МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное
учреждение высшего образования
**"Южно-Уральский государственный университет
(национальный исследовательский университет)"**
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

РАБОТА ПРОВЕРЕНА

Рецензент
к.ф.-м.н., доцент,
заместитель декана по учебной
работе математического факультета
ЧелГУ

_____ С.А. Никитина

“___”_____ 2017 г.

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой,
д.ф.-м.н., профессор

_____ Л.Б. Соколинский

“___”_____ 2017 г.

РАЗРАБОТКА МЕТОДОВ ИССЛЕДОВАНИЯ ПАРАЛЛЕЛИЗМА АЛГОРИТМОВ НА ОСНОВЕ КОНЦЕПЦИИ Q-ДЕТЕРМИНАНТА И ИХ ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 02.04.02.2017.115-063.ВКР

Научный руководитель
к.ф.-м.н., доцент
_____ В.Н. Алеева

Автор работы,
студент группы КЭ-217
_____ А.Р. Багаутдинов

Ученый секретарь
(нормоконтролер)

_____ О.Н. Иванова

“___”_____ 2017 г.

Челябинск-2017

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность темы

В мире современных информационных технологий время однопроцессорных вычислительных систем закончилось. Уже сейчас в широком доступе для обычных пользователей находятся многоядерные архитектурные решения персональных компьютеров. Также технологии многопроцессорных вычислений используются все больше и больше, например, суперкомпьютеры, распределяющие вычисления по разным вычислительным узлам.

Со временем объем хранимых данных увеличивается, алгоритмы для их обработки становятся все сложнее и, таким образом, развиваются технологии, чтобы справляться с резко возросшими объемами хранения и обработки информации.

Как следствие, активно ведутся работы по разработке параллельного программного обеспечения и специальных средств для создания и оптимизации параллельных алгоритмов.

В основе любых программ, лежат некоторые алгоритмы, которые позволяют решать те или иные задачи. Считается, что алгоритм задает верхний предел возможности распараллеливания программы.

Представление алгоритма в форме Q-детерминанта – один из возможных методов его анализа. Такое представление позволяет определить ресурс параллелизма алгоритма, рассчитать такие характеристики, как число процессоров и число тактов работы вычислительной системы, которые требуются для выполнения его максимально быстрой реализации.

Таким образом, актуальной является задача разработки методов исследования параллелизма алгоритмов на основе концепции Q-детерминанта и их программная реализация.

Цель и задачи

Цель данной работы заключается в разработке методов исследования параллелизма алгоритмов на основе концепции Q-детерминанта и их программной реализации.

Для достижения этой цели необходимо решить следующие задачи:

1) изучить концепцию Q-детерминанта;

2) разработать методы исследования параллелизма алгоритмов на основе концепции Q-детерминанта;

3) разработать и протестировать программную систему для реализации полученных методов.

Обзор литературы

В ходе выполнения данной работы была изучена работа [4]. В этой работе описана концепция Q-детерминанта. Подход, основанный на концепции Q-детерминанта, дает возможность распараллелить максимально любой численный алгоритм, допускающий распараллеливание. Q-детерминант показывает Q-эффективную реализацию и делает алгоритм прозрачным с точки зрения структуры и реализации.

Также была изучена работа [12], в которой описывается обобщенная форма записи численных алгоритмов, методы анализа параллельных алгоритмов с использованием представления алгоритма в форме Q-детерминанта.

В работе [7] описан еще один подход к созданию параллельных программ – синтез параллельных программ. Этот подход заключается в том, чтобы из базы знаний существующих реализованных параллельных алгоритмов конструировать новые параллельные алгоритмы для решения более крупных задач.

Структура и объем работы

Работа состоит из введения, трех основных разделов, заключения и списка литературы. Объем работы составляет 32 страницы, объем библиографии – 22 наименования.

Краткое содержание работы

Первый раздел «Концепция Q-детерминанта» содержит описание представление алгоритма в форме Q-детерминанта, определение Q-эффективной реализации и описание ее характеристик.

Раздел «Проектирование и реализация программной системы» посвящен разработке методов исследования, проектированию и реализации программной системы, которая использует полученные методы.

Раздел «Тестирование программной системы» содержит описание проведенных тестов, показывающих корректность реализации программной системы.

В заключении суммируются результаты, полученные при выполнении данной работы.

1. КОНЦЕПЦИЯ Q-ДЕТЕРМИНАНТА

Пусть $B = \{b_1, b_2, \dots\}$ – счетное множество переменных арифметического или логического типа, Q – конечное множество операций, используемых алгоритмом.

Определим понятие выражения над множествами B и Q.

1. Константа и $b_i \in B$ являются выражениями. Их будем называть выражениями нулевого уровня вложенности.

2. Выражение, заключенное в круглые скобки, является выражением. Заключение выражения в круглые скобки не меняет его уровня вложенности.

3. Применяя одноместную операцию к выражению $(i-1)$-го уровня вложенности, где $i \geq 1$, получаем выражение i-го уровня вложенности, а исходное выражение называется его подвыражением $(i-1)$-го уровня вложенности. Саму операцию будем называть в этом случае операцией i-го уровня вложенности.

4. Применяя двухместную операцию к выражениям соответственно i_1 и i_2 уровней вложенности, получаем выражение $(i-1)$-го уровня вложенности, где $i = max(i_1, i_2) + 1$. Исходные выражения – его подвыражения соответственно i_1 и i_2 уровней вложенности. Саму базовую операцию назовем операцией i-го уровня вложенности.

Выражение, образованное в результате применения к n выражениям $n-1$ раз одной из ассоциативных операций множества Q, называется цепочкой данных выражений длины n.

Интерпретацией переменной $b_i \in B$ будем называть присвоение переменной b_i конкретного значения. Если задана интерпретация всех переменных, входящих в выражение, то будем говорить, что задана интерпретация выражения. Если задана интерпретация выражения, то его можно вычислить.

Рассмотрим множество параметров N, такое, что либо $N = \emptyset$, либо $N = \{n_1, \dots, n_л\}$, где $k \geq 1$, а n_i $(1 \leq i \leq k)$ принимает любые целые положительные значения.

Пусть $N = \{n_1, \ldots, n_k\}$. Тогда через *N* обозначим вектор $(\bar{n}_1, \ldots, \bar{n}_k)$, где n_i – некоторое заданное значение параметра n_i $(1 \leq i \leq k)$. Множество всевозможных векторов N будем обозначать $\{\bar{N}\}$.

Если $N = \emptyset$, то любое выражение над *B* и *Q* будем называть безусловным Q-термом. Таким образом, при $N = \emptyset$, понятие безусловного Q-терма и выражения над *B* и *Q* совпадают.

Пусть *V* – множество, состоящее из пустого выражения и всех выражений над *B* и *Q*.

Если $N \neq \emptyset$, то любое однозначное отображение $w: \{\bar{N}\} \to \mathrm{V}$ будем также называть безусловным Q-термом.

Далее под вычислением безусловного Q-терма *w* при интерпретации *B* будем понимать вычисление выражения *w*, если $N \neq \emptyset$, и вычисление выражения *w(N)* при некотором $N \in \{\bar{N}\}$, если $N \neq \emptyset$.

Если $N \neq \emptyset$, то уровнем вложенности безусловного Q-терма *w* будем называть уровень вложенности выражения *w* и обозначать T^w. Если $N \neq \emptyset$, то уровнем вложенности безусловного Q-терма *w* будем называть функцию $T^w: N \to T^w(\bar{N})$, где $T^w(\bar{N})$ – уровень вложенности выражения $w(\bar{N})$. Пусть $N \neq \emptyset$. Тогда, если выражение *w* при любой интерпретации *B* принимает значение логического типа, то безусловный Q-терм будем *w* называть безусловным логическим Q-термом.

Пусть $N \neq \emptyset$. Если при любом $N \in \{\bar{N}\}$ и любой интерпретации переменных принимает значение логического типа, то будем называть безусловным логическим Q-термом.

Пусть $w_1, \ldots, w_l$ – безусловные Q-термы, $u_1, \ldots, u_l$ – безусловные логические Q-термы.

Множество пар (u_i, w_i), где $i = 1, \ldots l$, обозначается $(\bar{u}, \bar{w}) = \{(u_i, w_i)\}_{i=1,\ldots,l}$ и называется условным Q-термом длины *l*.

Счетное множество пар безусловных Q-термов $(\bar{u}, \bar{w}) = \{(u_i, w_i)\}_{i=1,2,\ldots}$ называется условным бесконечным Q-термом, если $\{(u_i, w_i)\}_{i=1,\ldots,l}$ является

условным Q-термом для любого конечного *l*. Если не имеет значения, является Q-терм безусловным, условным или условным бесконечным, то его называют Q-термом.

Под вычислением безусловного Q-терма *w* при интерпретации *B* следует понимать вычисление выражения $w(\bar{N})$ при некотором $N \in \{\bar{N}\}$. Для вычисления при заданной интерпретации *B* и некотором $N \in \{\bar{N}\}$ условного Q-терма $(\bar{u}, \bar{w}) = \{(u_i, w_i)\}_{i=1,,\dots,l}$ необходимо, вычисляя выражения $u_i(\bar{N}), w_i(\bar{N})$, найти такие $u_{i_0}(\bar{N}), w_{i_0}(\bar{N})$, что $u_{i_0}(\bar{N})$ принимает значение *true*, а значение $w_{i_0}(\bar{N})$ определено. В качестве значения $(\bar{u}, \bar{w})$ необходимо взять $w_{i_0}(\bar{N})$,. Если установлено, что выражений $u_i(\bar{N}), w_i(\bar{N})$ не существует, то значение $(\bar{u}, \bar{w})$ для данной интерпретации *B* и данного *N* не определено. Для вычисления при заданной интерпретации *B* и некотором $N \in \{\bar{N}\}$ условного бесконечного Q-терма $(\bar{u}, \bar{w}) = \{(u_i, w_i)\}_{i=1,2,\dots}$ необходимо, вычисляя выражения $u_i(\bar{N}), w_i(\bar{N})$, найти $u_{i_0}(\bar{N}), w_{i_0}(\bar{N})$ такие, что $u_{i_0}(\bar{N})$ принимает значение true, а значение $w_{i_0}(\bar{N})$ определено. В качестве значения $(\bar{u}, \bar{w})$ нужно взять $w_{i_0}(\bar{N})$. Если установлено, что выражений $u_{i_0}(\bar{N}), w_{i_0}(\bar{N})$ не существует, то значение $(\bar{u}, \bar{w})$ для данной интерпретации *B* и данного *N* не определено.

Предположим, что I_1, I_2, I_3 – подмножества множества $I = (1, \dots, m)$ такие, что:

1) $I_1 \cup I_2 \cup I_3 = I$;
2) $I_i \cap I_j = \emptyset$ $(i \neq j; i, j = 1,2,3)$;
3) одно или два из множеств I_1, I_2, I_3 могут быть пустыми.

Рассмотрим множество Q-термов $\{f_i\}_{i \in I}$, которое удовлетворяет условиям:

1) $f_{i_1}(i_1 \in I_1)$ – безусловный Q-терм, $f_{i_1} = w^{i_1}$;
2) $f_{i_2}(i_2 \in I_2)$ – условный Q-терм, $f_{i_2} = \left\{\left(u_j^{l_2}, w_j^{l_2}\right)\right\}_{j=1,\dots,l_2}$, l_2 является вычислимой функцией параметров *N*;

3) $f_{i_3}(i_3 \in I_3)$ – условный бесконечный Q-терм, $f_{i_2} = \left\{\left(u_j^{i_2}, w_j^{i_2}\right)\right\}_{j=1,2,\dots}$.

Предположим, что алгоритм состоит в том, что для определения y_i $(i \in I)$ требуется вычислить Q-терм f_i. Тогда множество Q-термов f_i $(i \in I)$ называется Q-детерминантом алгоритма, а представление алгоритма в виде $y_i = f_i$ $(i \in I)$ представлением в форме Q-детерминанта.

Реализацией алгоритма, представленного в форме Q-детерминанта $y_i = f_i$ $(i \in I)$, называется вычисление Q-термов f_i $(i \in I)$ при заданной интерпретации B и некотором $N \in \{\overline{N}\}$.

Если реализация такова, что выражения

$W(\overline{N}) = \left\{w^{i_1}(\overline{N})(i_1 \in I_1);\ u_j^{i_2}(\overline{N}), w_j^{i_2}(\overline{N})\left(i_2 \in I_2, j = 1, \dots, l_{i_2}\right); u_j^{i_3}(\overline{N}), w_j^{i_3}(\overline{N}), (i_3 \in I_3, j = 1,2, \dots)\right\}$

вычисляются одновременно и при вычислении каждого из выражений операции выполняются по мере их готовности, то такая реализация называется Q-эффективной. С формальной точки зрения Q-эффективная реализация алгоритма является максимально быстрой.

Выполнимой реализацией алгоритма будем называть такую реализацию, в которой одновременно необходимо выполнять конечное число операций. Существуют такие алгоритмы, для которых Q-эффективная реализация не является выполнимой.

Q-эффективная реализация обладает следующими характеристиками:

- $D\alpha$ – число тактов работы ВС, необходимое для выполнения Q-эффективной реализации;
- $P\alpha$ – количество вычислительных узлов ВС, требующихся для Q-эффективной реализации;
- $S\alpha$ – ускорение выполнения параллельного алгоритма относительно последовательного варианта выполнения вычислений;
- $E\alpha$ – эффективность использования вычислительных узлов, определяет среднюю долю времени выполнения алгоритма на $P\alpha$ вычислительных узлах, в течение которого узлы действительно используются для решения поставленной задачи;

- $T\alpha$ – иерархическая структура, описывающая последовательность выполненных операций, данную структуру можно представить в виде ориентированного графа, вершины которого содержит данные об операциях, выполняемых алгоритмом.

2. ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ ПРОГРАММНОЙ СИСТЕМЫ

2.1. Форма записи алгоритма

Для анализа ресурса параллелизма необходимо построить Q-детерминант. В связи с этим необходимо решить, на основе какого способа записи алгоритма он будет построен. Для записи алгоритма существуют различные способы. Один из таких способов – графическая запись с помощью блок-схемы. В данной работе было принято решение выполнять построение Q-детерминанта по его блок-схеме.

Описание алгоритма с помощью блок-схем осуществляется изображением последовательности геометрических фигур, каждая из которых подразумевает выполнение некоторого определенного действия алгоритма. Написание алгоритмов с помощью блок-схем регламентируется ГОСТом [11].

Из всего множества допустимых элементов блок-схемы выделим следующие:

- данные;
- процесс – отображает функцию обработки данных любого вида;
- решение – отображает решение или функцию переключательного типа, имеющую один вход и ряд альтернативных выходов, один и только один из которых может быть активизирован после вычисления условий, определенных внутри этого символа;
- терминатор – отображает выход во внешнюю среду и вход из внешней среды.

Приведем некоторые ограничения и дополнительные уточнения на используемые блок-схемы.

Блок-схема алгоритма имеет два элемента-терминатора, один из которых означает начало алгоритма, а другой – конец. Элемент «Начало» имеет одну исходящую связь и не имеет входящих, элемент «Конец» имеет одну входящую и не имеет исходящих связей.

Элемент «Решение» имеет две исходящие связи, одна из которых соответствует передаче управления, в случае если данное условие истинно, а другая, если оно ложно.

Будем считать, что элемент «Процесс» содержит не более одного действия, при этом действием должно являться присваивание значения переменной. Группы операций, изменяющих переменные, заменяются на цепочки элементов.

Разделим элементы «Данные» на элементы «Входные данные» и «Выходные данные». Элемент «Входные данные» подразумевает получение алгоритмом некоторых данных и должен располагаться сразу после элемента «Начало». Элемент «Выходные данные» подразумевается вывод информации из алгоритма (получение результата выполнения) и должен располагаться перед элементом «Конец».

Такие ограничения обусловлены свойствами алгоритмов (детерминированностью, понятностью, конечностью записи, дискретностью), а также необходимостью однозначного определения входных и выходных данных.

2.2. Методы построения Q-детерминанта

Для разработки методов анализа параллелизма алгоритмов посредством построения Q-детерминанта по блок-схеме было необходимо рассмотреть множество различных алгоритмов и проанализировать особенности и проблемы, связанные с реализацией этих методов.

В ходе работы были разработаны следующие методы построения Q-детерминанта на основе блок-схемы:

- построение Q-детерминанта, состоящего из безусловных Q-термов;
- построение Q-детерминанта, состоящего из условных и условных бесконечных Q-термов.

Особенность алгоритмов, Q-детерминант которых состоит из безусловных Q-термов, заключается в том, что элементы присваивания в блок-схеме не затрагивают входных данных. Таким образом, можно осуществлять прохождение по блок-схеме последовательно, как если бы выполнялся рассматриваемый алгоритм. В Q-терм будет записано конечное выражение.

Для алгоритмов, Q-детерминант которых состоит из условных Q-термов, блок-схемы содержат элементы присваивания, которые затрагивают входные данные. Таким образом, появляется необходимость в создании контейнеров для входных переменных, в которых будут храниться новые значения переменных в результате присваиваний. Эти значения будут использоваться для формирования последующих выражений и Q-термов. Также прохождение по блок-схеме будет разветвлено и каждая ветвь будет обрабатываться отдельно. По завершению обработки одной ветви обработчик блок-схемы вернется на прежний блок и продолжит обработку уже с противоположным условием, записанном в данном блоке. Q-термы теперь представляют собой не некоторое значение, а набор пар условие-значение.

То же самое касается и алгоритмов, Q-детерминант которых состоит из условных бесконечных Q-термов. Проблема хранения условных бесконечных Q-термов решается ограничением количества итераций. Для таких алгоритмов можно определить условие сходимости, которое и позволит определить необходимое количество итераций.

Блок-схема обрабатывается сверху вниз. Такое направление обусловлено тем, что оно позволяет последовательно проходить по блок-схеме и следить за тем, как строится Q-детерминант.

2.3. Модули программной системы

В рамках данной работы была разработана программная система, позволяющая строить Q-детерминант алгоритма по его соответствующей блок-схеме.

Программная система реализована с использованием языка программирования C# на базе платформы .NET. Для описания входных и выходных данных используется декларативный язык JSON. Система разрабатывалась в соответствии парадигме объектно-ориентированного программирования. В качестве интегрированной среды разработки использовалась Microsoft Visual Studio 2015.

Программная система включает в себя следующие модули:

- модуль структуры данных – содержит в себе основные классы для работы с формированием Q-детерминанта;
- модуль обработки блок-схемы – содержит методы прохождения и формирования новых добавлений в Q-детерминант;
- модуль обработки выражений (парсер) – необходим для разбора выражений внутри блок-схемы.

На рис. 1 изображена диаграмма вариантов использования для разрабатываемой системы.

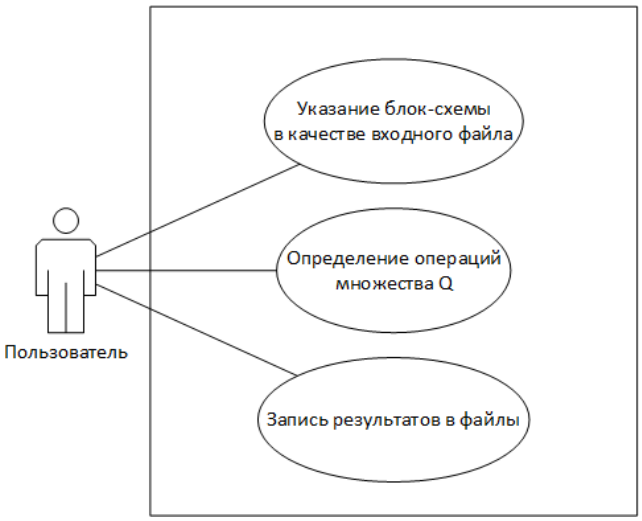

Рис. 1. Диаграмма вариантов использования

2.4. Структуры и форматы данных

В качестве формата представления блок-схемы алгоритма был формат JSON, так как он позволяет работать с собственным текстовым отображением.

Блок-схема представляется в виде описания блоков `Verticles` и соединений `Edges`. Блоки `Verticles` определяются номером `Id`, типом `Type`, и текстовым содержимым `Content`.

Значения `type` для блоков `Verticles`:

0 – блок «Начало»;

1 – блок «Конец»;

2 – блок присваивания;

3 – блок условия;

4 – блок ввода;

5 – блок вывода.

Соединения `Edges` определяются номерами начальных и конечных блоков `From` и `To` и типом соединения `type`.

Значения `type` для соединений `Edges`:

0 – проход по условию «нет»;

1 – проход по условию «да»;

2 – обычное соединение.

Таким образом, блок-схема алгоритма умножения матриц будет представлена в следующем виде:

```
{"Vertices":[     {"Id":1,"Type":0,"Content":"Start"},
{"Id":2,"Type":4,"Content":"l"},
{"Id":3,"Type":4,"Content":"m"},
{"Id":4,"Type":4,"Content":"n"},
{"Id":5,"Type":4,"Content":"A[l,m]"},
{"Id":6,"Type":4,"Content":"B[m,n]"},
{"Id":7,"Type":2,"Content":"C[l,n]=0"},
{"Id":8,"Type":2,"Content":"i=1"},
{"Id":9,"Type":3,"Content":"i<=l"},
{"Id":10,"Type":2,"Content":"j=1"},
{"Id":11,"Type":3,"Content":"j<=n"},
{"Id":12,"Type":2,"Content":"k=1"},
{"Id":13,"Type":3,"Content":"k<=m"},
{"Id":14,"Type":2,"Content":"C(i,j)=C(i,j)+A(i,k)*B(k,j)"},
{"Id":15,"Type":2,"Content":"k=k+1"},
{"Id":16,"Type":2,"Content":"j=j+1"},
{"Id":17,"Type":2,"Content":"i=i+1"},
{"Id":18,"Type":5,"Content":"C[l,n]"},
{"Id":19,"Type":1,"Content":"End"},
], "Edges":[
{"From":1,"To":2,"Type":2},   {"From":2,"To":3,"Type":2},
{"From":3,"To":4,"Type":2},   {"From":4,"To":5,"Type":2},
{"From":5,"To":6,"Type":2},   {"From":6,"To":7,"Type":2},
{"From":7,"To":8,"Type":2},   {"From":8,"To":9,"Type":2},
{"From":9,"To":10,"Type":1},  {"From":9,"To":18,"Type":0},
{"From":10,"To":11,"Type":2}, {"From":11,"To":12,"Type":1},
```

```
{“From”:11,”To”:17,”Type”:0}, {“From”:12,”To”:13,”Type”:2},
{“From”:13,”To”:14,”Type”:1}, {“From”:13,”To”:16,”Type”:0},
{“From”:14,”To”:15,”Type”:2}, {“From”:15,”To”:13,”Type”:2},
{“From”:16,”To”:11,”Type”:2}, {“From”:17,”To”:9,”Type”:2},
{“From”:18,”To”:19,”Type”:2},]}
```

Для получаемых выражений в Q-термах используется следующая структура: выражение может быть числом, бинарным или унарным выражением, а операндами выражения могут являться любые из перечисленных выше типов выражений. Выражение представлено в виде интерфейса `IExpression`, этот интерфейс реализуют такие классы, как `BinaryExpression`, `UnaryExpression`, `Number` и `Variable`. `Number` является числом в формате `double`, `Variable` представляет собой текстовое наименование переменной. В `BinaryExpression` и `UnaryExpression` операндами являются два либо один элемент `IExpression`, а также математическая операция `ArithmeticOperation`. Условия хранятся аналогично бинарным выражениям в классе `ConditionExpression`, но вместо операции стоит знак сравнения `RelationalOperator`.

Безусловные Q-термы хранятся в виде выражения, условные Q-термы хранят в себе список пар условие-выражение. В случае условных бесконечных Q-термов ограничивается количество итераций при прохождении по блок-схеме, поэтому при хранении они вырождаются в условные Q-термы.

На рис. 2 представлена диаграмма классов, описывающая структуру формируемых данных.

Q-детерминанты могут храниться как в строковом формате, так и в формате JSON. Например, один из Q-термов для алгоритма умножения матриц будет выглядеть в строковом формате следующим образом:

+(+(+(*(A(1,1), B(1,1)), *(A(1,2), B(2,1))), *(A(1,3), B(3,1)))), *(A(1,4), B(4,1))).

Список возможных операций представлен в файле `operations.json`, в котором описываются бинарные и унарные операции и соответствующие им функции.

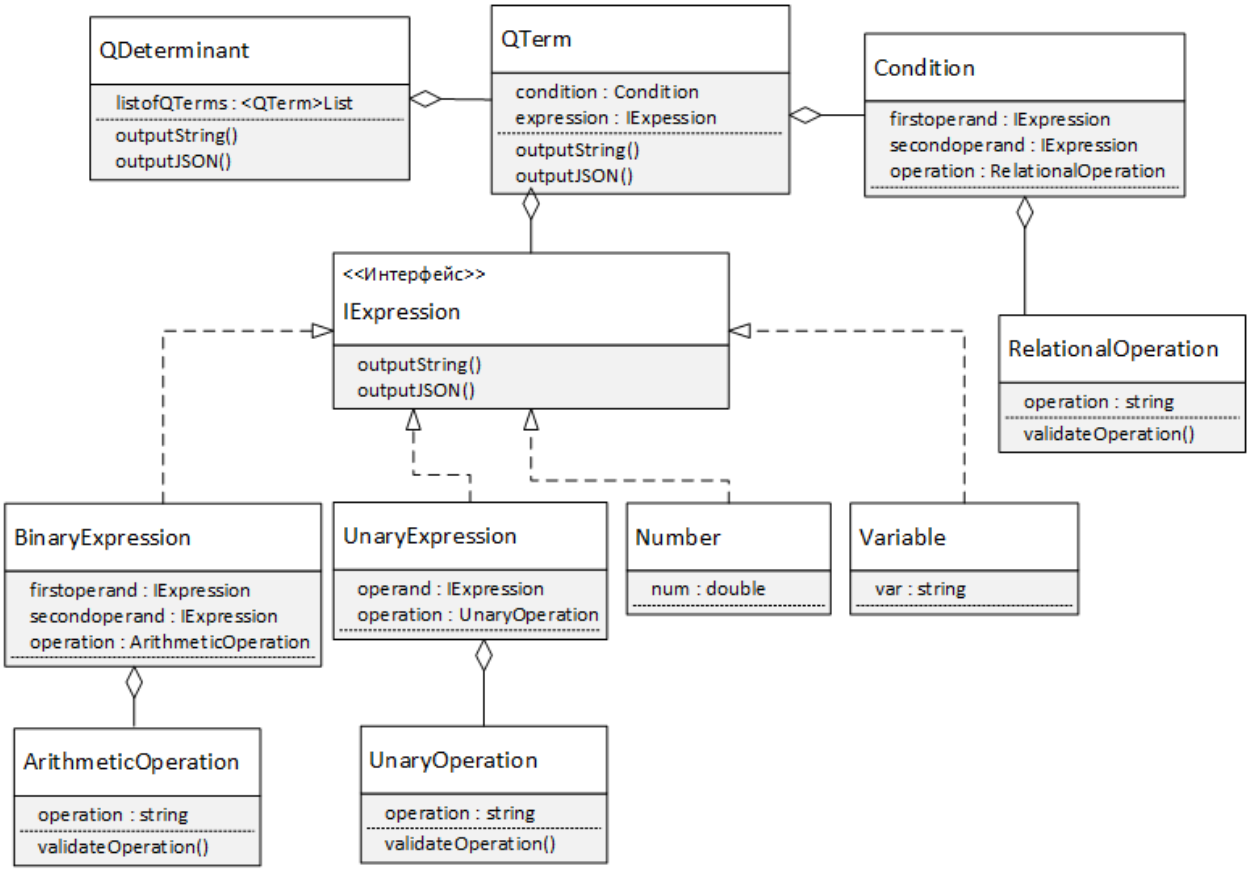

Рис. 2. Диаграмма классов, описывающая структуру формируемых данных

В формате JSON:

```
{
	"operation": "+",
	"firstOperand":
	{
		"operation": "+",
		"firstOperand":
		{
			"operation": "+",
			"firstOperand":
			{
				"operation": "*",
				"firstOperand": "A(1,1)",
				"secondOperand": "B(1,1)"
			},
			"secondOperand":
			{
				"operation": "*",
				"firstOperand": "A(1,2)",
				"secondOperand": "B(2,1)"
			}
		},
		"secondOperand":
		{
			"operation": "*",
			"firstOperand": "A(1,3)",
```

```
                "secondOperand": "B(3,1)"
            }
        },
        "secondOperand":
        {
            "operation": "*",
            "firstOperand": "A(1,4)",
            "secondOperand": "B(4,1)"
        }
}
```

2.5. Модуль обработки блок-схемы

Диаграмма классов модуля обработки блок-схемы представлена на рис. 3.

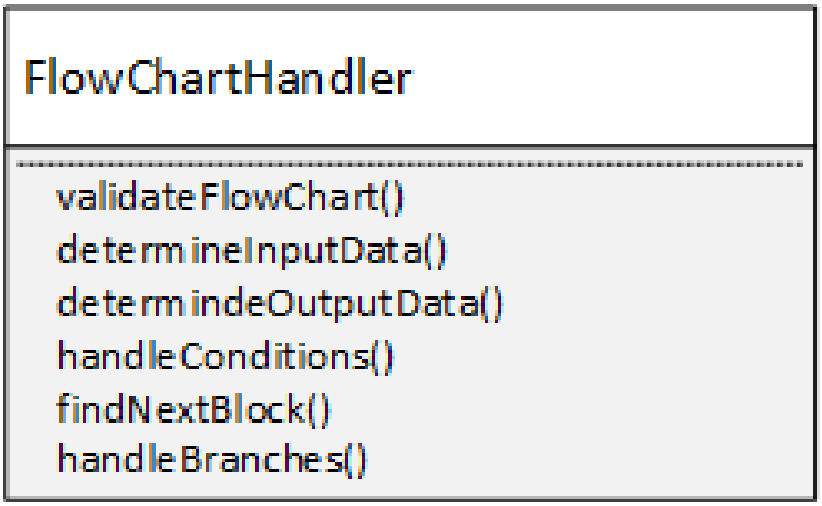

Рис. 3. Диаграмма классов модуля обработки блок-схемы

Фунция `validateFlowChart()` проверяет блок-схему на ее корректность. Функции `determineInputData()` и `determineOutputData()` определяют входные и выходные данные алгоритма и в соответствии с ними формируют контейнеры переменных и необходимое количество Q-термов. Функция `handleConditions()` обрабатывает условия и проверяет, включают ли эти условия входные данные. Если да, то это говорит о необходимости формирования условных Q-термов с соответствующими условиями. Функции `findNextBlock()` и `handleBranches()` отвечают за прохождение по блок-схеме и обработку различных ветвей в ходе прохождения по ним.

2.6. Модуль обработки выражений

Диаграмма классов модуля обработки выражений представлена на рис. 4.

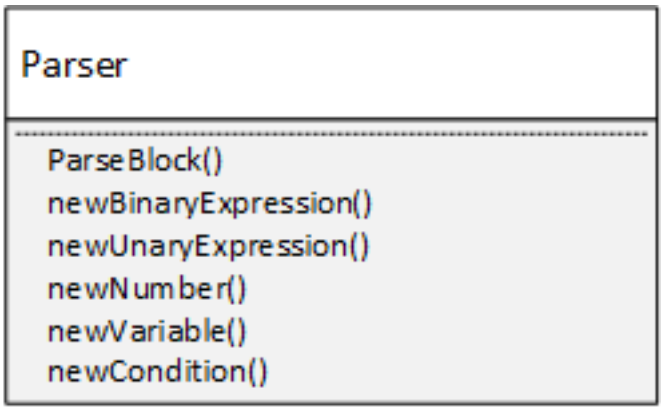

Рис. 4. Диаграмма классов модуля обработки выражений

Функция `ParseBlock()` обрабатывает содержимое текущего блока и формирует новые выражения, числа, переменные и условия при помощи функций `newBinaryExpression()`, `newUnaryExpression`, `newNumber()`, `newVariable()`, `newCondition()`.

3. ТЕСТИРОВАНИЕ ПРОГРАММНОЙ СИСТЕМЫ

Для того, чтобы убедиться в правильности и адекватности метода построения Q-детерминанта, необходимо провести некоторые контрольные эксперименты. Для этого рассмотрим некоторые из алгоритмов, Q-детерминанты которых будут состоять из безусловных, условных и условных бесконечных Q-термов. Результаты построения и анализа приводятся далее в этой главе.

3.1. Умножение матриц

Алгоритм умножения матриц $A_{[l\times m]}$ и $B_{[m\times n]}$ в виде блок-схемы представлен на рис. 5.

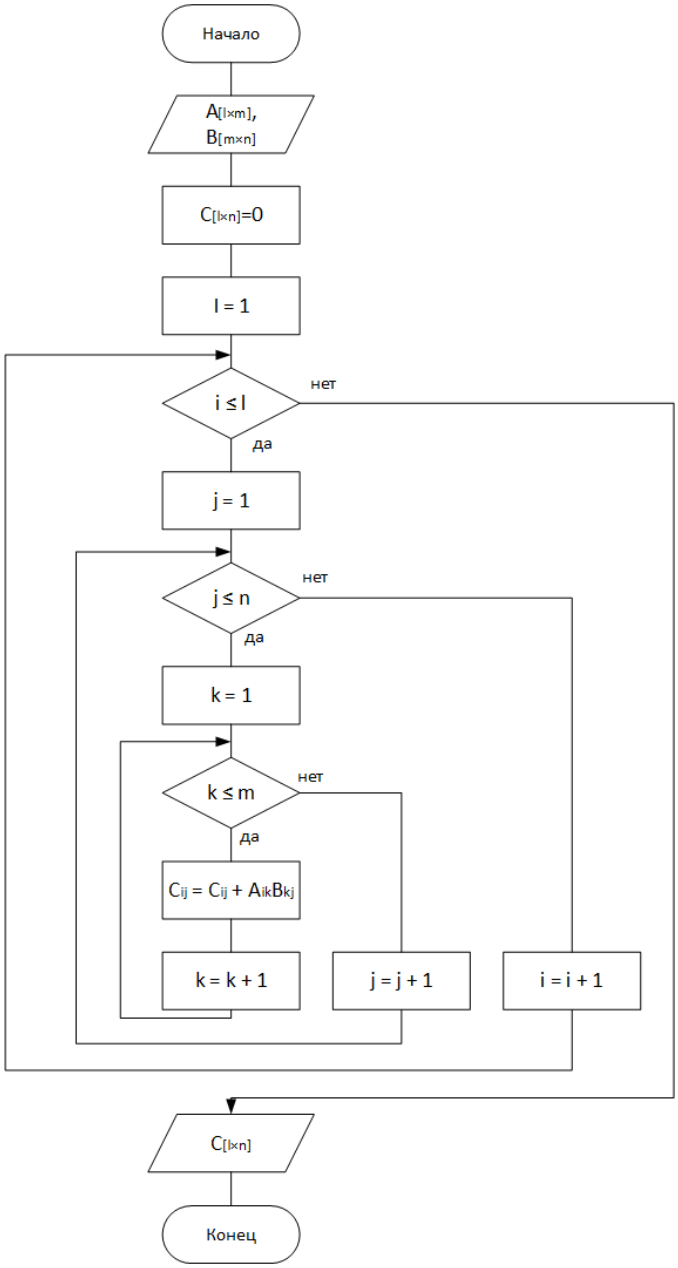

Рис. 5. Блок-схема алгоритма умножения матриц

Алгоритм умножения матриц представляет собой тройной вложенный цикл.

В каждом цикле счетчики не зависят от входных данных, но определяют, какие элементы входных данных повлияют на конкретный Q-терм.

Таким образом, прохождение блок-схемы будет строго последовательным.

Q-детерминант будет представлен в виде $l \cdot n$ безусловных Q-термов:

$$C_{ij} = \sum_{k=1}^{m} A_{ik} B_{kj}, (i = 1, \dots, l, j = 1, \dots, n)$$

Результат выполнения работы для размерностей l = 3, m = 2, n = 4 представлен на рис. 6.

```
+(*(A(1,1), B(1,1)), *(A(1,2), B(2,1))
+(*(A(1,1), B(1,2)), *(A(1,2), B(2,2))
+(*(A(1,1), B(1,3)), *(A(1,2), B(2,3))
+(*(A(1,1), B(1,4)), *(A(1,2), B(2,4))
+(*(A(2,1), B(1,1)), *(A(2,2), B(2,1))
+(*(A(2,1), B(1,2)), *(A(2,2), B(2,2))
+(*(A(2,1), B(1,3)), *(A(2,2), B(2,3))
+(*(A(2,1), B(1,4)), *(A(2,2), B(2,4))
+(*(A(3,1), B(1,1)), *(A(3,2), B(2,1))
+(*(A(3,1), B(1,2)), *(A(3,2), B(2,2))
+(*(A(3,1), B(1,3)), *(A(3,2), B(2,3))
+(*(A(3,1), B(1,4)), *(A(3,2), B(2,4))
```

Рис. 6. Результат обработки алгоритма умножения матриц

3.2. Нахождение максимального элемента массива

Алгоритм поиска максимального числа в массиве в виде блок-схемы представлен на рис. 7.

При итерации счетчика Q-детерминант будет формироваться следующим образом, представленном на таблице 1.

Таким образом, при увеличении счетчика к каждому из уже сформированных условий будут формироваться два новых, полученных при помощи операции конъюнкции.

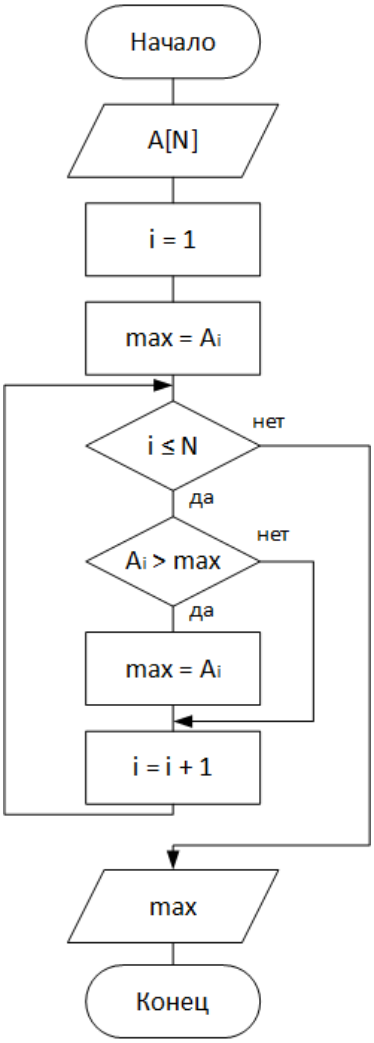

Рис. 7. Блок-схема алгоритма нахождения максимального числа в массиве

Табл. 1. Формирование Q-детерминанта для алгоритма нахождения максимального числа в массиве

i	max
1	A_1
2	$(A_2 > A_1, A_2)$, $(A_2 \leq A_1, A_1)$
3	$((A_2 > A_1)\ \&\ (A_3 > A_2), A_3)$, $((A_2 > A_1)\ \&\ (A_3 \leq A_2), A_2)$, $((A_2 \leq A_1)\ \&\ (A_3 > A_1), A_3)$, $((A_2 \leq A_1)\ \&\ (A_3 \leq A_1), A_1)$
…	…

Результат выполнения работы для размерности N = 3 представлен на рис. 8.

```
(&(>(A(2), A(1)), >(A(3), A(2))), A(3)), (&(>(A(2), A(1)), <=(A(3), A(2))),
A(2)),(&(<=(A(2), A(1)), >(A(3), A(1))), A(3)), (&(<=(A(2), A(1)), >(A(3), A(1))), A(1))
```

Рис. 8. Результат обработки алгоритма нахождения максимального числа в массиве

3.3. Алгоритм Евклида

Алгоритм Евклида для нахождения наибольшего общего делителя в виде блок-схемы представлен на рис. 9.

Выражение *a* mod *b* означает вычисление остатка от деления числа *a* на число *b*.

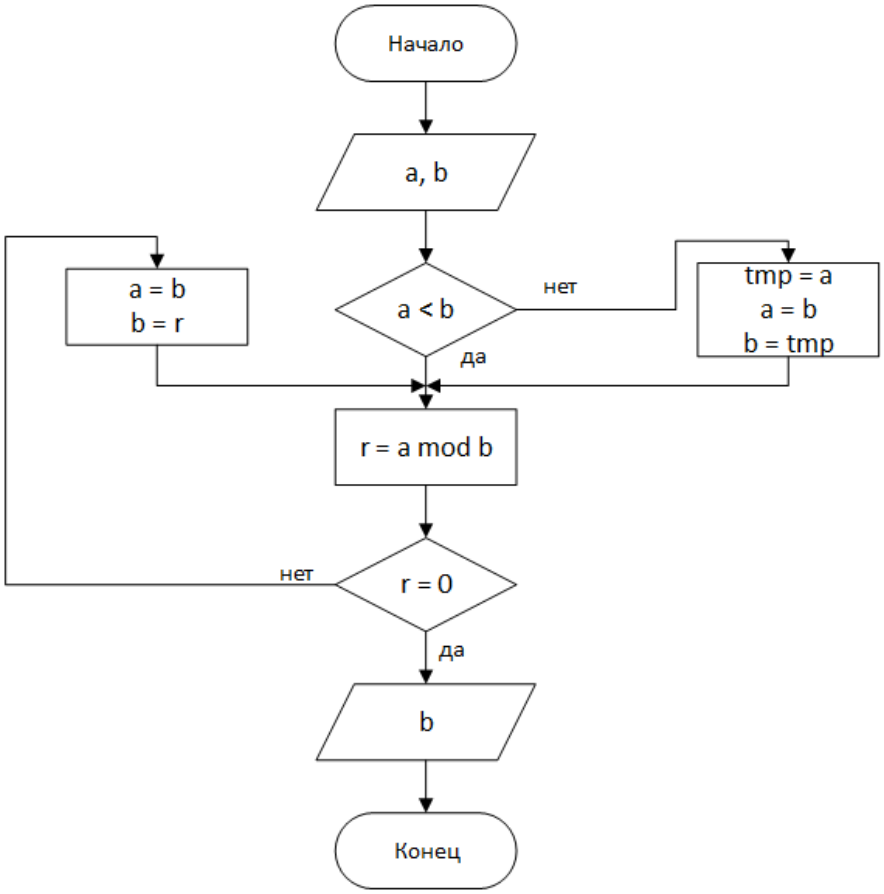

Рис. 9 – Блок-схема алгоритма Евклида для нахождения наибольшего общего делителя

В данном алгоритме входные переменные *a* и *b* с каждой итерацией принимают новые значения. Таким образом, обработчик блок-схемы будет сохранять новые значения в соответствующие контейнеры входных переменных, и

относительно этих новых значений строить получаемые выражения. Это же касается блока с перестановкой переменных. Новые значения входных переменных на каждой последующей представлены в таблице 2.

Табл. 2. Формируемые значения переменных

a	b
b	$a \bmod b$
$a \bmod b$	$b \bmod (a \bmod b)$
$b \bmod (a \bmod b)$	$(a \bmod b) \bmod (b \bmod (a \bmod b))$
$(a \bmod b) \bmod (b \bmod (a \bmod b))$	$(b \bmod (a \bmod b)) \bmod (a \bmod b) \bmod (b \bmod (a \bmod b))$
…	

Таким образом, Q-детерминант для алгоритма Евклида примет следующий вид:

$\{\big((a \geq b)\&(a \bmod b) = 0, b\big),$

$\big((a \geq b)\&(a \bmod b \neq 0)\&(b \bmod (a \bmod b) = 0), a \bmod b\big),$

$((a \geq b)\&(a \bmod b \neq 0)\&(b \bmod (a \bmod b) \neq 0)\&$

$\&((a \bmod b) \bmod \big(b \bmod (a \bmod b)\big) = 0, b \bmod (a \bmod b))),$

…,

$\big((a < b)\&(b \bmod a) = 0, a\big),$

$\big((a < b)\&(b \bmod a \neq 0)\&(a \bmod (b \bmod a) = 0), b \bmod a\big),$

$((a < b)\&(b \bmod a \neq 0)\&(a \bmod (b \bmod a) \neq 0)\&$

$\&((b \bmod a) \bmod \big(a \bmod (b \bmod a)\big) = 0, a \bmod (b \bmod a)), \dots\}$

Q-детерминант состоит из условного бесконечного Q-терма, а значит необходимо ограничить количество повторов тела цикла. Можно сделать это непосредственно в самой блок-схеме.

Результат выполнения работы при ограничении до трех итераций будет выглядеть так, как представлено на рис. 10.

```
(&(>=(a, b), =(mod(a, b), 0)), b), (&(&(>=(a, b), !=(mod(a, b), 0)), =(mod(b, mod(a,
b)), 0)), mod(a, b)), (&(&(&(>=(a, b), !=(mod(a, b), 0)), !=(mod(b, mod(a, b)), 0)),
=(mod(mod(a, b), mod(b, mod(a, b))), 0)), mod(b, mod(a, b))), (&(<(a, b), =(mod(b, a),
0)), a), (&(&(<(a, b), !=(mod(b, a), 0)), =(mod(a, mod(b, a)), 0)), mod(b, a)),
(&(&(&(<(a, b), !=(mod(b, a), 0)), !=(mod(a, mod(b, a)), 0)), =(mod(mod(b, a), mod(a,
mod(b, a))), 0)), mod(a, mod(b, a)))
```

Рис. 10. Результат обработки алгоритма Евклида с ограничением итераций

3.4. Метод Якоби для решения систем линейных уравнений

Пусть дана система линейных уравнений

$A\bar{x} = \bar{b}$, где $A = \left|a_{ij}\right|_{i,j=1,\dots,n}$, $\bar{x} = (x_1, \dots, x_n)^T$, $\bar{b} = (b_1, \dots, b_n)^T$.

Предполагаем, что $a_{ii} \neq 0, \;\; i = 1, \dots, n.$

Примем некоторое начальное приближение $\bar{x}^0$. Тогда итерационный процесс будет выглядеть следующим образом:

$$x_i^{k+1} = \frac{b_i}{a_{ii}} - \sum_{j=1,\dots,n; j\neq i} \frac{a_{ij}}{a_{ii}} x_j^k.$$

Критерий для окончания итерационного процесса:

$\left\|\bar{x}^{k+1} - \bar{x}^k\right\| < \varepsilon$, где ε – точность вычислений.

Алгоритм решения систем линейных уравнений методом Якоби в виде блок-схемы представлен на рис. 12.

Q-детерминант алгоритма решения систем линейных уравнений методом Якоби состоит из n условных бесконечных Q-термов и представляется следующим образом:

$$x_i = \left\{\left(\|\bar{x}^1 - \bar{x}^0\| < \varepsilon, x_i^1\right), \left(\|\bar{x}^2 - \bar{x}^1\| < \varepsilon, x_i^2\right), \dots, \left(\left\|\bar{x}^{k+1} - \bar{x}^k\right\| < \varepsilon, x_i^k\right), \dots\right\}.$$

Результат обработки блок-схемы программой для Q-терма x_1 на первой итерации в строковом формате представлен на рис. 11.

```
(<(+(+(+(abs(-(X(1), -(-(-(B(1), *(A(1,2), X(2))), *(A(1,3), X(3))), *(A(1,4), X(4))))),
abs(-(X(2), -(-(-(B(2), *(A(2,1), X(1))), *(A(2,3), X(3))), *(A(2,4), X(4)))))),
abs(-(X(3), -(-(-(B(3), *(A(3,1), X(1))), *(A(3,2), X(2))), *(A(3,4), X(4)))))),
abs(-(X(4), -(-(-(B(4), *(A(4,1), X(1))), *(A(4,2), X(2))), *(A(4,3), X(3)))))), e),
-(-(-(B(1), *(A(1,2), X(2))), *(A(1,3), X(3))), *(A(1,4), X(4))))
```

Рис. 11. Q-терм x_1, полученный на первой итерации

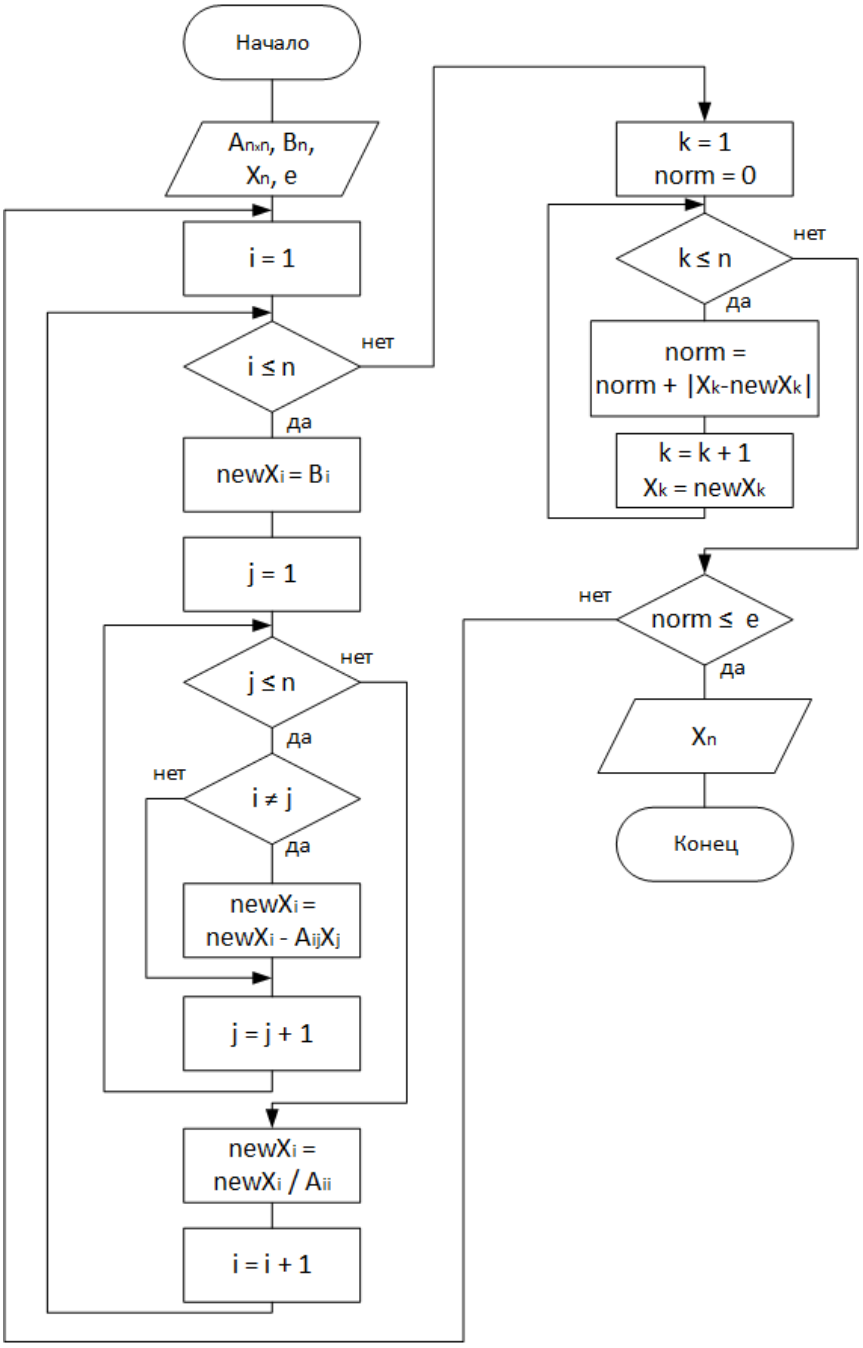

Рис. 12. Блок-схема алгоритма решения систем линейных уравнений методом Якоби

ЗАКЛЮЧЕНИЕ

Данная работа посвящена разработке методов исследования параллелизма алгоритмов на основе концепции Q-детерминанта и их программной реализации.

В ходе работы были получены следующие результаты:

1) изучена концепция Q-детерминанта;

2) разработаны методы исследования параллелизма алгоритмов на основе концепции Q-детерминанта;

3) разработана и протестирована программная система для реализации полученных методов.

В дальнейшем необходимо дополнить систему такими возможностями, как анализ Q-детерминантов алгоритма для различных размерностей, нахождение характеристик параллельной сложности алгоритма, таких как число тактов работы и количество процессоров, требуемых для выполнения Q-эффективной реализации.

По данной работе имеется публикация [5].

Работа была выполнена при финансовой поддержке РФФИ в рамках научного проекта № 17-07-00865-а (2017-2019 гг.) «Метод исследования ресурса параллелизма численных алгоритмов и суперкомпьютерный кодизайн на основе концепции Q-детерминанта».

ЛИТЕРАТУРА

1. JSON Schema: core definitions and terminology. // The Internet Engineering Task Force (IETF). [Электронный ресурс] URL: https://tools.ietf.org/id/draft-zyp-json-schema-04.html (дата обращения: 16.04.2017).

2. JSON.NET – Documentation. // Json.NET – James Newton-King. [Электронный ресурс] URL: http://www.newtonsoft.com/json/help/html/Introduction.htm (дата обращения: 16.04.2017).

3. The application/json Media Type for JavaScript Object Notation (JSON). // The Internet Engineering Task Force (IETF). [Электронный ресурс] URL: http://tools.ietf.org/pdf/rfc4627.pdf (дата обращения: 25.04.2017).

4. Алеева В.Н. Анализ параллельных численных алгоритмов: Препринт № 590. – Новосибирск, 1985. – 23 с. В надзаг.: ВЦ СО АН СССР.

5. Алеева В.Н., Багаутдинов А.Р., Валькевич Н.В. Исследование ресурса параллелизма численных алгоритмов на основе концепции Q-детерминанта. // Параллельные вычислительные технологии – XI международная конференция, ПаВТ’2017, г. Казань, 3–7 апреля 2017 г. Короткие статьи и описания плакатов. – Челябинск: Издательский центр ЮУрГУ, 2017. – С. 504.

6. Буч Г., Рамбо Д., Якобсон И. Язык UML. Руководство пользователя. 2-е изд. / Пер. с англ. – М.: ДМК Пресс, 2007. – 496 с.

7. Вальковский В.А., Малышкин В.Э. Синтез параллельных программ и систем на вычислительных моделях. – Новосибирск: Наука. 1988. – 128 с.

8. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. – СПб.: БХВ-Петербург, 2002. – 606 с.

9. Гамма Э. Приемы объектно-ориентированного проектирования. / Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес. – Паттерны проектирования. – СПб.: Питер, 2012. – 368 с.

10. Гергель В.П., Стронгин Р.Г. Основы параллельных вычислений для многопроцессорных вычислительных систем. Учебное пособие: Изд. 2-е, доп-е. – Нижний Новгород: Изд-во ННГУ, 2003. – 400 с.

11. ГОСТ 19.701-90. Схемы алгоритмов, программ, данных и систем: условные обозначения и правила выполнения.

12. Игнатьев С.В. Определение ресурса параллелизма алгоритмов на базе концепции Q-детерминанта: Вып. квалиф. работа магистра прикладной математики и информатики: 010500.68 / Южно-Уральский государственный университет. – Челябинск, 2009. – 75 л.

13. Кнут Д. Искусство программирования. Том 1: Основные алгоритмы. 3-е изд. / Пер. с англ. / Уч. пос. – М. : Издательский дом «Вильямс», 2000. – 720 с.

14. Мак-Дональд М. WPF: Windows Presentation Foundation в .NET 4.5 с примерами на C# 5.0 для профессионалов. / Пер. с англ. – М.: Вильямс, 2013. – 1024 с. : ил.

15. Рамбо Д. Блаха М. UML 2.0. Объектно-ориентированное моделирование и разработка. 2-е изд. – СПб.: Питер, 2007. – 554 с

16. Свирихин Д.И., Алеева В.Н. Определение максимально эффективной реализации алгоритма на основе концепции Q-детерминанта. // Параллельные вычислительные технологии (ПаВТ'2013): труды международной научной конференции (1 – 5 апреля 2013 г., г. Челябинск). – Челябинск: Издательский центр ЮУрГУ, 2013. – С. 617.

17. Свирихин Д.И. Определение ресурса параллелизма алгоритма и его эффективного использования для конечного числа процессоров. // Научный сервис в сети Интернет: поиск новых решений: Труды Международной суперкомпьютерной конференции (17 – 22 сентября 2012 г., г. Новороссийск). – М.: Изд-во МГУ, 2012. – С. 257-260.

18. Свирихин Д.И. Программная система для определения ресурса параллелизма метода релаксации для решения СЛАУ: Вып. квалиф. работа бакалавра информационных технологий: 010400.62 / Южно-Уральский государственный университет. – Челябинск, 2011. – 37 л.

19. Система V-RAY. // [Лаборатория Параллельных информационных технологий НИВЦ МГУ]. [Электронный ресурс] URL: http://v-ray.parallel.ru (дата обращения: 02.04.2017).

20. Троелсен Э. Язык программирования C# 2010 и платформа .NET 4, 5-е изд. – М.: Вильямс, 2011. – 1392 с.

21. Фаулер М. UML. Основы. 3-е издание. – СПб: Символ-Плюс, 2004. – 192 с.

22. Шилдт Г. C# 4.0: Полное руководство. – М.: Вильямс, 2011. – 1056 с.