Разработка компонентов графоориентированного программного каркаса для реализации сложных вычислительных методов

Тришин Илья Вадимович, группа РК6-81Б

МГТУ им. Н.Э. Баумана

Россия, Москва, – 20 июня 2022

Введение

↳ Актуальные прикладные задачи и проблемы программной реализации их решений

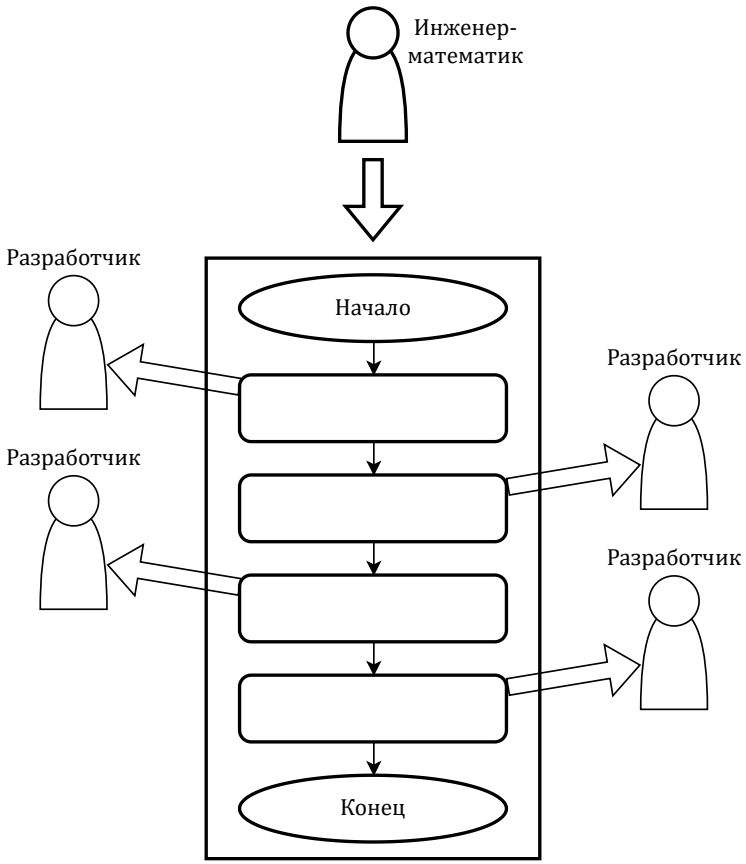
- Задачи математического моделирования.
- Задачи анализа данных.
- Задачи проектных инженерных расчётов.
- и т.д.

- **Практически значимая программная реализация методов решений прикладных задач, как правило, является нетривиальной**

Введение

↳ Программная реализация прикладных решений в команде

- Наличие в команде нескольких человек и распределение обязанностей по разработке отдельных этапов между ними положительно сказывается на качестве реализуемого ПО



- Введение новых разработчиков в проект влечёт за собой необходимость описывать для них логику реализуемого решения.
- Целесообразно включить разработку описания логики решения в процесс его программной реализации.

Введение

↳ Современные средства, направленные на упрощение реализации решений прикладных задач

Среди современных средств упрощения разработки можно выделить следующие:

- научные системы управления потоком задач.
- платформы малокодовой разработки (LCPD).

Основной принцип

Использование формальных описаний алгоритма решения при его непосредственной программной реализации.

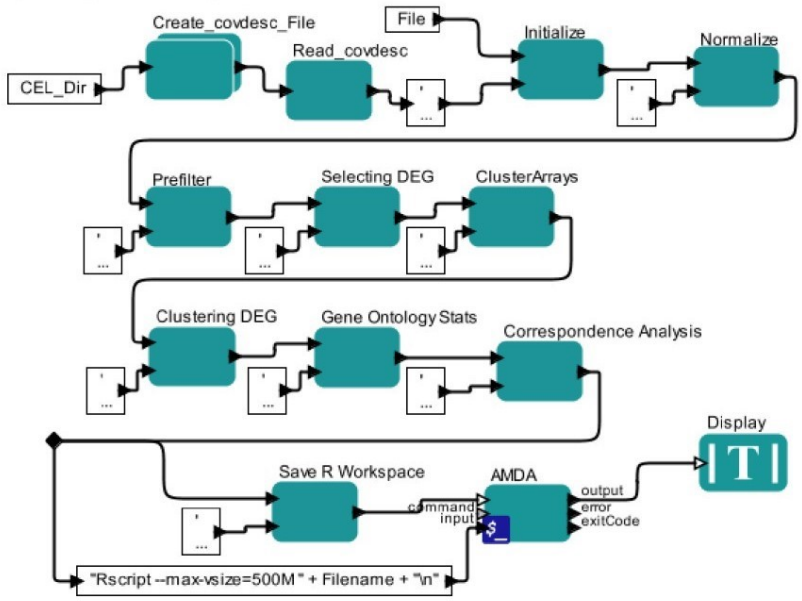


Рис. 1: Пример описания алгоритма в научной системе управления потоком задач Kepler

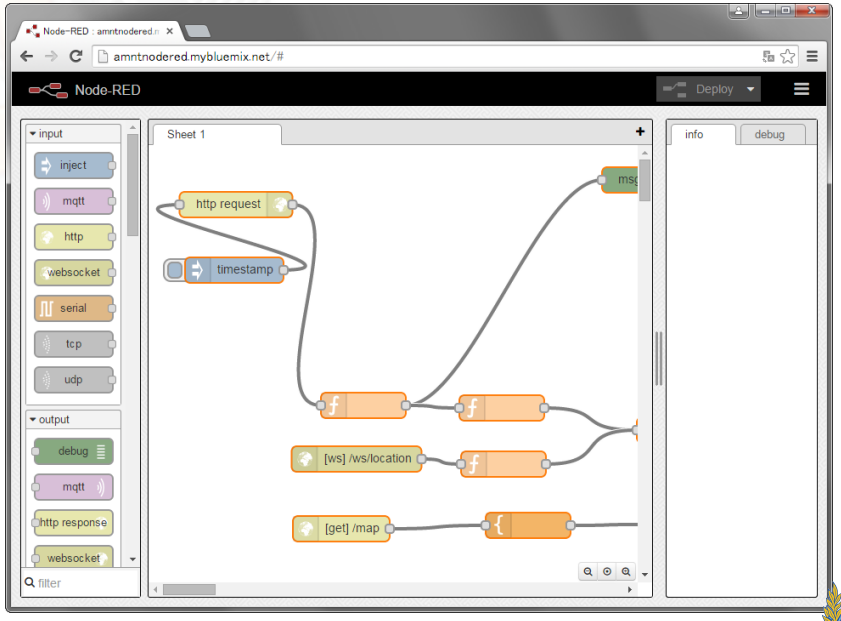


Рис. 2: Пример описания алгоритма в LCPD Node-RED

- Можно выделить следующие подходы:
  - Диаграммы потоков данных (DFD).
  - Диаграммы потока управления.
  - Сетевые графики
  - и т.д.

- **В данной работе в первую очередь рассматривается подход, получивший название Graph-Based Software Engineering (GBSE)**.

Введение
↳ Диаграммы потоков данных

- Описание алгоритма представляет собой ориентированный граф.
- С его вершинами связываются процессы обработки данных.
- Его рёбра определяют пути данных между процессами.

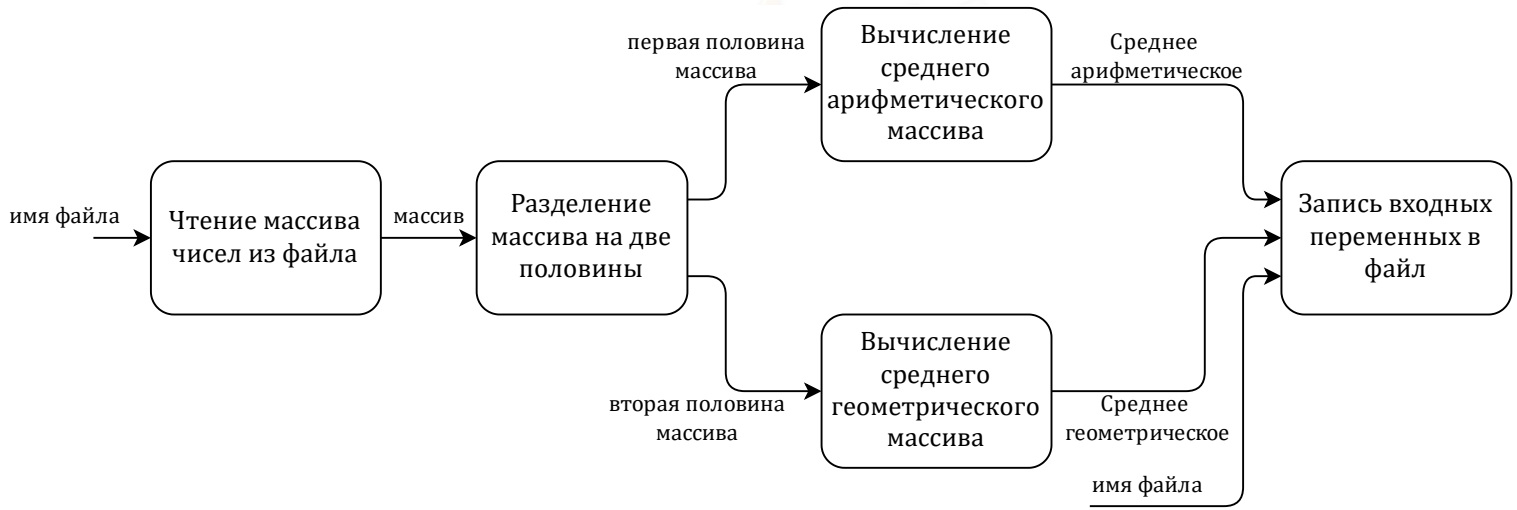


Рис. 3: Пример диаграммы потоков данных, описывающей вычисление среднего арифметического и среднего геометрического двух половин массива целых чисел с последующей записью результатов в файл

Введение

↳ pSeven – реализация диаграмм потоков данных

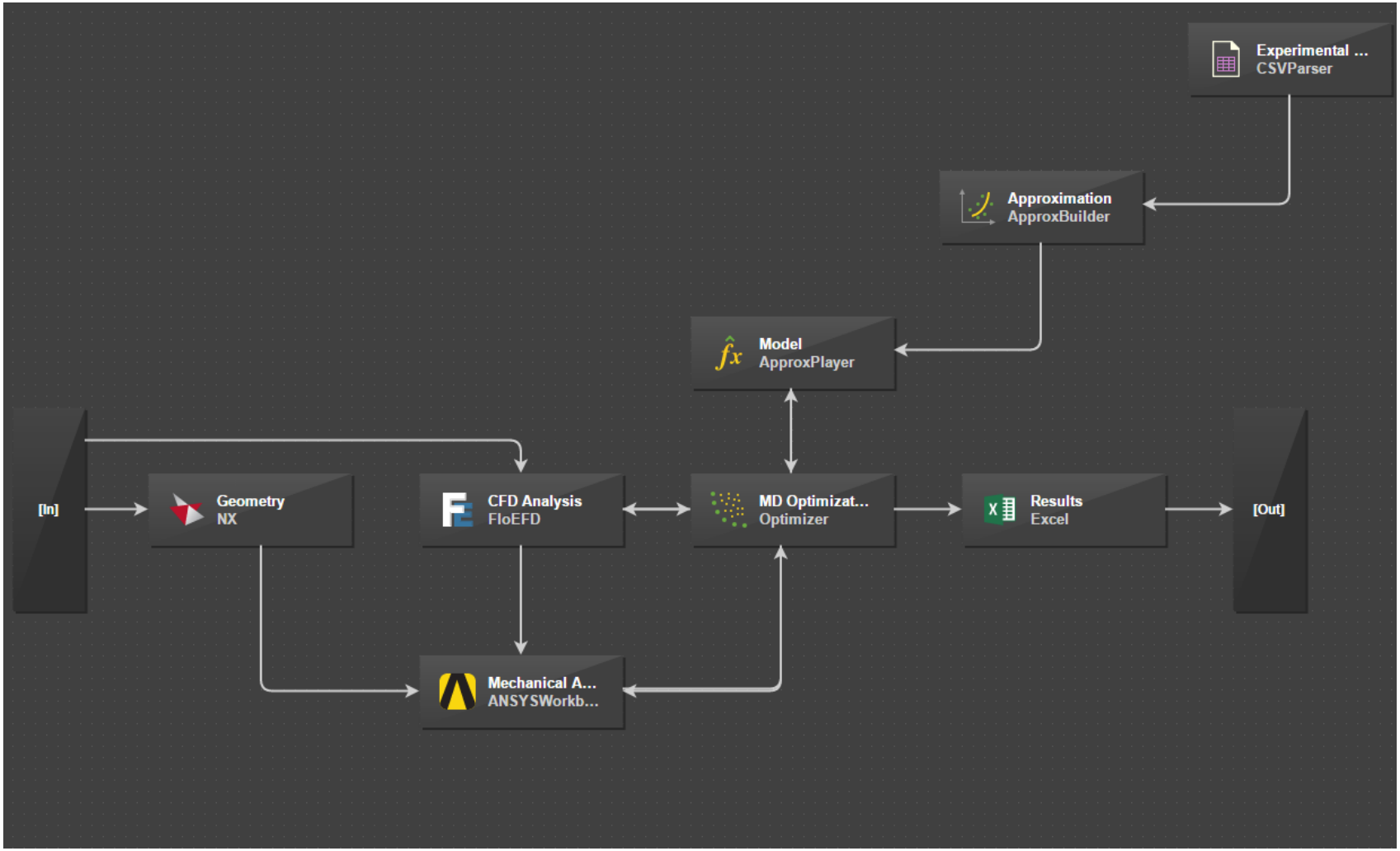


Рис. 4: Графический пользовательский интерфейс pSeven

Основные достоинства comsdk по сравнению с pSeven:

- Нет необходимости указывать входные и выходные данные при описании алгоритма.
- По умолчанию поддерживаются алгоритмы, подразумевающие взаимодействие с пользователем

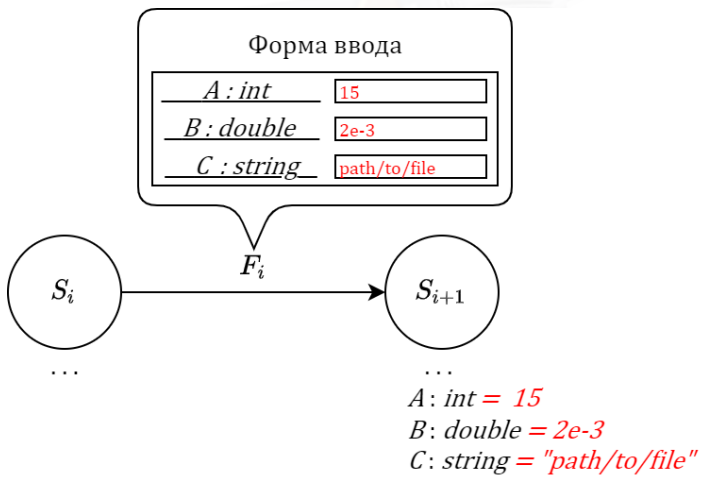


Рис. 5: Пример получения данных от пользователя при помощи генерируемой формы ввода

- Результат применения – компилируемая программа с возможностью запуска на различных платформах.

Основные недостатки comsdk по сравнению с pSeven:

- Отсутствие поддержки матричных типов данных.
- Отсутствие средств визуализации результатов расчётов.
- Отсутствие возможности использования при расчётах распределённых вычислительных систем;

Цель

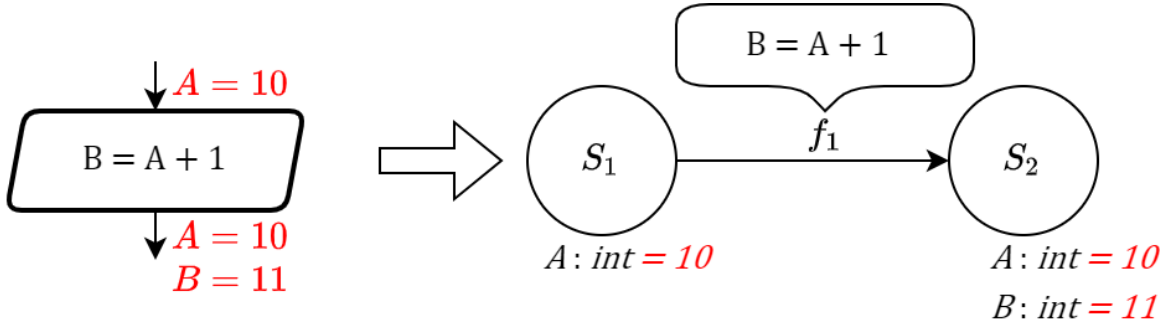
Разработать обновлённые программные средства для создания и интерпретации графовых описаний вычислительных методов в программном каркасе comsdk.

Задачи

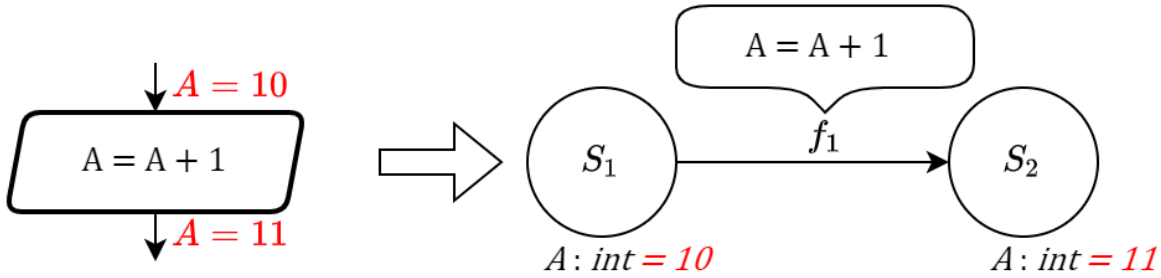
1. Сформировать требования к алгоритму, выполняющему этапы алгоритма по его описанию, составленному по методологии GBSE.
2. Спроектировать структуры данных для описания и представления описаний алгоритмов и их элементов в программном каркасе comsdk.
3. Разработать алгоритм обхода графовых моделей с использованием спроектированных структур данных.
4. Представить интерфейсы или реализации разработанных алгоритмов и структур данных на языке C++.

$$\underbrace{S = \left\langle \begin{array}{l} A : int \\ B : double \\ C : string \end{array} \right\rangle}_{\text{Состояние данных}} \qquad \underbrace{D \multimap S = \left\langle \begin{array}{l} A : int = 10 \\ B : double = 2e\text{-}3 \\ C : string = \text{"path/to/file"} \end{array} \right\rangle}_{\text{Данные } D \text{ в состоянии } S}$$

- Состояние данных определяет, какие переменные какого типа должны быть определены на данном этапе алгоритма.
- Данные алгоритма модифицируются по ходу его выполнения.



Вариант 1. Функция-обработчик модифицирует состояние данных



Вариант 2. Функция-обработчик модифицирует только сами данные

- Функции-обработчики отвечают за обработку данных и их перевод из одного состояния в другое.

Постановка задачи

↳ Основные понятия GBSE. Функции-предикаты и функции перехода

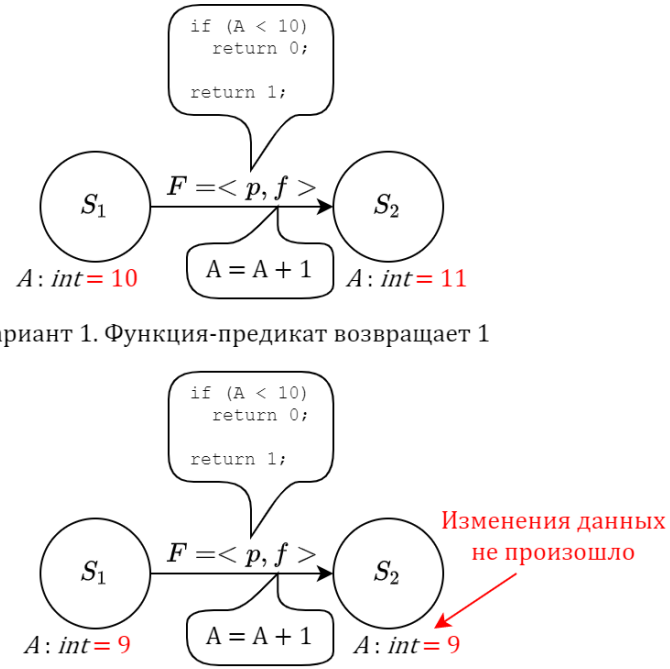


Рис. 6: Принцип работы функции-предиката

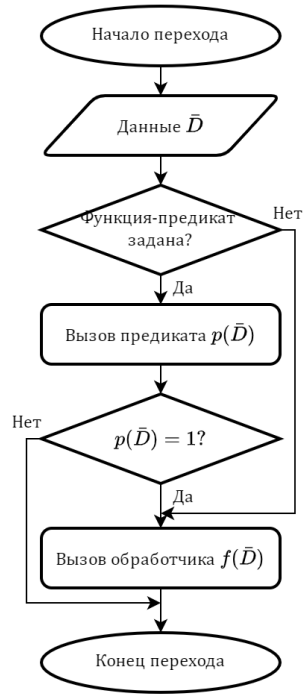


Рис. 7: Блок-схема логики функции перехода

- Функции-предикаты отвечают за предварительную проверку данных перед их обработкой.
- Функция перехода – составная функция $F =< p, f >$, содержащая в себе функцию-предикат p и функцию-обработчик f.

- Функции-селекторы отвечают за проверку условий при ветвлении алгоритма.

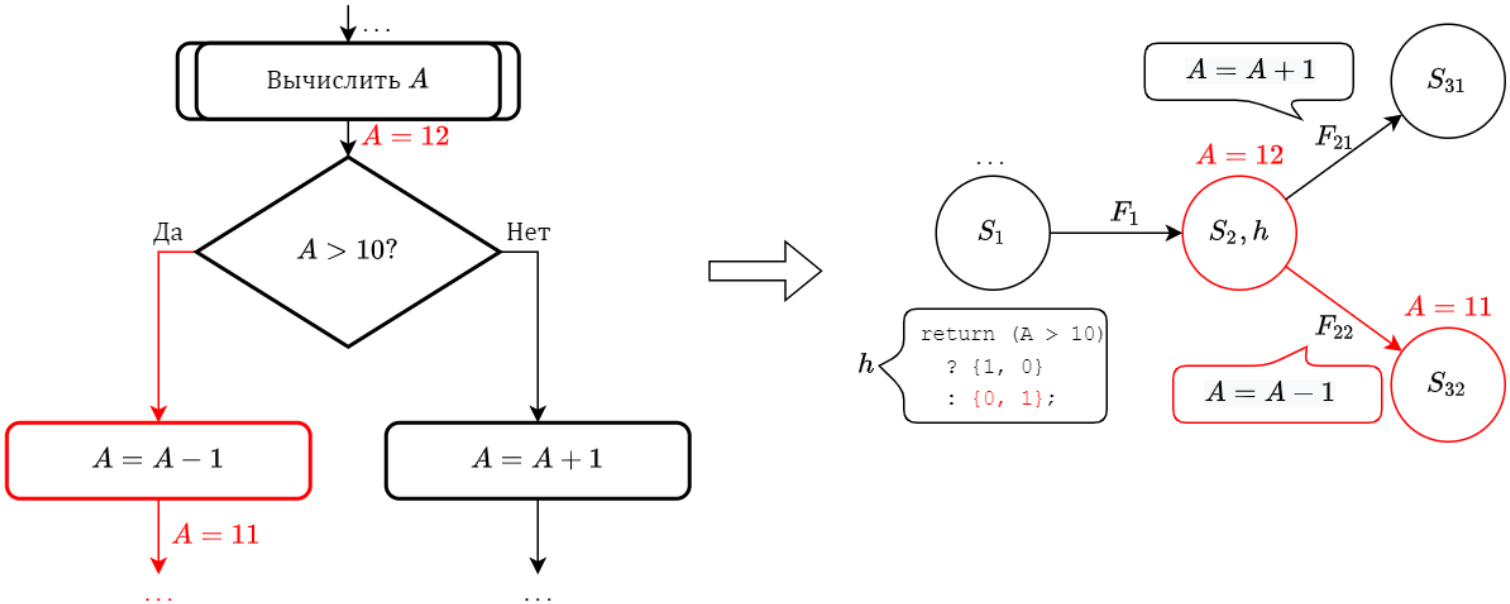


Рис. 8: Принцип работы функций-селекторов. h – функция селектор. Красным показана ветвь алгоритма, которая будет выполнена.

Постановка задачи

↳ Основные понятия GBSE. Графовая модель

- Графовая модель сложного вычислительного метода описывает его логику в виде ориентированного графа.
- Узлам ставятся в соответствие состояния данных.
- Рёбрам ставятся в соответствие функции перехода.

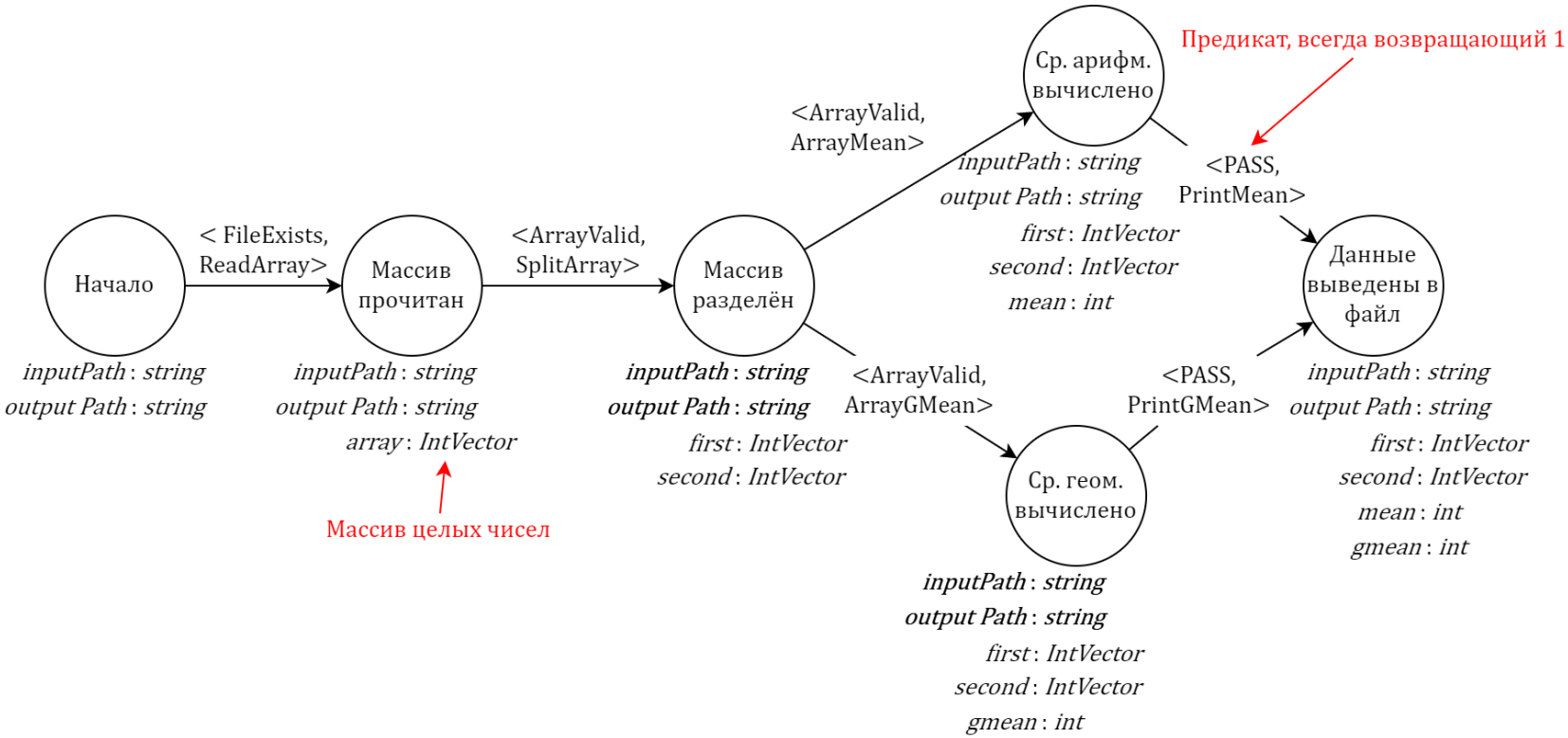
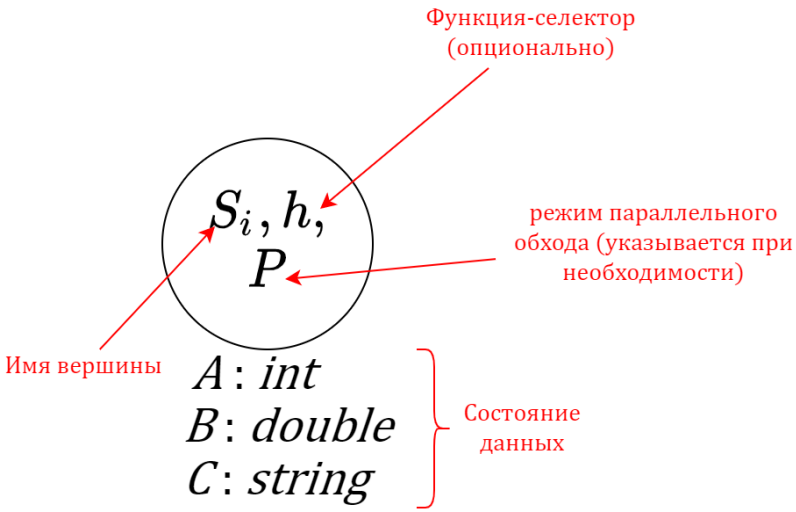


Рис. 9: Пример графовой модели, описывающей вычисление среднего арифметического и среднего геометрического двух половин массива целых чисел с последующей записью результатов в файл

Постановка задачи

↳ Основные понятия GBSE. Графовая модель



Атрибуты вершины:

1. имя;
2. состояние данных;
3. селектор;
4. режим параллельного обхода исходящих ветвей

- Были разработаны структуры данных, представляющие функции-предикаты, обработчики и селекторы.

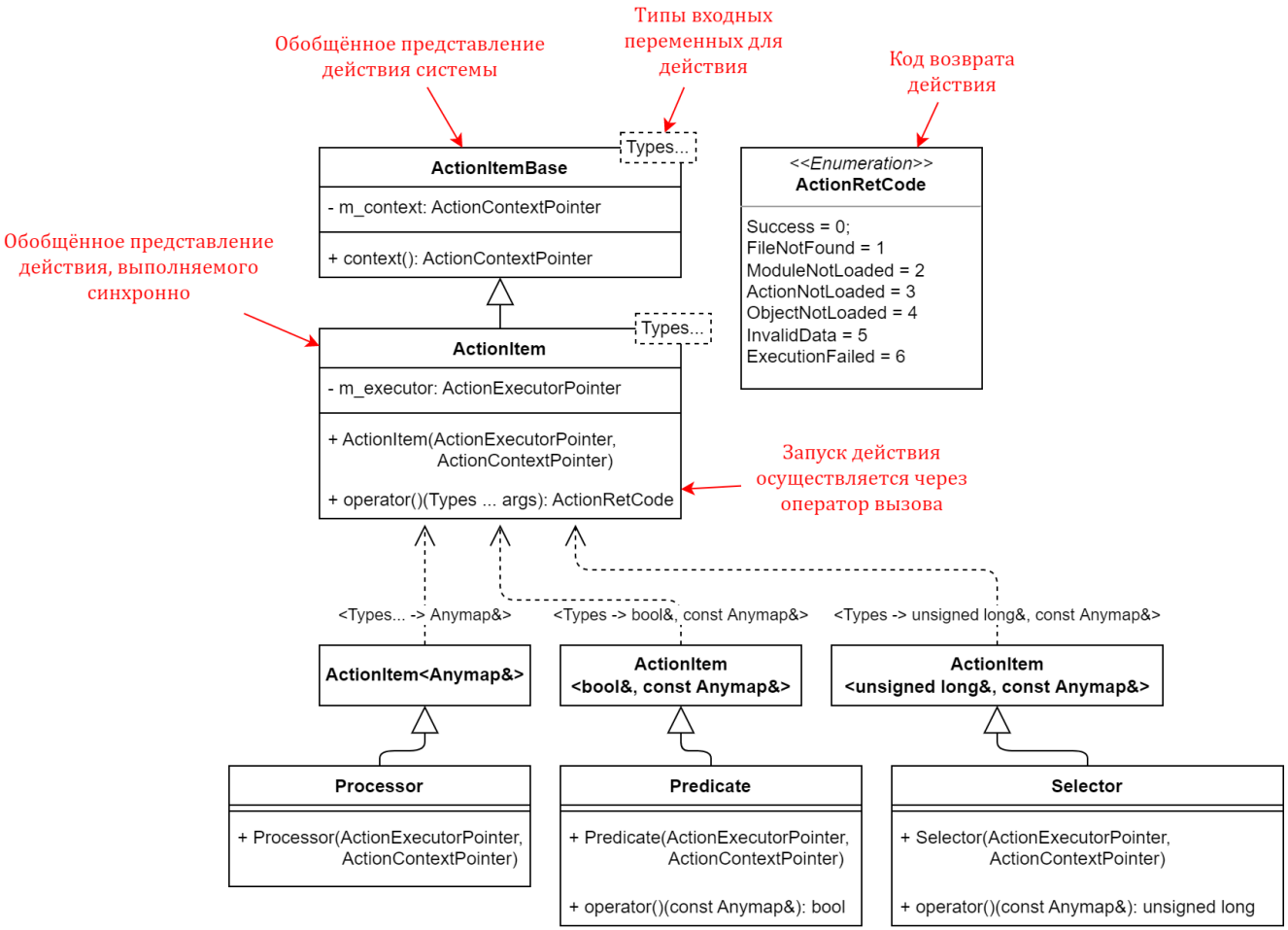


Рис. 10: UML-диаграмма разработанных функциональных структур данных

Программная реализация
↳ Информационные структуры данных

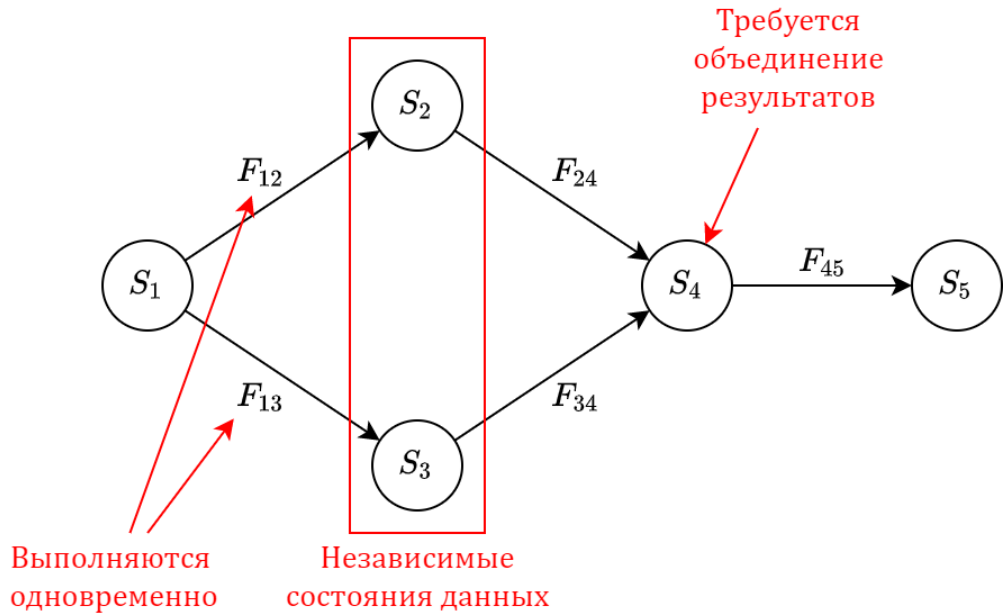


Рис. 11: Пример графовой модели, подразумевающей паралельный обход ветвей

- Режим параллельного исполнения в вершине S_1 определяет, какие ресурсы будут задействованы для выполнения ветвей $S_1 \rightarrow S_2 \rightarrow S_4$ и $S_1 \rightarrow S_3 \rightarrow S_4$

Программная реализация

↳ Управляющие структуры данных

В результате выполнения работы:

1) расширены функциональные возможности библиотеки comsdk
2) создана новая архитектура классов, позволяющая упростить процесс формирования программного представления графовых моделей;
3) разработаны структуры данных для программного представления графовых моделей алгоритмов и их элементов;
4) был разработан алгоритм, осуществляющий выполнение этапов алгоритма в соответствии с его графовой моделью;

Перспективы развития программного каркаса comsdk включают в себя:

- реализации алгоритма обхода графовой модели с задействованием различных вычислительных ресурсов
- интеграцию средства генерации форм ввода[1];
- разработку средства визуализации обрабатываемых данных;
- разработку средства автоматической документации реализуемых алгоритмов;

[1] Соколов А.П Першин А.Ю. Программный инструментарий для создания подсистем ввода данных при разработке систем инженерного анализа // Программная инженерия. 2017. Т. 8, № 12. С. 543–555.

Спасибо за внимание!