared ILogger mLogger;
    Project mProject;
    ApplicationWindow mWindow;
    Application app;
}
}

```

6 Пакет GUI.RESULTS

```

module gui.results;

import gtk.Builder;
import gtk.MenuItem;
import gtk.ApplicationWindow;
import gtk.TreeView;
import gtk.TreeIter;
import gtk.TextView;
import gtk.Image;
import gtk.ListStore;

import gui.util;
import gui.generic;

import dlogg.log;

import project;
import application;

import std.conv;
import std.file;
import std.path;
import std.stdio;
import std.process;

import devol.individ;

class ResultsWindow : GenericWindow
{
    this(Application app, Builder builder, shared ILogger logger
        , Project project
        , ApplicationWindow settingsWindow
        , ApplicationWindow evolutionWindow
        , ApplicationWindow resultsWindow)
    {
        super(app, builder, logger, project, resultsWindow);

        resultsWindow.hide();
        resultsWindow.addOnHide( (w) => onWindowHideShow(AppWindow.Results, true) );
        resultsWindow.addOnShow( (w) => onWindowHideShow(AppWindow.Results, false) );
        resultsWindow.addonDelete( (e, w) { resultsWindow.hide; return true; } );

        auto showSettingsWndItem = cast(MenuItem)builder.getObject("ShowSettingsWndItem3");
        if(showSettingsWndItem is null)
        {

```

```

        logger.logError("ResultsWnd: failed to get show settings wnd item!");
        assert(false);
    }
    showSettingsWndItem.addOnActivate( (w) => settingsWindow.showAll() );
}

auto showEvolutionWndItem = cast(MenuItem) builder.getObject("ShowEvolutionWndItem3");
if(showEvolutionWndItem is null)
{
    logger.logError("ResultsWnd: failed to get show evolution wnd item!");
    assert(false);
}
showEvolutionWndItem.addOnActivate( (w) => evoluitionWindow.showAll() );

initProjectSaveLoad("3");
initAboutDialog("3");

initPopulationView();
}

private void setImage(IndAbstract graph, string wname)
{
    try
    {
        enum tempImageDir = "./images";
        if(!tempImageDir.exists())
        {
            mkdirRecurse(tempImageDir);
        }

        string dotfilename = buildPath(tempImageDir, wname + ".dot");
        string imagefilename = buildPath(tempImageDir, wname + ".png");

        auto file = File(dotfilename, "w");
        file.writeln(graph.genDot());
        file.close();

        shell(text("dot -Tpng ", dotfilename, " > ", imagefilename));

        programImage.setFromFile(imagefilename);
    }
    catch(Exception e)
    {
        logger.logError("Failed to load image from graph for " ~ wname);
        logger.logError(e.msg);
    }
}

private TreeView individsView;
private ListStore individsViewModel;
private TextView programView;
private Image programImage;

void updatePopulation()
{
    individsViewModel.clear();
    programView.getBuffer().setText("");
    programImage.clear();

    if(project.population !is null)
    {
        foreach(i, ind; project.population)
        {
            auto iter = new TreeIter();
            individsViewModel.insert(iter, -1);

            individsViewModel.setValue(iter, 0, ind.name);
            individsViewModel.setValue(iter, 1, toString(ind.fitness));
            individsViewModel.setValue(iter, 2, cast(int)i);
        }
    }
}

override void updateContent()
{
    super.updateContent();
    updatePopulation();
}

void initPopulationView()
{
    individsView = cast(TreeView) builder.getObject("IndividsTreeView");
    assert(individsView !is null);

    individsViewModel = new ListStore([GType.STRING, GType.STRING, GType.INT]);
    individsView.setModel(individsViewModel);
}

```

```

programView = cast(TextView) builder.getObject("ProgramTextView");
assert(programView != null);

programImage = cast(Image) builder.getObject("ProgramImage");
assert(programImage != null);

individsView.addOnCursorChanged((v)
{
    auto model = individsView.getModel();
    assert(model != null);

    auto iter = individsView.getSelectedIter();
    if(iter == null)
    {
        programImage.clear();
        programView.getBuffer().setText("");
    }
    else
    {
        auto individId = model.getValueInt(iter, 2);
        if(individId < 0 || individId >= project.population.length) return;

        auto individ = project.population[cast(size_t)individId];
        assert(individ != null);

        programView.getBuffer().setText(individ.programString);
        setImage(individ, individ.name);
    }
});
}
}

```

7 Пакет GUI.SETTINGS

```
module gui.settings;

import gtk.Builder;
import gtk.MenuItem;
import gtk.ApplicationWindow;
import gtk.Entry;
import gtk.MessageDialog;
import gtk.Widget;
import gtk.TreeView;
import gtk.TextView;
import gtk.TextIter;
import gtk.ListStore;
import gtk.TreeIter;

import gdk.Event;
import gobject.Value;

import gui.util;
import gui.generic;

import dlogg.log;

import evol.progtype;
import devol.operator;
import devol.operatormng;

import project;
import application;

import std.string;

class SettingsWindow : GenericWindow
{
    this(Application app, Builder builder, shared ILogger logger
        , Project project
        , ApplicationWindow settingsWindow
        , ApplicationWindow evolutionWindow
        , ApplicationWindow resultsWindow)
    {
        super(app, builder, logger, project, settingsWindow);

        settingsWindow.showAll();
        settingsWindow.addOnHide( (w) => onWindowHideShow(AppWindow.Settings, true) );
        settingsWindow.addOnShow( (w) => onWindowHideShow(AppWindow.Settings, false) );
        settingsWindow.addonDelete( (e, w) { settingsWindow.hide; return true; } );

        auto showEvolutionWndItem = cast(MenuItem)builder.getObject("ShowEvolutionWndItem1");
        if(showEvolutionWndItem is null)
```

```

{
    logger.logError("SettingsWnd: failed to get show evolution wnd item!");
    assert(false);
}
showEvolutionWndItem.addOnActivate( (w) => evolutionWindow.showAll() );

auto showResultsWndItem = cast(MenuItem) builder.getObject("ShowResultsWndItem1");
if(showResultsWndItem is null)
{
    logger.logError("SettingsWnd: failed to get show results wnd item!");
    assert(false);
}
showResultsWndItem.addOnActivate( (w) => resultsWindow.showAll() );

initProgtypeEntries();
initOperatorsView();
initProjectSaveLoad("1");
initAboutDialog("1");
}

override void updateContent()
{
    super.updateContent();
    reloadSettings();
}

private
{
    static char cupper(char c)
    {
        immutable source = "qwertyuiopasdfghjklzxcvbnm";
        immutable dist   = "QWERTYUIOPASDFGHJKLMZXCVBNM";

        foreach(i, sc; source)
        {
            if(sc == c) return dist[i];
        }

        return c;
    }

    template genEntryGetter(tt...)
    {
        enum field = tt[0];
        enum genEntryGetter = "auto " ~ field
            ~`Entry = cast(Entry) builder.getObject(
            ~[cupper(cast(char)field[0])] ~ field[1..$] ~ `Entry") ; " \n"
            ~`assert(~ field ~`Entry !is null);`;
    }

    template genInitialSetupText(tts...)
    {
        enum field = tts[0];
        enum genInitialSetupText = field ~`Entry.setText(project.programType.~ field ~`.to!string);`;
    }
}

void initProgtypeEntries()
{
    mixin(genEntryGetter!"progMinSize");
    mixin(genEntryGetter!"progMaxSize");
    mixin(genEntryGetter!"scopeMinSize");
    mixin(genEntryGetter!"scopeMaxSize");
    mixin(genEntryGetter!"newOpGenChance");
    mixin(genEntryGetter!"newScopeGenChance");
    mixin(genEntryGetter!"newLeafGenChance");
    mixin(genEntryGetter!"mutationChangeChance");
    mixin(genEntryGetter!"mutationReplaceChance");
    mixin(genEntryGetter!"mutationDeleteChance");
    mixin(genEntryGetter!"mutationAddLineChance");
    mixin(genEntryGetter!"mutationRemoveLineChance");
    mixin(genEntryGetter!"maxMutationChange");
    mixin(genEntryGetter!"mutationChance");
    mixin(genEntryGetter!"crossingoverChance");
    mixin(genEntryGetter!"copyingPart");
    mixin(genEntryGetter!"deleteMutationRiseGenomeSize");
    mixin(genEntryGetter!"maxGenomeSize");
    mixin(genEntryGetter!"populationSize");
    mixin(genEntryGetter!"graphPermuteChance");
    mixin(genEntryGetter!"graphNodesCountMin");
    mixin(genEntryGetter!"graphNodesCountMax");
    mixin(genEntryGetter!"graphLinksCountMin");
    mixin(genEntryGetter!"graphLinksCountMax");
    mixin(genEntryGetter!"graphPermutatesCountMin");
    mixin(genEntryGetter!"graphPermutatesCountMax");
}

```

```

void showInvalidValueDialog(T)(string value)
{
    auto dialog = new MessageDialog(window
        , GtkDialogFlags.MODAL
        , GtkMessageType.ERROR
        , GtkButtonsType.CLOSE
        , "%s"
        , text("Expected value of type ", T.stringof, " , but got '", value,"'"));
    dialog.addOnResponse( (r, d) => dialog.destroy );
    dialog.run();
}

bool delegate(Event, Widget) tryFillValue(T, string field)(Entry entry)
{
    return (e, w)
    {
        scope(failure)
        {
            showInvalidValueDialog!T(entry.getText);
            return false;
        }
        mixin("project.programType.^ field^ = entry.getText.to!T;");
        return false;
    };
}

template genFocusSignal(tts...)
{
    enum field = tts[0];
    mixin(`alias T = typeof(project.programType.^ field^);`);
    enum genFocusSignal = field^"Entry.addOnFocusOut(tryFillValue!("~T.stringof^,"^ field^")(^ field^
        ↬ Entry));`;

    mixin(genFocusSignal!"progMinSize");
    mixin(genFocusSignal!"progMaxSize");
    mixin(genFocusSignal!"scopeMinSize");
    mixin(genFocusSignal!"scopeMaxSize");
    mixin(genFocusSignal!"newOpGenChance");
    mixin(genFocusSignal!"newScopeGenChance");
    mixin(genFocusSignal!"newLeafGenChance");
    mixin(genFocusSignal!"mutationChangeChance");
    mixin(genFocusSignal!"mutationReplaceChance");
    mixin(genFocusSignal!"mutationDeleteChance");
    mixin(genFocusSignal!"mutationAddLineChance");
    mixin(genFocusSignal!"mutationRemoveLineChance");
    mixin(genFocusSignal!"maxMutationChange");
    mixin(genFocusSignal!"mutationChance");
    mixin(genFocusSignal!"crossingoverChance");
    mixin(genFocusSignal!"copyingPart");
    mixin(genFocusSignal!"deleteMutationRiseGenomeSize");
    mixin(genFocusSignal!"maxGenomeSize");
    mixin(genFocusSignal!"populationSize");
    mixin(genFocusSignal!"graphPermuteChance");
    mixin(genFocusSignal!"graphNodesCountMin");
    mixin(genFocusSignal!"graphNodesCountMax");
    mixin(genFocusSignal!"graphLinksCountMin");
    mixin(genFocusSignal!"graphLinksCountMax");
    mixin(genFocusSignal!"graphPermutatesCountMin");
    mixin(genFocusSignal!"graphPermutatesCountMax");

    mixin(genInitialSetupText!"progMinSize");
    mixin(genInitialSetupText!"progMaxSize");
    mixin(genInitialSetupText!"scopeMinSize");
    mixin(genInitialSetupText!"scopeMaxSize");
    mixin(genInitialSetupText!"newOpGenChance");
    mixin(genInitialSetupText!"newScopeGenChance");
    mixin(genInitialSetupText!"newLeafGenChance");
    mixin(genInitialSetupText!"mutationChangeChance");
    mixin(genInitialSetupText!"mutationReplaceChance");
    mixin(genInitialSetupText!"mutationDeleteChance");
    mixin(genInitialSetupText!"mutationAddLineChance");
    mixin(genInitialSetupText!"mutationRemoveLineChance");
    mixin(genInitialSetupText!"maxMutationChange");
    mixin(genInitialSetupText!"mutationChance");
    mixin(genInitialSetupText!"crossingoverChance");
    mixin(genInitialSetupText!"copyingPart");
    mixin(genInitialSetupText!"deleteMutationRiseGenomeSize");
    mixin(genInitialSetupText!"maxGenomeSize");
    mixin(genInitialSetupText!"populationSize");
    mixin(genInitialSetupText!"graphPermuteChance");
    mixin(genInitialSetupText!"graphNodesCountMin");
    mixin(genInitialSetupText!"graphNodesCountMax");
    mixin(genInitialSetupText!"graphLinksCountMin");
    mixin(genInitialSetupText!"graphLinksCountMax");
    mixin(genInitialSetupText!"graphPermutatesCountMin");
}

```

```

        mixin(genInitialSetupText!"graphPermutatesCountMax");
    }

    void reloadSettings()
    {
        mixin(genEntryGetter!"progMinSize");
        mixin(genEntryGetter!"progMaxSize");
        mixin(genEntryGetter!"scopeMinSize");
        mixin(genEntryGetter!"scopeMaxSize");
        mixin(genEntryGetter!"newOpGenChance");
        mixin(genEntryGetter!"newScopeGenChance");
        mixin(genEntryGetter!"newLeafGenChance");
        mixin(genEntryGetter!"mutationChangeChance");
        mixin(genEntryGetter!"mutationReplaceChance");
        mixin(genEntryGetter!"mutationDeleteChance");
        mixin(genEntryGetter!"mutationAddLineChance");
        mixin(genEntryGetter!"mutationRemoveLineChance");
        mixin(genEntryGetter!"maxMutationChange");
        mixin(genEntryGetter!"mutationChance");
        mixin(genEntryGetter!"crossoverChance");
        mixin(genEntryGetter!"copyingPart");
        mixin(genEntryGetter!"deleteMutationRiseGenomeSize");
        mixin(genEntryGetter!"maxGenomeSize");
        mixin(genEntryGetter!"populationSize");
        mixin(genEntryGetter!"graphPermuteChance");
        mixin(genEntryGetter!"graphNodesCountMin");
        mixin(genEntryGetter!"graphNodesCountMax");
        mixin(genEntryGetter!"graphLinksCountMin");
        mixin(genEntryGetter!"graphLinksCountMax");
        mixin(genEntryGetter!"graphPermutatesCountMin");
        mixin(genEntryGetter!"graphPermutatesCountMax");

        mixin(genInitialSetupText!"progMinSize");
        mixin(genInitialSetupText!"progMaxSize");
        mixin(genInitialSetupText!"scopeMinSize");
        mixin(genInitialSetupText!"scopeMaxSize");
        mixin(genInitialSetupText!"newOpGenChance");
        mixin(genInitialSetupText!"newScopeGenChance");
        mixin(genInitialSetupText!"newLeafGenChance");
        mixin(genInitialSetupText!"mutationChangeChance");
        mixin(genInitialSetupText!"mutationReplaceChance");
        mixin(genInitialSetupText!"mutationDeleteChance");
        mixin(genInitialSetupText!"mutationAddLineChance");
        mixin(genInitialSetupText!"mutationRemoveLineChance");
        mixin(genInitialSetupText!"maxMutationChange");
        mixin(genInitialSetupText!"mutationChance");
        mixin(genInitialSetupText!"crossoverChance");
        mixin(genInitialSetupText!"copyingPart");
        mixin(genInitialSetupText!"deleteMutationRiseGenomeSize");
        mixin(genInitialSetupText!"maxGenomeSize");
        mixin(genInitialSetupText!"populationSize");
        mixin(genInitialSetupText!"graphPermuteChance");
        mixin(genInitialSetupText!"graphNodesCountMin");
        mixin(genInitialSetupText!"graphNodesCountMax");
        mixin(genInitialSetupText!"graphLinksCountMin");
        mixin(genInitialSetupText!"graphLinksCountMax");
        mixin(genInitialSetupText!"graphPermutatesCountMin");
        mixin(genInitialSetupText!"graphPermutatesCountMax");
    }

    void initOperatorsView()
    {
        auto operatorsView = cast(TreeView)builder.getObject("OperatorsView");
        auto operatorNameEntry = cast(Entry)builder.getObject("OperatorNameEntry");
        auto operatorDescriptionView = cast(TextView)builder.getObject("OperatorDescriptionView");

        auto model = new ListStore([GType.STRING]);
        operatorsView.setModel(model);

        operatorsView.addOnCursorChanged((v)
        {
            auto opmng = OperatorMng.getSingleton();
            auto model = operatorsView.getModel();
            assert(model != null);

            auto iter = operatorsView.getSelectedIter();
            if(iter == null)
            {
                operatorNameEntry.setText("");
                operatorDescriptionView.getBuffer().setText("");
            }
            else
            {
                auto value = model.getValue(iter, 0);
                auto opName = value.getString();
                auto operator = opmng.getOperator(opName);

```

```

        operatorNameEntry.setText(operator.name);
        operatorDescriptionView.getBuffer().setText(operator.disrc);
    }
});

auto opmng = OperatorMng.getSingleton();
foreach(operator; OperatorMng.getSingleton)
{
    auto iter = new TreeIter;
    model.insert(iter, -1);
    model.setValue(iter, 0, operator.name);
}
}
}

```

8 Пакет GUI.UTIL

```

module gui.util;

import gtk.ApplicationWindow;
import gtk.Main;
import gtk.TextView;
import gtk.TextIter;

enum AppWindow
{
    Settings,
    Evolution,
    Results
}

void onWindowHideShow(AppWindow type, bool isClosed)
{
    static bool settingsClosed = false;
    static bool evolutionClosed = true;
    static bool resultsClosed = true;

    final switch(type)
    {
        case(AppWindow.Settings): settingsClosed = isClosed; break;
        case(AppWindow.Evolution): evolutionClosed = isClosed; break;
        case(AppWindow.Results): resultsClosed = isClosed; break;
    }

    if(settingsClosed && evolutionClosed && resultsClosed)
    {
        Main.quit();
    }
}

```

9 Пакет GRAPH.CONNECTIVITY

```

module graph.connectivity;

import graph.directed;
import std.algorithm;
import std.range;
import std.array;
import std.conv;

< /**
 * Graph implemented via connectivity lists.
 */
class ConnListGraph : IDirectedGraph
{
    /**
     * Loading graph from raw data.
     */
    void load(InputRange!Edge input)
    {
        foreach(edge; input)
        {
            if(edge.source !in lists)
            {
                Weight[Node] empty;
                lists[edge.source] = empty;
            }
        }
    }
}

```

```

    auto list = lists[edge.source];
    if(edge.dist in list)
    {
        assert(edge.weight == list[edge.dist]);
    }
    list[edge.dist] = edge.weight;
    lists[edge.source] = list;
}
}

/**
 * Returns graph nodes set
 */
InputRange!string nodes()
{
    bool[Node] nodes;

    foreach(source, list; lists)
    {
        nodes[source] = true;
        foreach(dist, weight; list)
        {
            nodes[dist] = true;
        }
    }

    return nodes.keys.inputRangeObject;
}

/**
 * Returns graph weights set
 */
InputRange!string weights()
{
    bool[Weight] weights;

    foreach(source, list; lists)
    {
        foreach(dist, weight; list)
        {
            weights[weight] = true;
        }
    }
    return weights.keys.inputRangeObject;
}

/**
 * Returns graph edges set
 */
InputRange!Edge edges()
{
    auto builder = appender!(Edge[]);

    foreach(source, list; lists)
    {
        foreach(dist, weight; list)
        {
            builder.put(Edge(source, dist, weight));
        }
    }

    return builder.data.inputRangeObject;
}

/**
 * Returns indexed graph edges set
 */
InputRange!IndexedEdge indexedEdges()
{
    auto builder = appender!(IndexedEdge[]);
    auto nodesArr = nodes.array;

    foreach(edge; edges)
    {
        builder.put(IndexedEdge(
            nodesArr.countUntil(edge.source),
            nodesArr.countUntil(edge.dist) ));
    }

    return builder.data.inputRangeObject;
}

string genDot()
{
    Weight[Node] getFirst(out Node node)

```

```

    {
        foreach(k, list; lists)
        {
            node = k;
            return list;
        }
        assert(false);
    }

    string genNodeName(size_t i)
    {
        if(i >= 1000)
        {
            return text("n",i);
        } else if(i >= 100)
        {
            return text("n0",i);
        } else if(i >= 10)
        {
            return text("n00",i);
        } else
        {
            return text("n00",i);
        }
    }

    string genForList(ref size_t i, Node node, Weight[Node] list, ref size_t[Node] nodeMap)
    {
        auto builder = appender!string;

        string nodeName;
        if(node !in nodeMap)
        {
            nodeName = genNodeName(i);
            nodeMap[node] = i;
            i+=1;

            builder.put(nodeName);
            builder.put(" ;\n");

            builder.put(nodeName);
            builder.put('[' label="');
            builder.put(node.tostring());
            builder.put(']' ;'"'\n");
        } else
        {
            nodeName = genNodeName(nodeMap[node]);
        }

        foreach(dist, weight; list)
        {
            bool genLabel = false;
            if(dist !in nodeMap)
            {
                nodeMap[dist] = i;
                i+=1;
                genLabel = true;
            }
            builder.put(nodeName);
            builder.put(" -> ");
            builder.put(genNodeName(nodeMap[dist]));
            builder.put('[' label = '');
            builder.put(weight.tostring());
            builder.put(']' ;');
            builder.put("\n");

            if(genLabel)
            {
                builder.put(genNodeName(nodeMap[dist]));
                builder.put('[' label="');
                builder.put(dist.tostring());
                builder.put('"' ] ;'"'\n");
            }
        }
        return builder.data;
    }

    auto builder = appender!string;

    builder.put('digraph "' {'"\n");
    size_t[Node] nodeMap;

    size_t i = 0;
    foreach(node, list; lists)
    {
        builder.put(genForList(i, node, list, nodeMap));
    }
}

```

```

        }

        builder.put('}' + "\n");

        return builder.data;
    }

    private
    {
        Weight[Node][Node] lists;
    }
}

unittest
{
    import std.algorithm;

    auto edges =
        IDirectedGraph.Edge("a", "b", "1"),
        IDirectedGraph.Edge("b", "a", "2")
    ];

    auto graph = new ConnListGraph();
    graph.load(edges.inputRangeObject);

    assert(graph.nodes.array.sort.equal(["a", "b"]));
    assert(graph.weights.array.sort.equal(["1", "2"]));
}

unittest
{
    import std.process;
    import std.stdio;

    auto edges =
        IDirectedGraph.Edge("a", "b", "1"),
        IDirectedGraph.Edge("b", "a", "2"),
        IDirectedGraph.Edge("c", "a", "3"),
        IDirectedGraph.Edge("c", "b", "1")
    ];

    auto graph = new ConnListGraph();
    graph.load(edges.inputRangeObject);

    auto file = File("test.dot", "w");
    file.writeln(graph.genDot);
    file.close();

    shell("dot -Tpng test.dot > test.png");
    shell("gwenview test.png");
}
}

```

10 Пакет GRAPH.DIRECTED

```

module graph.directed;

import std.range;
import std.conv;

/**
* Generic interface for directed graph. Nodes are marked with
* strings, edges are marked with strings too.
*/
interface IDirectedGraph
{
    /**
     * Unpacked graph edge.
     */
    struct Edge
    {
        /// Edge source
        string source;
        /// Edge dist
        string dist;
        /// Edge weight
        string weight;

        string toString()
        {
            if(weight == "")
            {
                return source ~ " -> " ~ dist;
            }
        }
    }
}

```

```

        else
        {
            return source ~ " - " ~ weight ~ " -> " ~ dist;
        }
    }

< /**
 * Operating with indexes is more handy for genetic programs.
 */
struct IndexedEdge
{
    size_t source;
    size_t dist;

    string toString()
    {
        return text(source, " -> ", dist);
    }
}

alias string Node;
alias string Weight;

< /**
 * Loading graph from raw data.
 */
void load(InputRange!Edge input);

< /**
 * Returns graph nodes set
 */
InputRange!Node nodes();

< /**
 * Returns graph weights set
 */
InputRange!Weight weights();

< /**
 * Returns graph edges set
 */
InputRange!Edge edges();

< /**
 * Returns indexed graph edges set
 */
InputRange!IndexedEdge indexedEdges();

< /**
 * Generates dot description for
 * visualization.
 */
string genDot();
}

```

11 Пакет EVOL.COMPLIER

```

module evol.compiler;

import devol.compiler;
import devol.population;

import evol.individ;
import evol.world;
import evol.progtype;

import project;

alias Population!( getDefChars, GraphIndivid ) GraphPopulation;

class GraphCompilation : GameCompilation
{
    Project project;

    this( Project project, void delegate() updateGenerationInfo )
    {
        this.project = project;
        this.updateGenerationInfo = updateGenerationInfo;
    }

    bool stopCond(ref int step, IndAbstract ind, WorldAbstract world)

```

```

    {
        return step >= 1;
    }

void drawStep(IndAbstract ind, WorldAbstract world)
{
}

void drawFinal(PopAbstract pop, WorldAbstract world)
{
    project.population = cast(GraphPopulation)pop;
    updateGenerationInfo();
}

int roundsPerInd()
{
    return 10;
}

private void delegate() updateGenerationInfo;
}

```

```

alias GraphCompiler = Compiler!(
    GraphCompilation
    , Evolutor
    , ProgramType
    , GraphPopulation
    , GraphWorld);

```

12 Пакет EVOL.INDIVID

```

module evol.individ;

import devol.individ;
import devol.argument;
import devol.std.argvoid;

import dyaml.all;

import std.container;

import evol.types.argedge;
import evol.world;
import devol.std.argpod;

class GraphIndivid : Individ
{
    this()
    {

    }

    this(Individ ind)
    {
        this();
        loadFrom(ind);
    }

    override void initialize(WorldAbstract aworld)
    {
        super.initialize(aworld);

        GraphWorld world = cast(GraphWorld)aworld;
        assert(world);

        mStack.mStack.clear();

        mFirstGraph.mStack.clear();
        foreach(edge; world.firstGraph.indexedEdges)
        {
            mFirstGraph.stackPush(new ArgEdge(edge));
        }

        mSecondGraph.mStack.clear();
        foreach(edge; world.secondGraph.indexedEdges)
        {
            mSecondGraph.stackPush(new ArgEdge(edge));
        }
    }
}

```

```

override @property GraphIndivid dup()
{
    auto ind = new GraphIndivid();
    ind.mFitness = mFitness;

    ind.mProgram = [];
    foreach(line; mProgram)
        ind.mProgram ~= line.dup;

    ind.mMemory = [];
    foreach(line; mMemory)
        ind.mMemory ~= line.dup;

    ind.inVals = [];
    foreach(line; inVals)
        ind.inVals ~= line.dup;

    ind.outVals = [];
    foreach(line; outVals)
        ind.outVals ~= line.dup;

    return ind;
}

static GraphIndivid loadYaml(Node node)
{
    auto ind = Individ.loadYaml(node);
    auto ant = new GraphIndivid();
    ant.mFitness = ind.fitness;

    foreach(line; ind.program)
        ant.mProgram ~= line.dup;
    foreach(line; ind.memory)
        ant.mMemory ~= line.dup;
    foreach(line; ind.invals)
        ant.inVals ~= line.dup;
    foreach(line; ind.outvals)
        ant.outVals ~= line.dup;

    return ant;
}

struct Stack(T)
{
    void stackPush(T arg)
    {
        mStack.insertFront = arg;
    }

    T stackPop()
    {
        if(mStack.empty)
        {
            return new T;
        }
        else
        {
            auto arg = mStack.front;
            mStack.removeFront();
            return arg;
        }
    }

    void stackSwap()
    {
        if(mStack.empty) return;

        auto a1 = mStack.front;
        mStack.removeFront();

        if(mStack.empty)
        {
            mStack.insertFront = a1;
        }
        else
        {
            auto a2 = mStack.front;
            mStack.removeFront();

            mStack.insertFront = a1;
            mStack.insertFront = a2;
        }
    }

    void stackDup()
    {
        if(mStack.empty) return;
    }
}

```

```

        mStack.insertFront = mStack.front;
    }

void stackOver()
{
    if(mStack.empty) return;

    auto a1 = mStack.front;
    mStack.removeFront;

    if(mStack.empty)
    {
        mStack.insertFront = a1;
    } else
    {
        auto a2 = mStack.front;
        mStack.removeFront;

        mStack.insertFront = a2;
        mStack.insertFront = a1;
        mStack.insertFront = a2;
    }
}

void stackRot()
{
    if(mStack.empty) return;

    auto a1 = mStack.front;
    mStack.removeFront;

    if(mStack.empty)
    {
        mStack.insertFront = a1;
    } else
    {
        auto a2 = mStack.front;
        mStack.removeFront;

        if(mStack.empty)
        {
            mStack.insertFront = a2;
            mStack.insertFront = a1;
        } else
        {
            auto a3 = mStack.front;
            mStack.removeFront;

            mStack.insertFront = a2;
            mStack.insertFront = a1;
            mStack.insertFront = a3;
        }
    }
}

void stackDrop()
{
    if(mStack.empty) return;

    mStack.removeFront;
}

DList!T mStack;
}

ref Stack!(ArgPod!double) genericStack()
{
    return mStack;
}

ref Stack!ArgEdge firstGraphStack()
{
    return mFirstGraph;
}

ref Stack!ArgEdge secondGraphStack()
{
    return mSecondGraph;
}

void answer(bool value)
{
    mAnswer = value;
}

```

```

bool answer()
{
    return mAnswer;
}

private
{
    bool mAnswer = false;
    Stack!(ArgPod!double) mStack;
    Stack!ArgEdge mFirstGraph;
    Stack!ArgEdge mSecondGraph;
}
}

```

13 Пакет EVOL.PROGTYPE

```

module evol.progtype;

import devol.programtype;
import devol.typemng;
import devol.operatormng;

import devol.std.typepod;
import std.conv;
import std.range;
import std.math;

import evol.operators.and;
import evol.operators.not;
import evol.operators.opif;
import evol.operators.opwhile;
import evol.operators.or;
import evol.operators.plus;
import evol.operators.mult;
import evol.operators.div;
import evol.operators.relation;
import evol.operators.gpop;
import evol.operators.gpush;
import evol.operators.gdup;
import evol.operators.gover;
import evol.operators.grot;
import evol.operators.gswap;
import evol.operators.ipop;
import evol.operators.ipush;
import evol.operators.idup;
import evol.operators.lover;
import evol.operators.irot;
import evol.operators.iswap;
import evol.operators.construct;
import evol.operators.dist;
import evol.operators.source;
import evol.operators.idcast;
import evol.operators.round;
import evol.operators.answer;

import evol.types.typeedge;
import evol.individ;
import evol.world;

import std.algorithm;
import std.range;

class ProgramType : ProgTypeAbstract
{
    this()
    {
        registerTypes();
    }

    void registerTypes()
    {
        auto tmng = TypeMng.getSingleton();
        auto omng = OperatorMng.getSingleton();

        auto types = tmng.strings;
        if(types.find("Typebool").empty)
        {
            tmng.registerType!TypeBool();
        }
        if(types.find("Typeint").empty)
        {
            tmng.registerType!TypeInt();
        }
    }
}

```

```

        }

        if(types.find("Typedouble").empty)
        {
            tmng.registerType!TypeDouble();
        }
        if(types.find("TypeEdge").empty)
        {
            tmng.registerType!TypeEdge();
        }

        auto ops = omng.strings;
        void registerOperator(T)(string name)
        {
            assert(name != "");
            if(ops.find(name).empty)
            {
                omng.registerOperator!T();
            }
        }

        registerOperator!IfOperator("if");
        registerOperator!WhileOperator("while");
        registerOperator!AndOperator("&&");
        registerOperator!OrOperator("||");
        registerOperator!NotOperator("!");

        registerOperator!PlusOperator("+");
        registerOperator!MultOperator("*");
        registerOperator!DivOperator("/");

        registerOperator!IntEqualOperator("== (int)");
        registerOperator!IntGreaterOperator("> (int)");
        registerOperator!IntLesserOperator("< (int)");
        registerOperator!IntGreaterEqualOperator(">= (int)");
        registerOperator!IntLesserEqualOperator("<= (int)");

        registerOperator!DoubleEqualOperator("== (double)");
        registerOperator!DoubleGreaterOperator("> (double)");
        registerOperator!DoubleLesserOperator("< (double)");

        registerOperator!GenericPopOperator("gpop");
        registerOperator!GenericPushOperator("gpush");
        registerOperator!GenericDupOperator("gdup");
        registerOperator!GenericOverOperator("gover");
        registerOperator!GenericRotOperator("grot");
        registerOperator!GenericSwapOperator("gswap");

        registerOperator!InputPopFirstOperator("ipop1");
        registerOperator!InputPushFirstOperator("ipush1");
        registerOperator!InputDupFirstOperator("idup1");
        registerOperator!InputOverFirstOperator("iover1");
        registerOperator!InputRotFirstOperator("irot1");
        registerOperator!InputSwapFirstOperator("iswap1");

        registerOperator!InputPopSecondOperator("ipop2");
        registerOperator!InputPushSecondOperator("ipush2");
        registerOperator!InputDupSecondOperator("idup2");
        registerOperator!InputOverSecondOperator("iover2");
        registerOperator!InputRotSecondOperator("irot2");
        registerOperator!InputSwapSecondOperator("iswap2");

        registerOperator!ConstructOperator("construct");
        registerOperator!GetSourceOperator("getSource");
        registerOperator!GetDistOperator("getDist");

        registerOperator!IntDoubleCastOperator("cast");
        registerOperator!RoundOperator("round");
        registerOperator!AnswerOperator("answer");
    }

    private uint mProgMinSize = 4;
    @property uint progMinSize()
    {
        return mProgMinSize;
    }

    @property void progMinSize(uint val)
    {
        mProgMinSize = val;
    }

    private uint mProgMaxSize = 8;
    @property uint progMaxSize()
    {
        return mProgMaxSize;
    }
}

```

```

@property void progMaxSize(uint val)
{
    mProgMaxSize = val;
}

private float mNewOpGenChacne = 0.3;
@property float newOpGenChance()
{
    return mNewOpGenChacne;
}

@property void newOpGenChance(float val)
{
    mNewOpGenChacne = val;
}

private float mNewScopeGenChance = 0.1;
@property float newScopeGenChance()
{
    return mNewScopeGenChance;
}

@property void newScopeGenChance(float val)
{
    mNewScopeGenChance = val;
}

private float mNewLeafGenChance = 0.6;
@property float newLeafGenChance()
{
    return mNewLeafGenChance;
}

@property void newLeafGenChance(float val)
{
    mNewLeafGenChance = val;
}

private uint mScopeMinSize = 2;
@property uint scopeMinSize()
{
    return mScopeMinSize;
}

@property void scopeMinSize(uint val)
{
    mScopeMinSize = val;
}

private uint mScopeMaxSize = 5;
@property uint scopeMaxSize()
{
    return mScopeMaxSize;
}

@property void scopeMaxSize(uint val)
{
    mScopeMaxSize = val;
}

private float mMutationChance = 0.3;
@property float mutationChance()
{
    return mMutationChance;
}

@property void mutationChance(float val)
{
    mMutationChance = val;
}

private float mCrossingoverChance = 0.7;
@property float crossingoverChance()
{
    return mCrossingoverChance;
}

@property void crossingoverChance(float val)
{
    mCrossingoverChance = val;
}

private float mMutationChangeChance = 0.5;
@property float mutationChangeChance()
{
}

```

```

        return mMutationChangeChance;
    }

@property void mutationChangeChance(float val)
{
    mMutationChangeChance = val;
}

private float mMutationReplaceChance = 0.3;
@property float mutationReplaceChance()
{
    return mMutationReplaceChance;
}

@property void mutationReplaceChance(float val)
{
    mMutationReplaceChance = val;
}

private float mMutationDeleteChance = 0.2;
@property float mutationDeleteChance()
{
    return mMutationDeleteChance;
}

@property void mutationDeleteChance(float val)
{
    mMutationDeleteChance = val;
}

private float mMutationAddLineChance = 0.1;
@property float mutationAddLineChance()
{
    return mMutationAddLineChance;
}

@property void mutationAddLineChance(float val)
{
    mMutationAddLineChance = val;
}

private float mMutationRemoveLineChance = 0.05;
@property float mutationRemoveLineChance()
{
    return mMutationRemoveLineChance;
}

@property void mutationRemoveLineChance(float val)
{
    mMutationRemoveLineChance = val;
}

private string mMaxMutationChange = "100";
@property string maxMutationChange()
{
    return mMaxMutationChange;
}

@property void maxMutationChange(string val)
{
    mMaxMutationChange = val;
}

private float mCopyingPart = 0.1;
@property float copyingPart()
{
    return mCopyingPart;
}

@property void copyingPart(float val)
{
    mCopyingPart = val;
}

private size_t mDeleteMutationRiseGenomeSize = 200;
@property size_t deleteMutationRiseGenomeSize()
{
    return mDeleteMutationRiseGenomeSize;
}

@property void deleteMutationRiseGenomeSize(size_t val)
{
    mDeleteMutationRiseGenomeSize = val;
}

private size_t mMaxGenomeSize = 300;

```

```

@property size_t maxGenomeSize()
{
    return mMaxGenomeSize;
}

@property void maxGenomeSize(size_t val)
{
    mMaxGenomeSize = val;
}

private size_t mPopulationSize = 10;
@property size_t populationSize()
{
    return mPopulationSize;
}

@property void populationSize(size_t val)
{
    mPopulationSize = val;
}

private double mGraphPermuteChance = 0.5;
@property double graphPermuteChance()
{
    return mGraphPermuteChance;
}

@property void graphPermuteChance(double val)
in
{
    assert(0.0 <= val && val <= 1.1, "Not a chance!");
}
body
{
    mGraphPermuteChance = val;
}

private size_t mGraphNodesCountMin = 3;
@property size_t graphNodesCountMin()
{
    return mGraphNodesCountMin;
}

@property void graphNodesCountMin(size_t val)
in
{
    assert(val <= mGraphNodesCountMax, "Must be <= graphNodesCountMax");
}
body
{
    mGraphNodesCountMin = val;
}

private size_t mGraphNodesCountMax = 10;
@property size_t graphNodesCountMax()
{
    return mGraphNodesCountMax;
}

@property void graphNodesCountMax(size_t val)
in
{
    assert(val >= mGraphNodesCountMin, "Must be >= graphNodesCountMin");
}
body
{
    mGraphNodesCountMax = val;
}

private size_t mGraphLinksCountMin = 3;
@property size_t graphLinksCountMin()
{
    return mGraphLinksCountMin;
}

@property void graphLinksCountMin(size_t val)
in
{
    assert(val <= mGraphLinksCountMax, "Must be <= graphLinksCountMax");
}
body
{
    mGraphLinksCountMin = val;
}

private size_t mGraphLinksCountMax = 6;

```

```

@property size_t graphLinksCountMax()
{
    return mGraphLinksCountMax;
}

@property void graphLinksCountMax(size_t val)
in
{
    assert(val >= mGraphLinksCountMin, "Must be >= graphLinksCountMin");
}
body
{
    mGraphLinksCountMax = val;
}

private size_t mGraphPermutatesCountMin = 2;
@property size_t graphPermutatesCountMin()
{
    return mGraphPermutatesCountMin;
}

@property void graphPermutatesCountMin(size_t val)
in
{
    assert(val <= mGraphPermutatesCountMax, "Must be <= graphPermutatesCountMax");
}
body
{
    mGraphPermutatesCountMin = val;
}

private size_t mGraphPermutatesCountMax = 4;
@property size_t graphPermutatesCountMax()
{
    return mGraphPermutatesCountMax;
}

@property void graphPermutatesCountMax(size_t val)
in
{
    assert(val >= mGraphPermutatesCountMin, "Must be >= graphPermutatesCountMin");
}
body
{
    mGraphPermutatesCountMax = val;
}

Line[] initValues(WorldAbstract pWorld)
{
    return new Line[0];
}

double getFitness(IndAbstract pInd, WorldAbstract pWorld, double time)
{
    auto ind = cast(GraphIndivid)pInd;
    auto world = cast(GraphWorld)pWorld;
    assert(ind);
    assert(world);

    size_t n = world.firstGraph.nodes.walkLength + world.secondGraph.nodes.walkLength;
    enum tPerNode = 0.1;
    double ft = cast(double)(1.0 / (1.0 + exp( 5.0 / (tPerNode*n) * time - 5.0)));
    double fa = ind.answer == world.correctAnswer ? 1.0 : 0.0;

    if(fa < 0.9) ft = 0.0;

    return (ft + fa) / 2;
}
}

```

14 Пакет EVOL.WORLD

```

module evol.world;

import devol.world;

import std.algorithm;
import std.array;
import std.random;
import std.range;
import std.file;

```

```

import std::stdio;
import std::path;
import std::process;

import graph::directed;
import graph::connectivity;

import evol::proctype;

class GraphWorld : WorldAbstract
{
    ProgramType programType;

    this()
    {
        assert(false);
    }

    this(ProgramType programType,
         void delegate(IDirectedGraph, IDirectedGraph) updateDrawDel)
    {
        this.programType = programType;
        this.updateDrawDel = updateDrawDel;
    }

    void initialize()
    {
        genUniqName(true);
        initInput();

        updateDrawDel(mInputGraphFirst, mInputGraphSecond);
    }

    IDirectedGraph firstGraph()
    {
        return mInputGraphFirst;
    }

    IDirectedGraph secondGraph()
    {
        return mInputGraphSecond;
    }

    bool correctAnswer()
    {
        return mAnswer;
    }

private
{
    void delegate(IDirectedGraph, IDirectedGraph) updateDrawDel;
    IDirectedGraph mInputGraphFirst;
    IDirectedGraph mInputGraphSecond;
    bool mAnswer;

    void initInput()
    {
        mInputGraphFirst = generateGraph(
            uniform
            , uniform);
        if(getChance(programType.graphPermuteChance))
        {
            mInputGraphSecond
                = permuteGraph(mInputGraphFirst
                    , uniform);
            mAnswer = true;
        } else
        {
            mInputGraphSecond
                = generateGraph(
                    uniform
                    , uniform);

            mAnswer = false;
        }
    }

    static bool getChance(float val)
    {
        return uniform <= val;
    }

    static string genUniqName(bool clearMemory = false)
    {
        static bool[string] memory;

```

```

    if(clearMemory)
    {
        bool[string] clean;
        memory = clean;
        return "";
    }

    immutable alphabet = "qwertyuiopasdfghjklzxcvbnm";
    string genString(size_t l)
    {
        auto builder = appender!string;
        foreach(i; 0..l)
            builder.put(alphabet[uniform(0,alphabet.length)]);
        return builder.data;
    }

    size_t i = 1;
    while(true)
    {
        string name = genString(i);
        if(name in memory)
        {
            i++;
        } else
        {
            return name;
        }
    }
}

static IDirectedGraph generateGraph(size_t nodesCount, size_t linksCount)
{
    auto nodesBuilder = appender!(string[]);
    foreach(i; 0..nodesCount)
    {
        nodesBuilder.put(genUniqName());
    }

    auto nodes = nodesBuilder.data;
    auto edgeBuilder = appender!(IDirectedGraph.Edge[]);
    foreach(i; 0..linksCount)
    {
        size_t a = uniform(0, nodesCount);
        size_t b = uniform(0, nodesCount);
        edgeBuilder.put(IDirectedGraph.Edge(nodes[a], nodes[b], ""));
    }

    auto graph = new ConnListGraph;
    graph.load(edgeBuilder.data[].inputRangeObject);
    return graph;
}

static IDirectedGraph permuteGraph(IDirectedGraph graph, size_t permuteCount)
{
    auto nodes = graph.nodes.array;
    IDirectedGraph.Edge[] edges;

    foreach(i; 0..permuteCount)
    {
        auto builder = appender!(IDirectedGraph.Edge[]);
        auto a = nodes[uniform(0, nodes.length)];
        auto b = nodes[uniform(0, nodes.length)];

        foreach(edge; graph.edges)
        {
            if(a != b)
            {
                if(edge.source == a) edge.source = b;
                else if(edge.source == b) edge.source = a;

                if(edge.dist == a) edge.dist = b;
                else if(edge.dist == b) edge.dist = a;
            }

            builder.put(edge);
        }

        edges = builder.data;
    }

    auto newGraph = new ConnListGraph;
    newGraph.load(edges.inputRangeObject);
    return newGraph;
}
}

```

15 Пакет EVOL.TYPES.ARGEEDGE

```
module evol.types.argedge;

import std.conv;
import std.random;
import devol.serializable;
import devol.typeMng;

import dyaml.all;

import graph.directed;

class ArgEdge : Argument, ISerializable
{
    this()
    {
        super( TypeMng.getSingleton().getType("TypeEdge") );
    }

    this(IDirectedGraph.IndexedEdge edge)
    {
        this();
        opAssign(edge);
    }

    ref ArgEdge opAssign(Argument val)
    {
        auto arg = cast(ArgEdge)(val);
        if (arg is null) return this;

        mEdge = arg.mEdge;
        return this;
    }

    ref ArgEdge opAssign(IDirectedGraph.IndexedEdge val)
    {
        mEdge = val;
        return this;
    }

    override @property string toString(uint depth=0)
    {
        return to!string(mEdge);
    }

    @property IDirectedGraph.IndexedEdge edge()
    {
        return mEdge;
    }

    @property IDirectedGraph.IndexedEdge val()
    {
        return mEdge;
    }

    override void randomChange()
    {
        size_t permuteIndex(size_t i)
        {
            int change = uniform;
            if(cast(int)i + change < 0) return 0;
            return i + change;
        }

        auto chance = uniform;
        if(chance == 0)
        {
            mEdge.source = permuteIndex(mEdge.source);
        } else if(chance == 1)
        {
            mEdge.dist = permuteIndex(mEdge.dist);
        }
    }

    override void randomChange(string maxChange)
    {
        randomChange();
    }

    override @property Argument dup()
    {
```

```

        auto darg = new ArgEdge();
        darg.mEdge = mEdge;
        return darg;
    }

    void saveBinary(OutputStream stream)
    {
        assert(false, "Not implemented!");
    }

    override Node saveYaml()
    {
        return Node([
            "class": Node("plain"),
            "source": Node(mEdge.source),
            "dist": Node(mEdge.dist),
        ]);
    }

    protected IDirectedGraph.IndexedEdge mEdge;
}

```

16 Пакет EVOL.TYPES.TYPEEDGE

```

module evol.types.typeedge;

import std.stream;

public
{
    import devol.argument;
    import devol.type;
    import evol.types.argedge;
}

import dyaml.all;

import graph.directed;

class TypeEdge : Type
{
    this()
    {
        super("TypeEdge");
    }

    override ArgEdge getNewArg()
    {
        auto arg = new ArgEdge();
        foreach(i; 0..10)
            arg.randomChange();

        return arg;
    }

    override ArgEdge getNewArg(string min, string max, string[] exVal)
    {
        return getNewArg();
    }

    override Argument loadArgument(InputStream stream)
    {
        assert(false, "Not implemented!");
    }

    override Argument loadArgument(Node node)
    {
        return new ArgEdge(IDirectedGraph.IndexedEdge(node["source"].as!size_t, node["dist"].as!size_t));
    }
}

```

17 Пакет EVOL.OPERATORS.AND

```

module evol.operators.and;

import devol.world;
import devol.std.line;
import devol.individ;

```

```

import devol.operator;
import devol.type;
import devol.typeMng;
import devol.std.argpod;
import devol.std.typepod;
import devol.argument;

class AndOperator : Operator
{
    TypePod!bool booltpe;

    this()
    {
        booltpe = cast(TypePod!bool)TypeMng.getSingleton().getType("Typebool");
        mRetType = booltpe;

        super("&&", "Логическое И", "для значений ложь истина", ., ArgsStyle.BINAR_STYLE);

        ArgInfo a1;
        a1.type = booltpe;
        args ~= a1;
        args ~= a1;
    }

    override Argument apply(IndAbstract individ, Line line, WorldAbstract world)
    {
        auto ret = booltpe.getNewArg();

        auto a1 = cast(ArgPod!bool)line[0];
        auto a2 = cast(ArgPod!bool)line[1];

        assert(a1 != null);
        assert(a2 != null);

        ret = a1.val && a2.val;
        return ret;
    }
}

```

18 Пакет EVOL.OPERATORS.ANSWER

```

module evol.operators.answer;

import devol.typeMng;

import devol.individ;
import devol.world;
import devol.operator;
import devol.std.typepod;

import evol.individ;

class AnswerOperator : Operator
{
    TypePod!bool booltpe;
    TypeVoid voidtype;

    enum description = "Записывает" ответ. True – графы изоморфны, False – неизоморфны .";

    this()
    {
        booltpe = cast(TypePod!bool)(TypeMng.getSingleton().getType("Typebool"));
        assert(booltpe, "We need bool type!");

        voidtype = cast(TypeVoid)(TypeMng.getSingleton().getType("TypeVoid"));
        assert(voidtype);

        mRetType = voidtype;
        super("answer", description, ArgsStyle.UNAR_STYLE);

        ArgInfo a1;
        a1.type = booltpe;
        args ~= a1;
    }

    override Argument apply(IndAbstract aind, Line line, WorldAbstract world)
    {
        auto ind = cast(GraphIndivid)aind;
        assert(ind);

        auto cond = cast(ArgPod!bool)(line[0]);
        assert(cond);
    }
}

```

```

        ind.answer = cond.val;

    return new ArgVoid;
}
}

```

19 Пакет EVOL.OPERATORS.CONSTRUCT

```

module evol.operators.construct;

import std.stdio;

import devol.typeMng;

public
{
    import devol.individ;
    import devol.world;
    import devol.operator;
    import devol.std.typevoid;
    import devol.std.argvoid;
    import devol.std.typepod;

    import evol.types.typeedge;
    import evol.types.argedge;
}

import evol.individ;
import graph.directed;

class ConstructOperator : Operator
{
    TypeEdge edgetype;
    TypePod!int inttype;

    enum description = "Создает" novoerrebografai z dvuh indexov : начала иконца .";

    this()
    {
        edgetype = cast(TypeEdge)(TypeMng.getSingleton().getType("TypeEdge"));
        assert(edgetype, "We need edge type!");

        inttype = cast(TypePod!int)(TypeMng.getSingleton().getType("Typeint"));
        assert(inttype);

        mRetType = edgetype;
        super("construct", description, ArgsStyle.CLASSIC_STYLE);

        ArgInfo a1;
        a1.type = inttype;
        a1.max = "20";
        a1.min = "0";

        args ~= a1;
        args ~= a1;
    }

    override Argument apply(IndAbstract ind, Line line, WorldAbstract world)
    {
        auto gind = cast(GraphIndivid)ind;
        assert(gind);

        auto a1 = cast(ArgPod!int)(line[0]);
        auto a2 = cast(ArgPod!int)(line[0]);
        assert(a1);
        assert(a2);

        return new ArgEdge(IDirectedGraph.IndexedEdge(a1.val, a2.val));
    }
}

```

20 Пакет EVOL.OPERATORS.DIST

```

module evol.operators.dist;

import std.stdio;

```

```

import devol.typeMng;

public
{
    import devol.individ;
    import devol.world;
    import devol.operator;
    import devol.std.typevoid;
    import devol.std.argvoid;
    import devol.std.typepod;

    import evol.types.typeedge;
    import evol.types.argedge;
}

import evol.individ;

class GetDistOperator : Operator
{
    TypeEdge edgetype;
    TypePod!int inttype;

    enum description = "Возвращает" индексвершины , вкоторуюприходитребографа .";

    this()
    {
        edgetype = cast(TypeEdge)(TypeMng.getSingleton().getType("TypeEdge"));
        assert(edgetype, "We need edge type!");

        inttype = cast(TypePod!int)(TypeMng.getSingleton().getType("Typeint"));
        assert(inttype);

        mRetType = inttype;
        super("getDist", description, ArgsStyle.UNAR_STYLE);

        ArgInfo a1;
        a1.type = edgetype;

        args ^= a1;
    }

    override Argument apply(IndAbstract ind, Line line, WorldAbstract world)
    {
        auto gind = cast(GraphIndivid)ind;
        assert(gind);

        auto a1 = cast(ArgEdge)(line[0]);
        assert(a1);

        return new ArgPod!int(cast(int)a1.edge.dist);
    }
}

```

21 Пакет EVOL.OPERATORS.DIV

```

module evol.operators.div;

import std.stdio;

import devol.typeMng;

public
{
    import devol.individ;
    import devol.world;
    import devol.operator;
    import devol.std.typepod;
}

class DivOperator : Operator
{
    TypePod!double doubletype;

    enum description = "Арифметическая" операцияделениядействительныхчисел .;

    this()
    {
        doubletype = cast(TypePod!double)(TypeMng.getSingleton().getType("Typedouble"));
        assert(doubletype, "We need double type!");

        mRetType = doubletype;
        super("/", description, ArgsStyle.BINAR_STYLE);
    }
}

```

```

        ArgInfo a1;
        a1.type = doubletype;
        a1.min = "-1000";
        a1.max = "+1000";

        args ~= a1;
        args ~= a1;
    }

    override Argument apply(IndAbstract ind, Line line, WorldAbstract world)
    {
        auto ret = doubletype.getNewArg();

        auto a1 = cast(ArgPod!double)(line[0]);
        auto a2 = cast(ArgPod!double)(line[1]);

        assert( a1 != null, "Critical error: Operator plus, argument 1 isn't a right value!");
        assert( a2 != null, "Critical error: Operator plus, argument 2 isn't a right value!");

        ret = a1.val / a2.val;
        return ret;
    }
}

```

22 Пакет EVOL.OPERATORS.GDUP

```

module evol.operators.gdup;

import std.stdio;

import devol.typemng;

public
{
    import devol.individ;
    import devol.world;
    import devol.operator;
    import devol.std.typevoid;
    import devol.std.argvoid;
}

import evol.individ;

class GenericDupOperator : Operator
{
    TypeVoid voidtype;

    enum description = "Дублирует" голову стека общего назначения .";

    this()
    {
        voidtype = cast(TypeVoid)(TypeMng.getSingleton().getType("TypeVoid"));
        assert(voidtype, "We need double type!");
    }

    mRetType = voidtype;
    super("gdup", description, ArgsStyle.NULAR_STYLE);
}

override Argument apply(IndAbstract ind, Line line, WorldAbstract world)
{
    auto gind = cast(GraphIndivid)ind;
    assert(gind);

    gind.genericStack.stackDup;
    return new ArgVoid;
}
}

```

23 Пакет EVOL.OPERATORS.GOVER

```

module evol.operators.gover;

import std.stdio;

import devol.typemng;

public

```

```

{
    import devol.individ;
    import devol.world;
    import devol.operator;
    import devol.std.typevoid;
    import devol.std.argvoid;
}

import evol.individ;

class GenericOverOperator : Operator
{
    TypeVoid voidtype;

    enum description = "Дублирует" значениеподголовойстеканавершину . Стекобщегоназначения .";

    this()
    {
        voidtype = cast(TypeVoid)(TypeMng.getSingleton().getType("TypeVoid"));
        assert(voidtype, "We need double type!");
    }

    mRetType = voidtype;
    super("gover", description, ArgsStyle.NULAR_STYLE);
}

override Argument apply(IndAbstract ind, Line line, WorldAbstract world)
{
    auto gind = cast(GraphIndivid)ind;
    assert(gind);

    gind.genericStack.stackOver;
    return new ArgVoid();
}
}

```

24 Пакет EVOL.OPERATORS.GPOP

```

module evol.operators.gpop;

import std.stdio;

import devol.typeMng;

public
{
    import devol.individ;
    import devol.world;
    import devol.operator;
    import devol.std.typePod;
}

import evol.individ;

class GenericPopOperator : Operator
{
    TypePod!double doubletype;

    enum description = "Взятие" головысостекаобщегоназначения .;

    this()
    {
        doubletype = cast(TypePod!double)(TypeMng.getSingleton().getType("Typedouble"));
        assert(doubletype, "We need double type!");

        mRetType = doubletype;
        super("gpop", description, ArgsStyle.NULAR_STYLE);
    }

    override Argument apply(IndAbstract ind, Line line, WorldAbstract world)
    {
        auto gind = cast(GraphIndivid)ind;
        assert(gind);

        return gind.genericStack.stackPop();
    }
}

```

25 Пакет EVOL.OPERATORS.GPUSH

```

module evol.operators.gpush;

import std.stdio;

import devol.typeMng;

public
{
    import devol.individ;
    import devol.world;
    import devol.operator;
    import devol.std.typePod;
    import devol.std.typevoid;
    import devol.std.argvoid;
}

import evol.individ;

class GenericPushOperator : Operator
{
    TypePod!double doubletype;
    TypeVoid voidtype;

    enum description = "Сохраняет" действительное числовое значение .";

    this()
    {
        doubletype = cast(TypePod!double)(TypeMng.getSingleton().getType("Typedouble"));
        assert(doubletype, "We need double type!");

        voidtype = cast(TypeVoid)(TypeMng.getSingleton().getType("TypeVoid"));
        assert(voidtype);

        mRetType = voidtype;
        super("gpush", description, ArgsStyle.UNAR_STYLE);

        ArgInfo a1;
        a1.type = doubletype;
        a1.min = "-1000";
        a1.max = "+1000";

        args ~= a1;
    }

    override Argument apply(IndAbstract ind, Line line, WorldAbstract world)
    {
        auto gind = cast(GraphIndivid)ind;
        assert(gind);

        auto a1 = cast(ArgPod!double)(line[0]);
        assert(a1);

        gind.genericStack.stackPush(a1);

        return new ArgVoid;
    }
}

```

26 Пакет EVOL.OPERATORS.GROT

```

module evol.operators.grot;

import std.stdio;

import devol.typeMng;

public
{
    import devol.individ;
    import devol.world;
    import devol.operator;
    import devol.std.typevoid;
    import devol.std.argvoid;
}

import evol.individ;

class GenericRotOperator : Operator
{
    TypeVoid voidtype;

    enum description = "Перемещает" третий элемент головы стека на вершину . Стек общего назначения .";

```

```

    this()
{
    voidtype = cast(TypeVoid)(TypeMng.getSingleton().getType("TypeVoid"));
    assert(voidtype, "We need double type!");

    mRetType = voidtype;
    super("grot", description, ArgsStyle.NULAR_STYLE);
}

override Argument apply(IndAbstract ind, Line line, WorldAbstract world)
{
    auto gind = cast(GraphIndivid)ind;
    assert(gind);

    gind.genericStack.stackRot;
    return new ArgVoid;
}
}

```

27 Пакет EVOL.OPERATORS.GSWAP

```

module evol.operators.gswap;

import std.stdio;

import devol.typeMng;

public
{
    import devol.individ;
    import devol.world;
    import devol.operator;
    import devol.std.typevoid;
    import devol.std.argvoid;
}

import evol.individ;

class GenericSwapOperator : Operator
{
    TypeVoid voidtype;

    enum description = "Меняет" mestamiдвапоследнихзначениявстекеобщегоназначения .";

    this()
    {
        voidtype = cast(TypeVoid)(TypeMng.getSingleton().getType("TypeVoid"));
        assert(voidtype, "We need double type!");

        mRetType = voidtype;
        super("gswap", description, ArgsStyle.NULAR_STYLE);
    }

    override Argument apply(IndAbstract ind, Line line, WorldAbstract world)
    {
        auto gind = cast(GraphIndivid)ind;
        assert(gind);

        gind.genericStack.stackSwap;
        return new ArgVoid;
    }
}

```

28 Пакет EVOL.OPERATORS.IDCAST

```

module evol.operators.idcast;

import std.stdio;

import devol.typeMng;

public
{
    import devol.individ;
    import devol.world;
    import devol.operator;
    import devol.std.typepod;
}
```

```

}

class IntDoubleCastOperator : Operator
{
    TypePod!double doubletype;
    TypePod!int inttype;

    enum description = Преобразует" целочисленное действительное число .";

    this()
    {
        inttype = cast(TypePod!int)(TypeMng.getSingleton().getType("Typeint"));
        assert(inttype, "We need int type!");

        doubletype = cast(TypePod!double)(TypeMng.getSingleton().getType("Typedouble"));
        assert(doubletype, "We need double type!");

        mRetType = doubletype;
        super("cast", description, ArgsStyle.UNAR_STYLE);

        ArgInfo a1;
        a1.type = inttype;
        a1.min = "-100";
        a1.max = "+100";

        args ~= a1;
    }

    override Argument apply(IndAbstract ind, Line line, WorldAbstract world)
    {
        auto ret = doubletype.getNewArg();

        auto a1 = cast(ArgPod!int)(line[0]);

        assert( a1 !is null, "Critical error: Operator plus, argument 1 isn't a right value!");

        ret = cast(double)a1.val;
        return ret;
    }
}

```

29 Пакет EVOL.OPERATORS.IDUP

```

module evol.operators.idup;

import std.stdio;

import devol.typemng;

public
{
    import devol.individ;
    import devol.world;
    import devol.operator;
    import devol.std.typevoid;
    import devol.std.argvoid;
}

import evol.individ;

alias InputDupFirstOperator = InputDupOperator!((ind) => ind.firstGraphStack, "idup1"
    , Копирует" вершину стека первого графа .);
alias InputDupSecondOperator = InputDupOperator!((ind) => ind.secondGraphStack, "idup2"
    , Копирует" вершину стека второго графа .);

class InputDupOperator(alias stack, string opname, string description) : Operator
{
    TypeVoid voidtype;

    this()
    {
        voidtype = cast(TypeVoid)(TypeMng.getSingleton().getType("TypeVoid"));
        assert(voidtype);

        mRetType = voidtype;
        super(opname, description, ArgsStyle.NULAR_STYLE);
    }

    override Argument apply(IndAbstract ind, Line line, WorldAbstract world)
    {
        auto gind = cast(GraphIndivid)ind;
        assert(gind);
    }
}

```

```

        stack(gind).stackDup();

        return new ArgVoid;
    }
}

```

30 Пакет EVOL.OPERATORS.IOVER

```

module evol.operators.iover;

import std.stdio;

import devol.typeMng;

public
{
    import devol.individ;
    import devol.world;
    import devol.operator;
    import devol.std.typevoid;
    import devol.std.argvoid;
}

import evol.individ;

alias InputOverFirstOperator = InputOverOperator!((ind) => ind.firstGraphStack, "iover1"
    , Копирует" второйэлементстекапервогографанавершину      .");
alias InputOverSecondOperator = InputOverOperator!((ind) => ind.secondGraphStack, "iover2"
    , Копирует" второйэлементстекавторогографанавершину      .");

class InputOverOperator(alias stack, string opname, string description) : Operator
{
    TypeVoid voidtype;

    this()
    {
        voidtype = cast(TypeVoid)(TypeMng.getSingleton().getType("TypeVoid"));
        assert(voidtype);

        mRetType = voidtype;
        super(opname, description, ArgsStyle.NULAR_STYLE);
    }

    override Argument apply(IndAbstract ind, Line line, WorldAbstract world)
    {
        auto gind = cast(GraphIndivid)ind;
        assert(gind);

        stack(gind).stackOver();

        return new ArgVoid;
    }
}

```

31 Пакет EVOL.OPERATORS.IPOP

```

module evol.operators.ipop;

import std.stdio;

import devol.typeMng;

public
{
    import devol.individ;
    import devol.world;
    import devol.operator;
    import evol.types.typeEdge;
    import evol.types.edge;
}

import evol.individ;

alias InputPopFirstOperator = InputPopOperator!((ind) => ind.firstGraphStack, "ipop1", Снимает
    " ивозвращаетголовустекапервогографа      .");
alias InputPopSecondOperator = InputPopOperator!((ind) => ind.secondGraphStack, "ipop2", Снимает
    " ивозвращаетголовустекавторогографа      .");

```

```

class InputPopOperator(alias stack, string opname, string description) : Operator
{
    TypeEdge edgetype;

    this()
    {
        edgetype = cast(TypeEdge)(TypeMng.getSingleton().getType("TypeEdge"));
        assert(edgetype, "We need edge type!");

        mRetType = edgetype;
        super(opname, description, ArgsStyle.NULAR_STYLE);
    }

    override Argument apply(IndAbstract ind, Line line, WorldAbstract world)
    {
        auto gind = cast(GraphIndivid)ind;
        assert(gind);

        return stack(gind).stackPop();
    }
}

```

32 Пакет EVOL.OPERATORS.IPUSH

```

module evol.operators.ipush;

import std.stdio;

import devol.typemng;

public
{
    import devol.individ;
    import devol.world;
    import devol.operator;
    import devol.std.typevoid;
    import devol.std.argvoid;

    import evol.types.typeedge;
    import evol.types.argedge;
}

import evol.individ;

alias InputPushFirstOperator = InputPushOperator!((ind) => ind.firstGraphStack, "ipush1"
    , Сохраняет" граньграфавходнойстекдляпервогографа");
alias InputPushSecondOperator = InputPushOperator!((ind) => ind.secondGraphStack, "ipush2"
    , Сохраняет" граньграфавходнойстекдлявторогографа");

class InputPushOperator(alias stack, string opname, string description) : Operator
{
    TypeEdge edgetype;
    TypeVoid voidtype;

    this()
    {
        edgetype = cast(TypeEdge)(TypeMng.getSingleton().getType("TypeEdge"));
        assert(edgetype, "We need edge type!");

        voidtype = cast(TypeVoid)(TypeMng.getSingleton().getType("TypeVoid"));
        assert(voidtype);

        mRetType = voidtype;
        super(opname, description, ArgsStyle.UNAR_STYLE);

        ArgInfo a1;
        a1.type = edgetype;

        args ~= a1;
    }

    override Argument apply(IndAbstract ind, Line line, WorldAbstract world)
    {
        auto gind = cast(GraphIndivid)ind;
        assert(gind);

        auto a1 = cast(ArgEdge)(line[0]);
        assert(a1);

        stack(gind).stackPush(a1);
    }
}

```

```

        return new ArgVoid;
    }
}

```

33 Пакет EVOL.OPERATORS.IROT

```

module evol.operators.irot;

import std.stdio;

import devol.typeMng;

public
{
    import devol.individ;
    import devol.world;
    import devol.operator;
    import devol.std.typevoid;
    import devol.std.argvoid;
}

import evol.individ;

alias InputRotFirstOperator = InputRotOperator!((ind) => ind.firstGraphStack, "irot1"
    , Перемещает" третийэлементстекапервогографанавершину .");
alias InputRotSecondOperator = InputRotOperator!((ind) => ind.secondGraphStack, "irot2"
    , Перемещает" третийэлементстекавторогографанавершину .");

class InputRotOperator(alias stack, string opname, string description) : Operator
{
    TypeVoid voidtype;

    this()
    {
        voidtype = cast(TypeVoid)(TypeMng.getSingleton().getType("TypeVoid"));
        assert(voidtype);

        mRetType = voidtype;
        super(opname, description, ArgsStyle.NULAR_STYLE);
    }

    override Argument apply(IndAbstract ind, Line line, WorldAbstract world)
    {
        auto gind = cast(GraphIndivid)ind;
        assert(gind);

        stack(gind).stackRot();

        return new ArgVoid;
    }
}

```

34 Пакет EVOL.OPERATORS.ISWAP

```

module evol.operators.iswap;

import std.stdio;

import devol.typeMng;

public
{
    import devol.individ;
    import devol.world;
    import devol.operator;
    import devol.std.typevoid;
    import devol.std.argvoid;
}

import evol.individ;

alias InputSwapFirstOperator = InputSwapOperator!((ind) => ind.firstGraphStack, "iswap1"
    , Меняет" первыхдваЗлементанастекедляпервогографаместами .");
alias InputSwapSecondOperator = InputSwapOperator!((ind) => ind.secondGraphStack, "iswap2"
    , Меняет" первыхдваЗлементанастекедлявторогографаместами .");

class InputSwapOperator(alias stack, string opname, string description) : Operator
{

```

```

TypeVoid voidtype;

this()
{
    voidtype = cast(TypeVoid)(TypeMng.getSingleton().getType("TypeVoid"));
    assert(voidtype);

    mRetType = voidtype;
    super(opname, description, ArgsStyle.NULAR_STYLE);
}

override Argument apply(IndAbstract ind, Line line, WorldAbstract world)
{
    auto gind = cast(GraphIndivid)ind;
    assert(gind);

    stack(gind).stackSwap();

    return new ArgVoid;
}
}

```

35 Пакет EVOL.OPERATORS.MULT

```

module evol.operators.mult;

import std.stdio;

import devol.typemng;

public
{
    import devol.individ;
    import devol.world;
    import devol.operator;
    import devol.std.typepod;
}

class MultOperator : Operator
{
    TypePod!double doubletype;

    enum description = "Арифметическая" операция умножения действительных чисел .";

    this()
    {
        doubletype = cast(TypePod!double)(TypeMng.getSingleton().getType("Typedouble"));
        assert(doubletype, "We need double type!");

        mRetType = doubletype;
        super("*", description, ArgsStyle.BINAR_STYLE);

        ArgInfo a1;
        a1.type = doubletype;
        a1.min = "-1000";
        a1.max = "+1000";

        args ~= a1;
        args ~= a1;
    }

    override Argument apply(IndAbstract ind, Line line, WorldAbstract world)
    {
        auto ret = doubletype.getNewArg();

        auto a1 = cast(ArgPod!double)(line[0]);
        auto a2 = cast(ArgPod!double)(line[1]);

        assert( a1 != null, "Critical error: Operator plus, argument 1 isn't a right value!");
        assert( a2 != null, "Critical error: Operator plus, argument 2 isn't a right value!");

        ret = a1.val * a2.val;
        return ret;
    }
}

```

36 Пакет EVOL.OPERATORS.NOT

```

module evol.operators.not;

import devol.world;
import devol.std.line;
import devol.individ;
import devol.operator;
import devol.type;
import devol.typeMng;
import devol.std.argPod;
import devol.std.typePod;
import devol.argument;

class NotOperator : Operator
{
    TypePod!bool booltpe;

    this()
    {
        booltpe = cast(TypePod!bool)TypeMng.getSingleton().getType("Typebool");
        mRetType = booltpe;

        super("!", "Логическое" НЕТ '' для значений ложь истина /., ArgsStyle.UNAR_STYLE);

        ArgInfo a1;
        a1.type = booltpe;
        args ~= a1;
    }

    override Argument apply(IndAbstract individ, Line line, WorldAbstract world)
    {
        auto ret = booltpe.getNewArg();

        auto a1 = cast(ArgPod!bool)line[0];

        assert(a1 != null);

        ret = !a1.val;
        return ret;
    }
}

```

37 Пакет EVOL.OPERATORS.OPIF

```

module evol.operators.opif;

import devol.typeMng;

import devol.individ;
import devol.world;
import devol.operator;
import devol.std.typePod;

debug import std.stdio;

class IfOperator : Operator
{
    TypePod!bool booltpe;
    TypeVoid voidtype;

    enum description = Условный" оператор, который берет три аргумента . Первый "имеет
    " логический тип , который относится к условиям действия . Если этот аргумент "вычисляется
    " значение ИСТИНА '', то возвращается второй аргумент , иначе "возвращается
    " третий аргумент . Второй и третий аргументы относятся к действиям , которые
    " имеют тип void";

    this()
    {
        booltpe = cast(TypePod!bool)(TypeMng.getSingleton().getType("Typebool"));
        assert(booltpe, "We need bool type!");

        voidtype = cast(TypeVoid)(TypeMng.getSingleton().getType("TypeVoid"));

        mRetType = voidtype;
        super("if", description, ArgsStyle.CONTROL_STYLE);

        ArgInfo a1;
        a1.type = booltpe;
        args ~= a1;

        a1.type = voidtype;
        args ~= a1;
        args ~= a1;
    }
}

```

```

    }

override Argument apply(IndAbstract ind, Line line, WorldAbstract world)
{
    auto cond = cast(ArgPod!bool)(line[0]);

    Line vthen = cast(Line)(line[1]);
    Line velse = cast(Line)(line[2]);

    ArgScope sthen = cast(ArgScope)(line[1]);
    ArgScope selse = cast(ArgScope)(line[2]);

    if (cond.val)
    {
        if (vthen !is null)
        {
            vthen.compile(ind, world);
        } else if (sthen !is null)
        {
            foreach(Line aline; sthen)
            {
                auto line = cast(Line)aline;
                line.compile(ind, world);
            }
        } else
        {
            debug writeln("Warning: invalid ThenArg: ", line.toString());
        } //else throw new Exception("If is confused! ThenArg is no line, no scope." ~ line.toString());
    } else
    {
        if (velse !is null)
        {
            velse.compile(ind, world);
        } else if (selse !is null)
        {
            foreach(Line aline; selse)
            {
                auto line = cast(Line)aline;
                line.compile(ind, world);
            }
        } else
        {
            debug writeln("Warning: invalid ElseArg: ", line.toString());
        } //else throw new Exception("If is confused! ElseArg is no line, no scope." ~ line.toString());
    }
    return voidtype.getNewArg();
}
}

```

38 Пакет EVOL.OPERATORS.OPWHILE

```

module evol.operators.opwhile;

import devol.typeMng;

import devol.individ;
import devol.world;
import devol.operator;
import devol.std.typePod;

debug import std.stdio;

class WhileOperator : Operator
{
    TypePod!bool booltpe;
    TypeVoid voidtype;

    enum MAX_ITERATIONS = 100;

    enum description = "Оператор ., управляющий потоком исполнения . Его второй аргумент " выполняется
    " dottedpor , пока первый аргумент вычисляется ИСТИНА . "Bo
    " избежание бесконечной программы накладывается ограничение на " максимальное
    " число итераций .";

    this()
    {
        booltpe = cast(TypePod!bool)(TypeMng.getSingleton().getType("Typebool"));
        assert(booltpe, "We need bool type!");

        voidtype = cast(TypeVoid)(TypeMng.getSingleton().getType("TypeVoid"));

        mRetType = voidtype;
    }
}

```

```

super("while", description, ArgsStyle.CONTROL_STYLE);

ArgInfo a1;
a1.type = booletype;
a1.eval = false;
args ~= a1;

a1.type = voidtype;
a1.eval = false;
args ~= a1;
}

override Argument apply(IndAbstract ind, Line line, WorldAbstract world)
{
    auto condLine = cast(Line)(line[0]);
    auto condConst = cast(ArgPod!bool)(line[0]);
    size_t iterations;

    Line vaction = cast(Line)(line[1]);
    ArgScope saction = cast(ArgScope)(line[1]);

    void iterateOnce()
    {
        if (vaction != null)
        {
            vaction.compile(ind, world);
        } else if (saction != null)
        {
            foreach(Line aline; saction)
            {
                auto line = cast(Line)aline;
                line.compile(ind, world);
            }
        } else
        {
            debug writeln("Warning: invalid ThenArg: ", line.toString());
        }
    }

    if(condLine == null)
    {
        assert(condConst);
        if(condConst)
        {
            foreach(i; 0..MAX_ITERATIONS)
            {
                iterateOnce();
            }
        }
    } else
    {
        foreach(i; 0..MAX_ITERATIONS)
        {
            if(!condLine.compile(ind, world)) break;
            iterateOnce();
        }
    }
}

return voidtype.getNewArg();
}
}

```

39 Пакет EVOL.OPERATORS.OR

```

module evol.operators.or;

import devol.world;
import devol.std.line;
import devol.individ;
import devol.operator;
import devol.type;
import devol.typemng;
import devol.std.argpod;
import devol.std.typepod;
import devol.argument;

class OrOperator : Operator
{
    TypePod!bool booletype;

    this()
    {

```

```

booltype = cast(TypePod!bool)TypeMng.getSingleton().getType("Typebool");
mRetType = booletype;

super("||", "Логическое" ИЛИ '' длязначенийложьистина / ., ArgsStyle.BINAR_STYLE);

ArgInfo a1;
a1.type = booletype;
args ~= a1;
args ~= a1;
}

override Argument apply(IndAbstract individ, Line line, WorldAbstract world)
{
    auto ret = booletype.getNewArg();

    auto a1 = cast(ArgPod!bool)line[0];
    auto a2 = cast(ArgPod!bool)line[1];

    assert(a1 != null);
    assert(a2 != null);

    ret = a1.val || a2.val;
    return ret;
}
}

```

40 Пакет EVOL.OPERATORS.PLUS

```

module evol.operators.plus;

import std.stdio;

import devol.typemng;

public
{
    import devol.individ;
    import devol.world;
    import devol.operator;
    import devol.std.typepod;
}

class PlusOperator : Operator
{
    TypePod!double doubletype;

    enum description = Арифметическая" операция сложения действительных чисел .";

    this()
    {
        doubletype = cast(TypePod!double)(TypeMng.getSingleton().getType("Typedouble"));
        assert(doubletype, "We need double type!");

        mRetType = doubletype;
        super("+", description, ArgsStyle.BINAR_STYLE);

        ArgInfo a1;
        a1.type = doubletype;
        a1.min = "-1000";
        a1.max = "+1000";

        args ~= a1;
        args ~= a1;
    }

    override Argument apply(IndAbstract ind, Line line, WorldAbstract world)
    {
        auto ret = doubletype.getNewArg();

        auto a1 = cast(ArgPod!double)(line[0]);
        auto a2 = cast(ArgPod!double)(line[1]);

        assert( a1 != null, "Critical error: Operator plus, argument 1 isn't a right value!");
        assert( a2 != null, "Critical error: Operator plus, argument 2 isn't a right value!");

        ret = a1.val + a2.val;
        return ret;
    }
}

```

41 Пакет EVOL.OPERATORS.RELATION

```
module evol.operators.relation;

import devol.typeMng;

public
{
    import devol.individ;
    import devol.world;
    import devol.operator;
    import devol.std.typePod;
}

alias IntEqualOperator = RelationOperator!("== (int)", "==", TypePod!int, ArgPod!int, "Typeint", Сравнение"
    ↪ наравенство целочисленных аргументов .");
alias DoubleEqualOperator = RelationOperator!("== (double)", "==", TypePod!double, ArgPod!double, "Typedouble",
    ↪ Сравнение" наравенство действительных аргументов .");

alias IntGreaterOperator = RelationOperator!("> (int)", ">", TypePod!int, ArgPod!int, "Typeint", Сравнение"
    ↪ целочисленных аргументов . Возвращает ИСТИНА , если первый больше второго .");
alias IntLesserOperator = RelationOperator!("< (int)", "<", TypePod!int, ArgPod!int, "Typeint", Сравнение"
    ↪ целочисленных аргументов . Возвращает ИСТИНА , если первый меньше второго .");
alias IntGreaterEqualOperator = RelationOperator!(">= (int)", ">=", TypePod!int, ArgPod!int, "Typeint", Сравнение"
    ↪ целочисленных аргументов . Возвращает ИСТИНА , если первый больше второго или равен второму .");
alias IntLesserEqualOperator = RelationOperator!("<= (int)", "<=", TypePod!int, ArgPod!int, "Typeint", Сравнение"
    ↪ целочисленных аргументов . Возвращает ИСТИНА , если первый меньше второго или равен второму .");

alias DoubleGreaterOperator = RelationOperator!("> (double)", ">", TypePod!double, ArgPod!double, "Typedouble",
    ↪ Сравнение" действительных аргументов . Возвращает ИСТИНА , если первый больше второго .");
alias DoubleLesserOperator = RelationOperator!("< (double)", "<", TypePod!double, ArgPod!double, "Typedouble",
    ↪ Сравнение" действительных аргументов . Возвращает ИСТИНА , если первый меньше второго .");

class RelationOperator( string opname, string relation, DslType, DslArg, string dslTypeName, string description) :
    ↪ Operator
{
    DslType inputType;
    TypePod!bool boolType;

    static assert(opname != "");

    this()
    {
        inputType = cast(DslType)(TypeMng.getSingleton().getType(dslTypeName));
        assert(inputType, "We need ~dslTypeName~ type!");

        boolType = cast(TypePod!bool)(TypeMng.getSingleton().getType("Typebool"));
        assert(boolType, "We need bool type!");

        mRetType = boolType;
        super(opname, description, ArgsStyle.BINAR_STYLE);

        ArgInfo a1;
        a1.type = inputType;
        a1.min = "-1000";
        a1.max = "+1000";

        args ~= a1;
        args ~= a1;
    }

    override Argument apply(IndAbstract ind, Line line, WorldAbstract world)
    {
        auto ret = boolType.getNewArg();

        auto a1 = cast(DslArg)(line[0]);
        auto a2 = cast(DslArg)(line[1]);

        assert( a1 !is null, "Critical error: Operator ~name~, argument 1 isn't a right value!");
        assert( a2 !is null, "Critical error: Operator ~name~, argument 2 isn't a right value!");

        ret = mixin(q{a1.val} ~ relation ~ q{a2.val});
        return ret;
    }
}
```

42 Пакет EVOL.OPERATORS.ROUND

```

module evol.operators.round;

import std.stdio;
import std.math;

import devol.typemng;

public
{
    import devol.individ;
    import devol.world;
    import devol.operator;
    import devol.std.typepod;
}

class RoundOperator : Operator
{
    TypePod!double doubletype;
    TypePod!int inttype;

    enum description = "Преобразует действительное в целочисленное число с помощью математического округления .";

    this()
    {
        inttype = cast(TypePod!int)(TypeMng.getSingleton().getType("Typeint"));
        assert(inttype, "We need int type!");

        doubletype = cast(TypePod!double)(TypeMng.getSingleton().getType("Typedouble"));
        assert(doubletype, "We need double type!");

        mRetType = inttype;
        super("round", description, ArgsStyle.UNAR_STYLE);

        ArgInfo a1;
        a1.type = doubletype;
        a1.min = "-100";
        a1.max = "+100";

        args ~= a1;
    }

    override Argument apply(IndAbstract ind, Line line, WorldAbstract world)
    {
        auto ret = inttype.getNewArg();

        auto a1 = cast(ArgPod!double)(line[0]);

        assert(a1 != null, "Critical error: Operator plus, argument 1 isn't a right value!");

        ret = cast(int)round(a1.val);
        return ret;
    }
}

```

43 Пакет EVOL.OPERATORS.SOURCE

```

module evol.operators.source;

import std.stdio;

import devol.typemng;

public
{
    import devol.individ;
    import devol.world;
    import devol.operator;
    import devol.std.typevoid;
    import devol.std.argvoid;
    import devol.std.typepod;

    import evol.types.typeedge;
    import evol.types.argedge;
}

import evol.individ;

class GetSourceOperator : Operator
{
    TypeEdge edgetype;
    TypePod!int inttype;
}

```

```
enum description = Возвращает" индексвершины , изкоторойвыходитребографа .";
this()
{
    edgetype = cast(TypeEdge)(TypeMng.getSingleton().getType("TypeEdge"));
    assert(edgetype, "We need edge type!");

    inttype = cast(TypePod!int)(TypeMng.getSingleton().getType("Typeint"));
    assert(inttype);

    mRetType = inttype;
    super("getSource", description, ArgsStyle.UNAR_STYLE);

    ArgInfo a1;
    a1.type = edgetype;

    args ~= a1;
}

override Argument apply(IndAbstract ind, Line line, WorldAbstract world)
{
    auto gind = cast(GraphIndivid)ind;
    assert(gind);

    auto a1 = cast(ArgEdge)(line[0]);
    assert(a1);

    return new ArgPod!int(cast(int)a1.edge.source);
}
}
```