Здесь будет титульник, листай ниже

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Язык программирования C++ - компилируемый язык общего назначения, наследник языка C, предоставляющий инструменты для реализации объектно-ориентированного программирования - современной парадигмы разработки, подразумевающей возможность создания совокупности (системы) взаимосвязанных и взаимозависимых объектов данных, определённых пользователем.

В рамках изучения курса объектно-ориентированного программирования требовалось выполнение курсовой работы на основе задания из системы "Аврора" по выбранной теме с целью освоения программы по вышеназванному предмету, получению навыков разработки программ на языке C++ и решению задач, содержащихся непосредственно в самой формулировке задания. Было выбрано задание уровня "К*" с темой К7 "Моделирование движения по лабиринту". До его отправки были представлены дополнительные базовые работы: 4.1.1, КЛ 3.1, КЛ 3.2, КЛ 3.3. В данных работах была последовательно заложена структурная и принципиальная основа курсовой работы.

Суть задания сводилась к разработке системы, определяющей наличие входов и выходов, расположенных в детерминированных позициях на поле. Сущность алгоритма решения, описанного ниже, заключается в поиске входов на внешнем контуре заданного поля и соответствующим им выходов при их наличии путём распространения маркирующего значения по тактам с учётом достижимости рассматриваемой позиции на поле из других достижимых позиций.

1 ПОСТАНОВКА ЗАДАЧИ

Выполнить моделирование работы системы следующей конструкции. Система содержит поле в виде квадратной матрицы клеток (позиций) 22 х 22. Каждая клетка поля либо заполнена, либо пустая. Координаты клетки меняются от 1 до 22 по горизонтали и по вертикали. Клетки в первой и последней строке, в первом и последнем столбце пустые (внешний контур). С помощью пустых клеток в квадрате с координатами от 2 и 21 по горизонтали и по вертикали заданы схемы лабиринтов. Имеется пульт управления и робот. Робот может:

Двигаться по сигналу (команде) от пульта вверх, вниз, влево и право на одну клетку, если она пустая;

Взаимодействовать с полем посредством сигнала и тот в ответ выдает сигнал с информацией о наличии пустых клеток относительно текущего положения робота;

Передать пульту управления сигнал необходимой согласно алгоритму информацией.

Перед запуском системы задается схема лабиринтов. Исходное положение робота координата (1, 2). Для поиска входа в лабиринт робот движется по часовой стрелке по пустым клеткам внешнего контура. Вход всегда изолирован, т.е. слева и справа, или сверху и снизу всегда присутствуют непустые клетки (стоят 1). Система ищет лабиринт и определяет, есть ли у лабиринта выход, отличный от входа или нет. Такой лабиринт может не существовать. На поле лабиринтов может быть несколько. Система завершает работу после обнаружения лабиринта с различными входом и выходом. Система функционирует по тактам, в рамках одного такта выполняется одно действие.

Построить систему, которая использует объекты:

1. Объект «система». Объект организует основной цикл тактов функционирования;

2. Объект для чтения данных. Считывает данные о схеме лабиринтов. После чтения очередной порции данных, объект выдает сигнал с текстом полученных данных. Все данные синтаксически корректны;
3. Объект пульта управления, для проведения анализа состояния системы и выдачи очередной команды посредством сигнала;
4. Объект, моделирующий поле размещения лабиринтов. Реагирует на получение сигнала с текущими координатами робота и сигналом возвращает информацию о соседних клетках. Пустой клетке соответствует значение 0, не пустой 1;
5. Объект, моделирующий устройство робота. Выполняет шаг по полю, принимает сигналы (команды) и делает запрос по сигналу объекту поле;
6. Объект для вывода результата отработки системы. Выводит информацию о найденных лабиринтах.

Все приведенные сигналы и соответствующие обработчики должны быть реализованы. Все сообщения на консоль выводятся с новой строки. В набор поддерживаемых команд добавить команду «SHOWTREE» и по этой команде вывести дерево иерархии объектов системы с отметкой о готовности и завершить работу системы (программы).

1.1 Описание входных данных

Первые 22 строки содержат схему расположения лабиринтов, которая задается посредством 0 и 1.

Пример ввода:

00000000000000000000

01011101111011111111110

01000111000000001111110

01011111111111111100000
01000111111101100001110
01111111111000011111110
01111111111111111111110
01111111111111111111110
01111111111111111111110
01110111111111111111110
01110111111111111110110
00000000001111111110110
01101111101111111110110
01101100000110000000000
01111111110000111110110
01111111111111110000110
01010110111111110111110
01010110111111110111110
00000000111111111111110
01111111111111111111110
01111111111111111111110
00000000000000000000000

1.2 Описание выходных данных

При обнаружении тупикового лабиринта выводиться координата входа по форме:

There is no way out of the maze («номер строки», «номер столбца»)

При обнаружении искомого лабиринта выводятся координаты входа и выхода по форме:

Maze («номер строки», «номер столбца») («номер строки», «номер столбца»)

Пример вывода

There is no way out of the maze (1, 3)

There is no way out of the maze (1, 7)

There is no way out of the maze (1, 12)

There is no way out of the maze (4, 22)

Maze (14, 22) (12, 1)

2 МЕТОД РЕШЕНИЯ

Для достижения поставленных целей используется основа, созданная и описанная при выполнении задач типа КЛ_*.

Помимо этого, также использованы и другие инструменты:

Структура пользовательского типа (struct) position - представление слоя исходной карты, карты выходов и карты машрутов, её поля и функционал.

Параметризованная директива препроцессора USICAST для взятия адреса объекта, а также - директивы препроцессора #define и #include.

Директива typedef для объявления нового типа.

Пользовательский тип SIG_TEXT - указатель на метод отправки сигнала сообщения с текстом (void(dummy_two::* SIG_TEXT)(string msg);).

Пользовательский тип HAN_TEXT - указатель на метод обработчика сигнала сообщения с текстом (typedef void(base::* HAN_TEXT)(string msg);).

Пользвовательский тип SIG_MOVE - указатель на метод отправки сигнала с командой движения (typedef bool(dummy_three::* SIG_MOVE)(int pos);).

Пользовательский тип HAN_MOVE - указатель на метод обработки сигнала - команды движения(typedef bool(dummy_five::* HAN_MOVE)(int pos);).

Пользвотельский тип SIG_LOC_HAN - указатель на метод отправки сигнал-запроса с данными о позиции (подобный обработчику возвращаемого сигнала) (typedef vector<bool>(dummy_five::*SIG_LOC_HAN)(int x, int y, bool noreverse);).

Пользовательский тип HAN_LOC_SIG - указатель на метод отправки ответного сигнала на запрос данных об окружении и обработчик сигнала-запроса с данными позиции (typedef vector<bool>(dummy_four::*HAN_LOC_SIG)(int x, int y, bool noreverse);).

Пользовательский тип обработчика ответа - HAN_LOC. Обрабатывает сигналы от поля к роботу. Указатель на метод, см. файл base.h.

Объекты пользовательских типов base (базовый элемент), cl_application (объект "система"), dummy_two (объект для чтения данных) , dummy_three (объект пульта управления), dummy_four (объект, моделирующий поле размещения лабиринтов), dummy_five (объект, моделирующий устройство робота), dummy_six (объект для вывода результатов отработки системы), их методы и свойства.

Оператор "ИЛИ" ||.

Оператор взятия адреса &.

Оператор взятия остатка % (для реализации кольца вычетов).

Оператор указателя на член (со "звёздочкой") ->*.

Иные программные конструкции, элементы синтаксиса языка или директивы препроцессора, обеспечивающие необходимый функционал для компиляции, записи алгоритма или оформления программы.

Описание изменений в основе.

Работа была модифицирована, переработана и дополнена в соответствии с поставленной задачей. Исправлены консрукторы объектов.

Примечание: для описанный ниже типов данных не описываются отдельно наследованные, заданные "по умолчанию" или уже описанные характеристики.

Структура position:

7. Свойства/поля:

- Поле значения информационного уровня схемы лабиринта в данной точке (1 - стена, 0 - проход).

1. Тип: int - целое число;
2. Наименование: i;
3. Модификатор доступа: по-умолчанию для структуры (public/открытый).

- Поле значения уровня карты входов лабиринта в данной точке (0 - отсутстующий или потенциальный,

1 - вертикальный сверху, 2 - вертикальный снизу, 3 - горизонтальный справа, 4 - горизонтальный слева).

1. Тип: int - целое число;
2. Наименование: t;
3. Модификатор доступа: по-умолчанию для структуры (public/открытый).
    - Поле значения уровня локальной схемы маршрутов лабиринта в данной точке (1 - в данный момент доступ в точку возможен либо она является отправной, 0 - в данный момент доступ в точку невозможен либо является избыточным).

Тип: int - целое число;

Наименование: p;

Модификатор доступа: по-умолчанию для структуры (public/открытый).

Класс reader_object (фигурировал ранее и используется с помощью директив препроцессора под именем dummy_two):

Функционал:

Метод чтения всех порций данных, передаваемых конструируемому полю лабиринтов по адресу &field - void elementalReader(vector<vector<position>> &field).

Метод отправки текстового сигнала со строкой pofd - sendSignal(string pofd).

Метод получения обработчика текстового сигнала типа HAN_TEXT getTextSigHan().

Метод получения отправителя текстового сигнала типа SIG_TEXT getTextSigSen().

Метод void setConnect(SIG_TEXT source, base* target, HAN_TEXT handler) - создание связи метода handler объекта target с методом source.

Метод void deleteConnect(SIG_TEXT source, base* target, HAN_TEXT handler) - разрушение связи метода handler объекта target с методом source.

Класс control_object (фигурировал ранее и используется с помощью директив препроцессора под именем dummy_three):

Свойства/поля:

Статическое поле текущего направления (все направления движения в лабиринте последовательно перпендикулярны) с номером от нуля до трёх.

Тип: int - целое число (начальное значение - 0);

Наименование: direction;

Модификатор доступа: public (public *static*);

Функционал:

Метод moveNextInStaticField() - метод отправления сообщения роботу о следующем корректном передвижении.

Метод HAN_MOVE getMoveSigHan() - метод получения обработчика сигнала-команды движения.

Метод SIG_MOVE getMoveSigSen() - метод получения отправителя сигнала-команды движения.

Метод void sendSignal(int msg) - метод отправки сообщения msg типа int (целое - код направления движения) роботу.

Метод void setConnect(SIG_MOVE source, base* target, HAN_MOVE handler) - создание связи метода handler объекта target с методом source.

Метод void deleteConnect(SIG_MOVE source, base* target, HAN_MOVE handler) - разрушение связи метода handler объекта target с методом source.

Класс field_object (начало; фигурировал ранее и используется с помощью директив препроцессора под именем dummy_four):

Функционал:

Метод получения вектора значений соседних клеток (один - стена, ноль - пропуск)

в особом порядке, int x и int y - координаты текущей клетки, bool noreverse (следует ли не инвертировать значения в векторе) - vector<int> possibles(int x, int y, bool noreverse = false).

Класс robot_object (фигурировал ранее и используется с помощью директив препроцессора под именем dummy_five):

Свойства/поля:

Поле координаты робота по оси OX.

Наименование: robx (начальное значение - 0);

Тип: int (целое число).

Модификатор доступа: public.

Поле координаты робота по оси OY.

Наименование: roby (начальное значение - 0);

Тип: int (целое число).

Модификатор доступа: public.

Поле последнего ответа на сигнальный запрос:

Наименование: last;

Тип: булев вектор;

Модификатор доступа: public;

Функционал:

Метод bool moveInStaticField(int pos) - осуществление движения робота в направлении с кодом pos целого типа и возвращение логического признака выполненности действия.

Метод HAN_LOC_SIG getLocSigHan() - получение приёмщика сигнала с запросом данных о соседних клетка и отправителя ответного сигнала.

Метод SIG_LOC_HAN getLocSigSen() - получение отправителя сигнала с запросом данных о соседних клектах и приёмщика ответного сигнала.

Метод vector<bool> sendSignal(int x, int y, bool noreverse = false) - функция(метод)

отправки сигнала полю с запросом данных о соседних клетках для позици (x, y) и принятия ответного сигнала с возвращением результата в виде вектора кодов позиций.

Метод void setConnect(SIG_LOC_HAN source, base* target, HAN_LOC_SIG handler) - создание связи метода handler объекта target с методом source.

Метод void deleteConnect(SIG_LOC_HAN source, base* target, HAN_LOC_SIG handler) - разрушение связи метода handler объекта target с методом source.

Метод void handler(vector<bool> response) - обработка ответа response на сигнальный запрос.

Класс printer_object (фигурировал ранее и используется с помощью директив препроцессора под именем dummy_six):

Свойства/поля:

Поле вектора данных для печати в консоль.

Наименование: data;

Тип: vector<string>;

Модификатор доступа: public;

Функционал:

Метод void unhold() - вывод всех накопленных сообщений.

Метод void hold(string msg) - добавление строки msg в конец списка накопленных данных.

Класс cl_application ("система"):

Свойства/поля:

Поле объекта системы (статическое).

Наименование: system;

Тип: cl_application*;

Модификатор доступа: public (public *static*).

Поле объекта чтения данных (статическое).

Наименование: reader;

Тип:dummy_two*;

Модификатор доступа: public (public *static*).

Поле объекта-поля (статическое).

Наименование: square;

Тип: dummy_four*;

Модификатор доступа: public (public *static*).

Поле объекта-пульта (статическое).

Наименование: control;

Тип:dummy_three*;

Модификатор доступа: public (public *static*).

Поле объекта-робота (статическое).

Наименование: robot;

Тип: dummy_five*;

Модификатор доступа: public (public *static*).

Поле объекта вывода данных (статическое).

Наименование: writer;

Тип: dummy_six*;

Модификатор дсотупа: public (public *static*).

Поле всех лабиринтов (статическое):

Наименование: locs;

Тип: vector<vector<position>>;

Модификатор доступа: public (public *static*).

Функционал:

Метод void bild_tree_objects() - метод организации основного цикла тактов функционирования, подразумевающий также построение дерева и иных структур, а также выполнение поставленной задачи.

Метод int exec_app() - метод получения признака корректности завершения работы системы.

Класс field_object (продолжение; методы до установок сигналов ранее находились в базовом классе и были перенесены по указанию преподавателя):

Функционал:

Метод int isExit(vector<vecor<position>> & field, int x, int y) - метод получения кода клетки на схеме входов и выходов (0 - отсутствующий или потенциальный, 1 - вертикальный сверху, 2 - вертикальный снизу, 3 - горизонтальный справа, 4 - горизонтальный слева) в точке (x, y) поля field.

Метод void floodInductive(vector<vector<position>> & field, int x, int y) - метод "просачивания" единиц из схемы маршрутов поля field в данную клетку (x, y) , составная часть индуктивного динамического итеративного определения признака достижимости позиции.

Метод void meltFlood(vector<vector<position>> & field) - метод "наводнения" единицами - индуквтивное распространение единиц в рамках локальной карты маршрутов.

Метод void demolishAll(vector<vector<position>> & field) - метод "очищения" локальной (текущей) карты маршрутов поля field путём её отката до состояния нулевой матрицы.

Метод prepareAll(vector<vector<position>> & field) - метод корректной инициализации схемы лабиринтов field, её подготовка.

Метод setupTypes(vector<vector<position>> & field) - метод установки меток с кодами типов входов и выходов на карте field.

vector<int> getExitCoordsCorrected(vector<vector<position>> & field, int x, int y) - метод получения координаты входа-выхода, располагающейся на внешнем контуре поля field исходя из исходных координат точки входа-выхода (x,y).

vector<int> getTsunami(vector<vector<position>> & field, int x, int y) - метод оценки

наличия входа-выхода в точке (x, y) поля field с получением координат точки, от которой можно начать каскадное итеративное заполнение локальной карты маршрутов.

vector<int> isLabyrinth(vector<vector<position>> & field, int xp, int yp) - метод проверки на существование в поле field лабиринта со входом в точке (xp, yp) и получением вектора последовательных координат входа и отличного от него выхода из лабиринта.

Метод void setConnect(HAN_LOC_SIG source, base* target, HAN_LOC handler) - создание связи метода handler объекта target с методом source.

Метод void deleteConnect(HAN_LOC_SIG source, base* target, HAN_LOC handler) - разрушение связи метода handler объекта target с методом source.

Структура bridge:

Свойства/поля:

Поле идентификатора связи:

Тип: int;

Наименование: bridge_type;

Модификатор доступа: нет (открытый член по умолчанию);

Поле обработчика связи:

Тип: int;

Наименование: responder;

Модификатор доступа: нет (открытый член по умолчанию);

Поле текстового сигнала:

Поле текстового обработчика:

Тип: HAN_TEXT;

Наименование: han_t;

Модификатор доступа: нет (открытый член по умолчанию);

Поле обработчика движения:

Тип: HAN_MOVE;

Наименование: han_m;

Модификатор доступа: нет (открытый член по умолчанию);

Поле сигнала движения:

Тип: SIG_MOVE;

Модификатор доступа: нет (открытый член по умолчанию);

Наименование: sig_m;

Поле запроса о положении:

Тип: SIG_LOC_HAN;

Модификатор доступа: нет (открытый член по умолчанию);

Наименование: sig_ld;

Поле ответа (обработчика) ответа обработчика на запрос о положении:

Тип: HAN_LOC;

Модификатор доступа: нет (открытый член по умолчанию);

Наименование: han_loc;

Поле обработчика запроса о положении с сигналом ответа:

Тип: HAN_LOC_SIG;

Модификатор доступа: нет (открытый член по умолчанию);

Наименование: han_ld;

Функционал:

bool operator==(const bridge& coll) - метод-аналог подобного для структуры pairz из КЛ 3.3, проверка равенства по всем членам структуре coll.

В базовом классе base метод обработчика сигнала переделан в метод void specComHan(string msg) - метод обработки сигнала с текстовым сообщением msg, выполняющий анализ команды msg. В базовом классе член списка связей

vector<pairz> relations заменён на vector<bridge> rels с сохранением смысла контейнера-члена как хранилища связей объекта.

Некоторые методы возвращают особые значения при "некорректном" использовании в проблемных случаях. Также часть методов является статической, чтобы существовала возможность их вызова без самого объекта.

Таблица наследования классов не менялась.

Стоит отметить, что программа использует свойства ссылок. Также иногда в описаниях опускаются вложенные циклы и иные элементы - путём эквивалентной замены. Содержание выражений опирается на язык C++. Под "да" подразумевается истина, под "нет" - ложь. Важно, что в данной системе все объекты являются активными и не меняют своего состояния готовности. В связи с этим условие проверки готовности объекта является избыточным и может подаруземаваться, однако не оказывает влияния на функционирование программы и опущено во всех алгоритмах.

3 ОПИСАНИЕ АЛГОРИТМОВ

Согласно этапам разработки, после определения необходимого инструментария в разделе «Метод», составляются подробные описания алгоритмов для методов классов и функций.

3.1 Алгоритм метода elementalReader класса reader_object

Функционал: Метод чтения всех порций данных, передаваемых конструируемому полю лабиринтов по ссылке &field..

Параметры: vector<vector<position>> & field - массив позиций на поле лабиринтов..

Возвращаемое значение: void - отсутствует..

Алгоритм метода представлен в таблице 1.

Таблица 1 – Алгоритм метода elementalReader класса reader_object

№	Предикат	Действия	№ перехода
1		Объявление целочисленной переменной xrow типа int с инициализацией значением нуля.	2
2		Объявление строки dataline типа string.	3
3	Предикат 1 - xrow меньше 22.	Считывание из cin в dataline.	4
		Заполнение карты выходов вызовом: field_object::setupTypes(field).	∅
4		Вызов сигнала по указателю: (this->*getTextSigSen())(dataline);	5
5	Предикат 2 - dataline не равно "SHOWTREE".	Добавление всех цифр из строки в качестве позиций на схему лабиринтов field.	6
			3

№	Предикат	Действия	№ перехода
6		Инкрементирование xrow.	3

3.2 Алгоритм метода getTextSigHan класса reader_object

Функционал: Метод получения обработчика текстового сигнала типа HAN_TEXT..

Параметры: void - отсутствуют..

Возвращаемое значение: HAN_TEXT - обработчик текстового сигнала..

Алгоритм метода представлен в таблице 2.

Таблица 2 – Алгоритм метода getTextSigHan класса reader_object

№	Предикат	Действия	№ перехода
1		Возвращение значения выражения USICAST(base::specComHan).	∅

3.3 Алгоритм метода getTextSigSen класса reader_object

Функционал: Метод получения отправителя текстового сигнала типа SIG_TEXT..

Параметры: void - отсутствуют..

Возвращаемое значение: SIG_TEXT - отправитель текстового сигнала..

Алгоритм метода представлен в таблице 3.

Таблица 3 – Алгоритм метода getTextSigSen класса reader_object

№	Предикат	Действия	№ перехода
1		Возвращение значения выражения: USICAST(dummy_two::sendSignal).	∅

3.4 Алгоритм метода sendSignal класса reader_object

Функционал: Метод отправки текстового сигнала со строкой pofd..

Параметры: string pofd - строка входных данных сигнала..

Возвращаемое значение: void - отсутствует..

Алгоритм метода представлен в таблице 4.

Таблица 4 – Алгоритм метода sendSignal класса reader_object

№	Предикат	Действия	№ перехода
1	Предикат 1 - Остались нерассмотренные сигнальные связи данного объекта.	Отправка сигнала вызовом метода обработчика объекта-слушателя очередной связи при корректности связи с сигнальным методом в очередной связи.	1
			∅

3.5 Алгоритм метода moveNextInStaticField класса control_object

Функционал: Метод отправления сообщения роботу о следующем корректном передвижении..

Параметры: void - отсутствуют..

Возвращаемое значение: void - отсутствует..

Алгоритм метода представлен в таблице 5.

Таблица 5 – Алгоритм метода moveNextInStaticField класса control_object

№	Предикат	Действия	№ перехода
1		Объявление логической переменной moved типа bool с инициализацией ложью.	2
2		Объявление переменной method типа SIG_MOVE с инициализацией значением врыжания getMoveSigSen().	3
3	Предикат 1 - this->direction равняется нулю.	Присвоение moved значения выражения (this->*method)(2).	4

№	Предикат	Действия	№ перехода
	Предикат 2 - this->direction равняется единице.	Присвоение moved значения выражения (this->*method)(0).	4
	Предикат 3 - this->direction равняется двум.	Присвоение moved значения выражения (this->*method)(3).	4
	Предикат 4 - this->direction равняется трём.	Присвоение moved значения выражения (this->*method)(1).	4
			4
4	Предикат 5 - Отрицание moved - истина.	Присвовение direction значения (direction + 1) % 4 (следующего в кольце вычетов по модулю 4).	∅
			∅

3.6 Алгоритм метода getMoveSigHan класса control_object

Функционал: Метод получения обработчика сигнала-команды движения..

Параметры: void - отсутствуют..

Возвращаемое значение: HAN_MOVE - обработчик сигнала движения..

Алгоритм метода представлен в таблице 6.

Таблица 6 – Алгоритм метода getMoveSigHan класса control_object

№	Предикат	Действия	№ перехода
1		Возвращение значения выражения USICAST(dummy_five::moveInStaticField).	∅

3.7 Алгоритм метода getMoveSigSen класса control_object

Функционал: Метод получения отправителя сигнала-команды движения..

Параметры: void - отсутствуют..

Возвращаемое значение: SIG_MOVE - сигнальный объект команды движения (здесь, выше и далее: сущностью введённых аналогичных

пользовательских типов сигналов и обработчиков является факт того, что они - указатели на методы-члены соответствующих классов)..

Алгоритм метода представлен в таблице 7.

Таблица 7 – Алгоритм метода getMoveSigSen класса control_object

№	Предикат	Действия	№ перехода
1		Возвращение значения выражения USICAST(dummy_three::sendSignal).	∅

3.8 Алгоритм метода sendSignal класса control_object

Функционал: Метод отправки сообщения msg типа int (целое - код направления движения) роботу..

Параметры: int msg - направление движения..

Возвращаемое значение: bool - логический признак успешности совершения требуемого манёвра..

Алгоритм метода представлен в таблице 8.

Таблица 8 – Алгоритм метода sendSignal класса control_object

№	Предикат	Действия	№ перехода
1	Предикат 1 - Имеются нерассмотренные сигнальные связи данного объекта.	Отправка сигнала вызовом метода обработчика объекта-слушателя очередной связи при корректности связи с сигнальным методом в очередной связи (с передачей сообщения в переданных аргументах и кешированием ответа).	1
			2
2		Возвращение значения, возвращённого последней отправкой оообщения.	∅

3.9 Алгоритм метода possibles класса field_object

Функционал: Метод получения вектора значений соседних клеток (один - стена, ноль - пропуск) в особом порядке, int x и int y - координаты текущей клетки, bool noreverse (следует ли не инвертировать значения в векторе)..

Параметры: int x - координата по x, int y - координата по y, bool noreverse - не стоит ли инвертировать выходные данные поэлементо..

Возвращаемое значение: vector<bool> - коды доступнсоти соседних позиций..

Алгоритм метода представлен в таблице 9.

Таблица 9 – Алгоритм метода possibles класса field_object

№	Предикат	Действия	№ перехода
1	Предикат 1 - locs[x][y].i равняется единице.	Возвращение {1,1,1,1}, если noreverse - истина, иначе - возвращение {0,0,0,0} (с дублированием возвращения сигналом всем слушателям).	∅
		Объявление переменных one, two, three, four логического типа bool.	2
2	Предикат 2 - x равняется 21.	Присвоить one значение 0.	3
		Присвоить one значение (locs[x+1][y].i == 0).	3
3	Предикат 3 - x равняется 0.	Присвоить two значение 0.	4
		Присвоить two значение (locs[x-1][y].i == 0).	4
4	Предикат 4 - y равняется 21.	Присвоить three значение 0.	5
		Присвоить three значение (locs[x][y+1].i == 0).	5
5	Предикат 5 - y равнеятся 0.	Присвоить four значение 0.	6
		Присвоить four значение (locs[x][y-1].i == 0).	6
6	Предикат 6 - noreverse - ИСТИНА.	Возвращение {!one, !two, !three, !four} с дублированием возвращения сигналом всем слушателям.	∅
		Возвращение {one, two, three, four} с	∅

№	Предикат	Действия	№ перехода
		дублированием возвращения сигналом всем слушателям.	

3.10 Алгоритм метода moveInStaticField класса robot_object

Функционал: Осуществление движения робота в направлении с кодом pos целого типа и возвращение логического признака выполненности действия..

Параметры: int pos - код направления движения..

Возвращаемое значение: bool - логический признак корректности манёвра..

Алгоритм метода представлен в таблице 10.

Таблица 10 – Алгоритм метода moveInStaticField класса robot_object

№	Предикат	Действия	№ перехода
1		Объявление переменной method типа SIG_LOC_HAN с инициализацией значением this->getLocSigSen().	2
2		Объявление логического вектора possible типа vector<bool> с инициализацией значением выражения (this->*method)(robx, roby, false).	3
3	Предикат 1 - possible[pos] - не логическая ложь.		4
		Возвращение лжи.	∅
4	Предикат 2 - pos это 0.	Инкрементирование: this->robx++.	5
	Предикат 3 - pos это 1.	Декрементирование:this->robx--.	5
	Предикат 4 - pos это 2.	Инкрементирование:this->roby++.	5
	Предикат 5 - pos это 3.	Декрементирование: this->roby--.	5
			5
5		Возвращение истины.	∅

3.11 Алгоритм метода getLocSigHan класса robot_object

Функционал: Получение приёмщика сигнала с запросом данных о соседних клетка и отправителя ответного сигнала..

Параметры: void - отсутствуют..

Возвращаемое значение: HAN_LOC_SIG - обработчик запроса данных о соседях и отправитель ответа..

Алгоритм метода представлен в таблице 11.

Таблица 11 – Алгоритм метода getLocSigHan класса robot_object

№	Предикат	Действия	№ перехода
1		Возвращение значения USICAST(dummy_four::possibles).	∅

3.12 Алгоритм метода getLocSigSen класса robot_object

Функционал: Получение отправителя сигнала с запросом данных о соседних клектах и приёмщика ответного сигнала..

Параметры: void - отсутствуют..

Возвращаемое значение: SIG_LOC_HAN - отправитель запроса данных о соседях и обработчик ответа..

Алгоритм метода представлен в таблице 12.

Таблица 12 – Алгоритм метода getLocSigSen класса robot_object

№	Предикат	Действия	№ перехода
1		Возвращение значения USICAST(dummy_five::sendSignal).	∅

3.13 Алгоритм метода sendSignal класса robot_object

Функционал: Функция(метод) отправки сигнала полю с запросом данных о соседних клетках для позици (x, y) и принятия ответного сигнала с возвращением

результата в виде вектора кодов позиций..

Параметры: int x - целая координата по OX точки, int y - целая координата по OY точки, bool noreverse - не инвертировать ли ответ.

Возвращаемое значение: vector<bool> - булев вектор ответа на запрос данных о свободных клетках окрестности..

Алгоритм метода представлен в таблице 13.

Таблица 13 – Алгоритм метода sendSignal класса robot_object

№	Предикат	Действия	№ перехода
1	Предикат 1 - Имеются нерассмотренные сигнальные связи данного объекта.	Отправка сигнала вызовом метода обработчика объекта-слушателя очередной связи при корректности связи с сигнальным методом в очередной связи (с передачей сообщения в переданных аргументах).	1
			2
2		Возвращение значения this->last (сохранённое значение последнего сигнального ответа).	∅

3.14 Алгоритм метода unhold класса printer_object

Функционал: Вывод всех накопленных сообщений..

Параметры: void - отсутствуют..

Возвращаемое значение: void - отсутствует..

Алгоритм метода представлен в таблице 14.

Таблица 14 – Алгоритм метода unhold класса printer_object

№	Предикат	Действия	№ перехода
1		Объявление булевой переменной first типа bool с инициализацией значением истины.	2
2	Предикат 1 - В data остались	Занести очередную строку из data в переменную	3

№	Предикат	Действия	№ перехода
	нерассмотренные строки.	msg типа string.	
			∅
3	Предкат 2 - first это ИСТИНА.	Присвоить first значение false.	4
		Возврат каретки.	4
4		Вывод msg в консоль.	2

3.15 Алгоритм метода hold класса printer_object

Функционал: Добавление строки msg в конец списка накопленных данных..

Параметры: string msg - строка запоминаемого сообщения..

Возвращаемое значение: void - отсутствует..

Алгоритм метода представлен в таблице 15.

Таблица 15 – Алгоритм метода hold класса printer_object

№	Предикат	Действия	№ перехода
1		Вызов метода добавления данных: this->data.push_back(msg).	∅

3.16 Алгоритм метода bild_tree_objects класса cl_application

Функционал: Метод организации основного цикла тактов функционирования, подразумевающий также построение дерева и иных структур, а также выполнение поставленной задачи..

Параметры: void - отсутствуют..

Возвращаемое значение: void - отсутствует..

Алгоритм метода представлен в таблице 16.

Таблица 16 – Алгоритм метода bild_tree_objects класса cl_application

№	Предикат	Действия	№ перехода
1		Присвоение system значения this.	2
2		Присвоение reader значения указателя на новый объект dummy_two, аргументы конструктора - (NULL, "Reader").	3
3		Установка родителя system для reader.	4
4		Присвоение robot значения указателя на новый объект dummy_five, аргументы конструктора - (NULL, "Robot").	5
5		Установка родителя system для robot.	6
6		Присвоение control значения указателя на новый объект dummy_three, аргументы конструктора - (NULL, "Control").	7
7		Установка родителя system для control.	8
8		Присвоение square значения указателя на новый объект dummy_four, аргументы конструктора - (NULL, "Field").	9
9		Установка родителя system для square.	10
10		Присвоение writer значения указателя на новый объект dummy_six, аргументы конструктора - (NULL, "Writer").	11
11		Установка родителя system для writer.	12
12		Подготовка системы: system->prepareAll(field_object::locs).	13
13		Анализ входных данных, обработка комманд: reader->elementalReader(field_object::locs).	14
14		Объявление вектора check типа vector<int> с инициализацией значением this->isLabyrinth(field_object::locs, robot->robx, robot-	15

№	Предикат	Действия	№ перехода
		>roby).	
15	Предикат 1 - размер check равен 4.	Запоминание информации о лабиринте с различным входом и выходом.	16
			18
16		Показ всех запомненных сообщений.	17
17		Возвращение управления в явном виде.	∅
18	Предикат 2 - check[0] не -1.	Запоминание сообщения о тупиковом лабиринте.	19
			19
19		Вызов: control->moveNextInStaticField().	20
20	Предикат 3 - Не возвращено управление/не произведен выход из цикла.	Объявление вектора check типа vector<int> с инициализацией значением this->isLabyrinth(field_object::locs, robot->robx, robot->roby).	21
			21
21	Предикат 3.5 - Робот вернулся в исходную клетку.	Выход из цикла в явном виде.	27
			22
22	Предикат 4 - размер check равен 4.	Запоминание информации о лабиринте с различным входом и выходом.	23
			25
23		Показ всех запомненных сообщений.	24
24		Возвращение управления в явном виде.	∅
25	Предикат 5 - check[0] не -1.	Запоминание сообщения о тупиковом лабиринте.	26
			26

№	Предикат	Действия	№ перехода
26		Вызов: control->moveNextInStaticField().	20
27		Показ всех запомненных сообщений.	∅

3.17 Алгоритм метода exec_app класса cl_application

Функционал: Метод получения признака корректности завершения работы системы..

Параметры: void - отсутствуют..

Возвращаемое значение: Целое число типа int - индикатор корректности завершения (ожидается 0)..

Алгоритм метода представлен в таблице 17.

Таблица 17 – Алгоритм метода exec_app класса cl_application

№	Предикат	Действия	№ перехода
1		Возвращение нуля.	∅

3.18 Алгоритм метода isExit класса field_object

Функционал: Метод получения кода клетки на схеме входов и выходов (0 - отсутствующий или потенциальный, 1 - вертикальный сверху, 2 - вертикальный снизу, 3 - горизонтальный справа, 4 - горизонтальный слева) в точке (x, y) поля field..

Параметры: vector<vector<position>> &field - поле лабиринтов, int x - координата проверяемой клетки поля по оси OX, , int y - координата клетки поля по оси OY..

Возвращаемое значение: int - целое число, "код выхода"..

Алгоритм метода представлен в таблице 18.

Таблица 18 – Алгоритм метода isExit класса field_object

№	Предикат	Действия	№ перехода
1	Предикат 1 - (x, y) это координаты (0,0).	Возвращение нуля.	∅
	Предикат 2 - (x, y) это координаты (0,21).	Возвращение нуля.	∅
	Предикат 3 - (x, y) это координаты (21,0).	Возвращение нуля.	∅
	Предикат 4 - (x, y) это координаты (21,21).	Возвращение нуля.	∅
	Предикат 5 - (x, y) это координаты стены.	Возвращение нуля.	∅
			2
2		Объявление булевой переменной vertical_up типа bool с инициализацией значением выражения (x -1 == 0) && (y > 1) && (y < 20).	3
3		Объявление булевой переменной vertical_down типа bool с инициализацией значением выражения (x + 1 == 21) && (y > 1) && (y < 20).	4
4		Объявление булевой переменной horizontal_right типа bool с инициализацией значением выражения (y + 1 == 21) && (x > 1) && (x < 20).	5
5		Объявление булевой переменной horizontal_left типа bool с инициализацией значением выражения (y - 1 == 0) && (x > 1) && (x < 20).	6
6	Предикат 6 - vectical_up это логическая единица.	Возвращение числа 4.	∅
	Предикат 7 - vectical_down это логическая единица.	Возвращение числа 3.	∅

№	Предикат	Действия	№ перехода
	Предикат 8 - horizontal_right это логическая единица.	Возвращение числа 2.	∅
	Предикат 9 - horizontal_left это логическая единица.	Возвращение числа 1.	∅
		Возвращение числа 0.	∅

3.19 Алгоритм метода floodInductive класса field_object

Функционал: Метод "просачивания" единиц из схемы маршрутов поля field в данную клетку (x, y) , составная часть индуктивного динамического итеративного определения признака достижимости позиции..

Параметры: vector<vector<position>> & field - поле лабиринтов , int x - координата проверяемой клетки поля по оси OX, , int y - координата клетки поля по оси OY..

Возвращаемое значение: void - отсутствует..

Алгоритм метода представлен в таблице 19.

Таблица 19 – Алгоритм метода floodInductive класса field_object

№	Предикат	Действия	№ перехода
1	Предикат 1 - x равен 0.	Вовращение управления.	∅
	Предикат 2 - x равен 21.	Вовращение управления.	∅
	Предикат 3 - y равен 0.	Вовращение управления.	∅
	Предикат 4 - y равен 21.	Вовращение управления.	∅
	Предикат 5 - (x, y) - стена на схеме.	Вовращение управления.	∅
			2
2		Объявление переменной canSinkAbove логического типа bool с инициализацией значением выражения	3

№	Предикат	Действия	№ перехода
		(field[x][y+1].p == 1).	
3		Объявление переменной canSinkDown логического типа bool с инициализацией значением выражения (field[x][y-1].p == 1).	4
4		Объявление переменной canSinkLeft логического типа bool с инициализацией значением выражения (field[x-1][y].p == 1).	5
5		Объявление переменной canSinkRight логического типа bool с инициализацией значением выражения (field[x+1][y].p == 1).	6
6	Предикат 6 - Минимум одна из переменных canSinkAbove, canSinkDown , canSinkLeft, canSinkRight - истина.	Присвоение field[x][y].p значения 1.	∅
			∅

3.20 Алгоритм метода meltFlood класса field_object

Функционал: Метод "наводнения" единицами - индуквтивное распространение единиц в рамках локальной карты маршрутов..

Параметры: vector<vector<position>> & field - поле лабиринтов.

Возвращаемое значение: void - отсутствует.

Алгоритм метода представлен в таблице 20.

Таблица 20 – Алгоритм метода meltFlood класса field_object

№	Предикат	Действия	№ перехода
1		Объявление целой переменной iteration типа int с инициализацией нулём.	2

№	Предикат	Действия	№ перехода
2		Объявление переменной целоц overcrit типа int с инициализацией значением 22 в кубе.	3
3	Предикат 1 - iteration меньше или равно overcrit.		4
			∅
4	Предикат 2 - Рассмотрены не все позиции с целочисленными координатами x, y на поле field.	Получение координат очередной позиции в x, y.	5
			∅
5		Вызов заполнения для позиции: floodInductive(field, x,y).	6
6		Инкрементиование iteration.	∅

3.21 Алгоритм метода demolishAll класса field_object

Функционал: Метод "очищения" локальной (текущей) карты маршрутов поля field путём её отката до состояния нулевой матрицы..

Параметры: vector<vector<position>> & field - поле лабиринтов.

Возвращаемое значение: void - отсутствует..

Алгоритм метода представлен в таблице 21.

Таблица 21 – Алгоритм метода demolishAll класса field_object

№	Предикат	Действия	№ перехода
1	Предикат 1 - Рассмотрены не все позиции с целочисленными координатами x, y на поле	Получение координат очередной позиции в x, y.	2

№	Предикат	Действия	№ перехода
	field.		
			∅
2		Присваивание field[x][y].p значения ноля.	1

3.22 Алгоритм метода prepareAll класса field_object

Функционал: Метод корректной инициализации схемы лабиринтов field, её подготовка..

Параметры: vector<vector<position>> & field - поле лабиринтов.

Возвращаемое значение: void - отсутствует..

Алгоритм метода представлен в таблице 22.

Таблица 22 – Алгоритм метода prepareAll класса field_object

№	Предикат	Действия	№ перехода
1		Объявление целочисленной переменной x типа int с инициализацией нулём.	2
2	Предикат 1 - x меньше 22.	Объявление целочисленной переменной y типа int с инициализацией нулём. (тестовый, необязательный шаг для данной реализации)	3
			∅
3		Объявление массива ряда позиций по OX vector<position> xrow с инициализацией пустым множеством.	4
4		Вызов метода: field.push_back(xrow).	5
5		Инкрементирование x.	2

3.23 Алгоритм метода setupTypes класса field_object

Функционал: Метод установки меток с кодами типов входов и выходов на

карте field..

Параметры: vector<vector<position>> & field - поле лабиринтов.

Возвращаемое значение: void - отсутствует..

Алгоритм метода представлен в таблице 23.

Таблица 23 – Алгоритм метода setupTypes класса field_object

№	Предикат	Действия	№ перехода
1	Предикат 1 - Рассмотрены не все позиции с целочисленными координатами x, y на поле field.	Получение координат очередной позиции в x, y.	2
			∅
2		Присваивание field[x][y].t значения isExit(field, x, y).	1

3.24 Алгоритм метода getExitCoordsCorrected класса field_object

Функционал: Метод получения координаты входа-выхода, располагающейся на внешнем контуре поля field исходя из исходных координат точки входа-выхода (x,y)..

Параметры: vector<vector<position>> & field - поле лабиринтов , int x - координата проверяемой клетки поля по оси OX, , int y - координата клетки поля по оси OY..

Возвращаемое значение: vector<int> - координаты прохода на внешнем контуре или служебные данные при отсутствии первых..

Алгоритм метода представлен в таблице 24.

Таблица 24 – Алгоритм метода getExitCoordsCorrected класса field_object

№	Предикат	Действия	№ перехода
1	Предикат 1 - Значение field[x][y].t равно 0.	Возвращение массива {-1, -1}.	∅
	Предикат 2 - Значение field[x][y].t равно 1.	Возвращение массива {x+1, y+1+1}.	∅
	Предикат 3 - Значение field[x][y].t равно 2.	Возвращение массива {x+1, y -1 + 1}.	∅
	Предикат 4 - Значение field[x][y].t равно 3.	Возвращение массива {x+1+1, y +1}.	∅
	Предикат 5 - Значение field[x][y].t равно 4.	Возвращение массива {x-1+1, y+1}.	∅
		Возвращение массива {-1}.	∅

3.25 Алгоритм метода getTsunami класса field_object

Функционал: Метод оценки наличия входа-выхода в точке (x, y) поля field с получением координат точки, от которой можно начать каскадное итеративное заполнение локальной карты маршрутов..

Параметры: vector<vector<position>> & field - поле лабиринтов , int x - координата проверяемой клетки поля по оси OX, , int y - координата клетки поля по оси OY..

Возвращаемое значение: vector<int> - координаты входа лабиринта, откуда можно начать каскадное заполнение схемы маршрутов, либо служебные данные при их отсутствии..

Алгоритм метода представлен в таблице 25.

Таблица 25 – Алгоритм метода getTsunami класса field_object

№	Предикат	Действия	№ перехода
1	Предикат 1 - (y равен нулю) И (field[x][y+1]).t больше ноля).	Возвращение массива {x, y+1}.	∅
	Предикат 2 - (y равен 21) И (field[x][y-1]).t больше ноля).	Возвращение массива {x, y -1}.	∅
	Предикат 3 - (x равен 0) И (field[x+1][y]).t больше ноля).	Возвращение массива {x+1, y}.	∅
	Предикат 4 - (x равен 21) И (field[x-1][y]).t больше ноля).	Возвращение массива {x-1,y}.	∅
		Возвращение массива {-1}.	∅

3.26 Алгоритм метода isLabyrinth класса field_object

Функционал: Метод проверки на существование в поле field лабиринта со входом в точке (xp, yp) и получением вектора последовательных координат входа и отличного от него выхода из лабиринта..

Параметры: vector<vector<position>> & field - поле лабиринтов , int xp - координата проверяемой клетки поля по оси OX, , int yp - координата клетки поля по оси OY..

Возвращаемое значение: vector<int> - данные о входе лабиринта, в соответствующих случаях дополняются данными о выходе либо служебной информацией..

Алгоритм метода представлен в таблице 26.

Таблица 26 – Алгоритм метода isLabyrinth класса field_object

№	Предикат	Действия	№ перехода
1		Объявление массива vector<int> source с инициализацией значением getTsunami(filed, xp, yp).	2
2	Предикат 1 - source[0] не -1.	Присваивание field[source[0]][source[1]].p значения единицы.	3
		Возвращение source.	∅
3		Вызов метода-члена: this->meltFlood(field).	4
4	Предикат 2 - Рассмотрены не все позиции с целочисленными координатами x, y на поле field, являющиеся выходами, смежными с внешним контуром и отмеченные на текущей локальной карте машрутов и не рассмотренными ранее.	Получение координат очередной названной позиции в x, y.	5
		Вызов метода-члена: this->demolishAll(field).	13
5	Предикат 3 - field[x][y].t равно 1.	Вызов метода-члена: this->demolishAll(field).	6
			7
6		Возвращение массива {xp, yp, x - 1, y}.	∅
7	Предикат 4 - field[x][y].t равно 2.	Вызов метода-члена: this->demolishAll(field).	8
			9
8		Возвращение массива {xp, yp, x + 1, y}.	∅
9	Предикат 5 - field[x][y].t равно 3.	Вызов метода-члена: this->demolishAll(field).	10

№	Предикат	Действия	№ перехода
			11
10		Возвращение массива {xp, yp, x, y + 1}.	∅
11	Предикат 6 - field[x][y].t равно 4.	Вызов метода-члена: this->demolishAll(field).	12
			4
12		Возвращение массива {xp, yp, x, y - 1}.	∅
13		Возвращение массива {xp, yp}.	∅

3.27 Алгоритм метода specComHan класса base

Функционал: Метод обработки сигнала с текстовым сообщением msg..

Параметры: string msg - сообщение сигнала..

Возвращаемое значение: void - отсутствует..

Алгоритм метода представлен в таблице 27.

Таблица 27 – Алгоритм метода specComHan класса base

№	Предикат	Действия	№ перехода
1	Предикат 1 - msg равно "SHOWTREE".	Показ дерева: this->prettyPrintAdvanced(0);	2
			∅
2		Возврат каретки.	3
3		Возвращение управления.	∅

3.28 Алгоритм метода setConnect класса reader_object

Функционал: Метод установки связи с заданными параметрами..

Параметры: SIG_TEXT source - сигнал, base* target - слушатель, HAN_TEXT handler - обработчик.

Возвращаемое значение: void (отсутствует).

Алгоритм метода представлен в таблице 28.

Таблица 28 – Алгоритм метода setConnect класса reader_object

№	Предикат	Действия	№ перехода
1		Объявление новой связи типа bridge.	2
2		Настройка параметров связи в соответствии с входными данными.	3
3		Добавление в массив связей (rels) данного объекта новой связи.	∅

3.29 Алгоритм метода deleteConnect класса reader_object

Функционал: Метод установки связи с заданными параметрами..

Параметры: SIG_TEXT source - сигнал, base* target - слушатель, HAN_TEXT handler - обработчик.

Возвращаемое значение: void (отсутствует).

Алгоритм метода представлен в таблице 29.

Таблица 29 – Алгоритм метода deleteConnect класса reader_object

№	Предикат	Действия	№ перехода
1		Объявление эталонной связи типа bridge.	2
2		Настройка параметров эталонной связи в соответствии с входными данными.	3
3		Удаление элемента из массива связей (rels) данного объекта новой связи, равного эталонному.	∅

3.30 Алгоритм метода setConnect класса robot_object

Функционал: Метод установки связи с заданными параметрами..

Параметры: SIG_LOC_HAN source - сигнал, base* target , HAN_LOC_SIG handler - слушатель.

Возвращаемое значение: void (отсутствует).

Алгоритм метода представлен в таблице 30.

Таблица 30 – Алгоритм метода setConnect класса robot_object

№	Предикат	Действия	№ перехода
1		Объявление новой связи типа bridge.	2
2		Настройка параметров связи в соответствии с входными данными.	3
3		Добавление в массив связей (rels) данного объекта новой связи.	∅

3.31 Алгоритм метода deleteConnect класса robot_object

Функционал: Метод удаления связи с заданными параметрами..

Параметры: SIG_LOC_HAN source - сигнал, base* target , HAN_LOC_SIG handler - слушатель.

Возвращаемое значение: void (отсутствует).

Алгоритм метода представлен в таблице 31.

Таблица 31 – Алгоритм метода deleteConnect класса robot_object

№	Предикат	Действия	№ перехода
1		Объявление эталонной связи типа bridge.	3
2		Настройка параметров эталонной связи в соответствии с входными данными.	2
3		Удаление элемента из массива связей (rels) данного объекта новой связи, равного эталонному.	∅

3.32 Алгоритм метода handler класса robot_object

Функционал: Сохранение ответа на запрос о соседних клетках..

Параметры: vector<bool> response - булев вектор ответа на запрос о соседних клетках..

Возвращаемое значение: void (отсутствует).

Алгоритм метода представлен в таблице 32.

Таблица 32 – Алгоритм метода handler класса robot_object

№	Предикат	Действия	№ перехода
1		Сохранение переданного аргумента в переменной last текущего объекта.	∅

3.33 Алгоритм метода setConnect класса control_object

Функционал: Метод установки связи с заданными параметрами..

Параметры: SIG_MOVE source - сигнал, base* target - слушатель, HAN_MOVE handler - обработчик.

Возвращаемое значение: void (отсутствует).

Алгоритм метода представлен в таблице 33.

Таблица 33 – Алгоритм метода setConnect класса control_object

№	Предикат	Действия	№ перехода
1		Объявление новой связи типа bridge.	2
2		Настройка параметров связи в соответствии с входными данными.	3
3		Добавление в массив связей (rels) данного объекта новой связи.	∅

3.34 Алгоритм метода deleteConnect класса control_object

Функционал: Метод удаления связи с заданными параметрами..

Параметры: SIG_MOVE source - сигнал, base* target - слушатель, HAN_MOVE handler - обработчик.

Возвращаемое значение: void (отсутствует).

Алгоритм метода представлен в таблице 34.

Таблица 34 – Алгоритм метода deleteConnect класса control_object

№	Предикат	Действия	№ перехода
1		Объявление эталонной связи типа bridge.	2
2		Настройка параметров эталонной связи в соответствии с входными данными.	3
3		Удаление элемента из массива связей (rels) данного объекта новой связи, равного эталонному.	∅

3.35 Алгоритм метода setConnect класса field_object

Функционал: Метод установки связи с заданными параметрами..

Параметры: HAN_LOC_SIG source - сигнал, base* target - слушатель, HAN_LOC handler - обработчик.

Возвращаемое значение: void (отсутствует).

Алгоритм метода представлен в таблице 35.

Таблица 35 – Алгоритм метода setConnect класса field_object

№	Предикат	Действия	№ перехода
1		Объявление новой связи типа bridge.	2
2		Настройка параметров связи в соответствии с входными данными.	3
3		Добавление в массив связей (rels) данного объекта новой связи.	∅

3.36 Алгоритм метода deleteConnect класса field_object

Функционал: Метод удаления связи с заданными параметрами..

Параметры: HAN_LOC_SIG source - сигнал, base* target - слушатель, HAN_LOC handler - обработчик.

Возвращаемое значение: void (отсутствует).

Алгоритм метода представлен в таблице 36.

Таблица 36 – Алгоритм метода deleteConnect класса field_object

№	Предикат	Действия	№ перехода
1		Объявление эталонной связи типа bridge.	2
2		Настройка параметров эталонной связи в соответствии с входными данными.	3
3		Удаление элемента из массива связей (rels) данного объекта новой связи, равного эталонному.	∅

4 БЛОК-СХЕМЫ АЛГОРИТМОВ

Представим описание алгоритмов в графическом виде на рисунках 1-19.

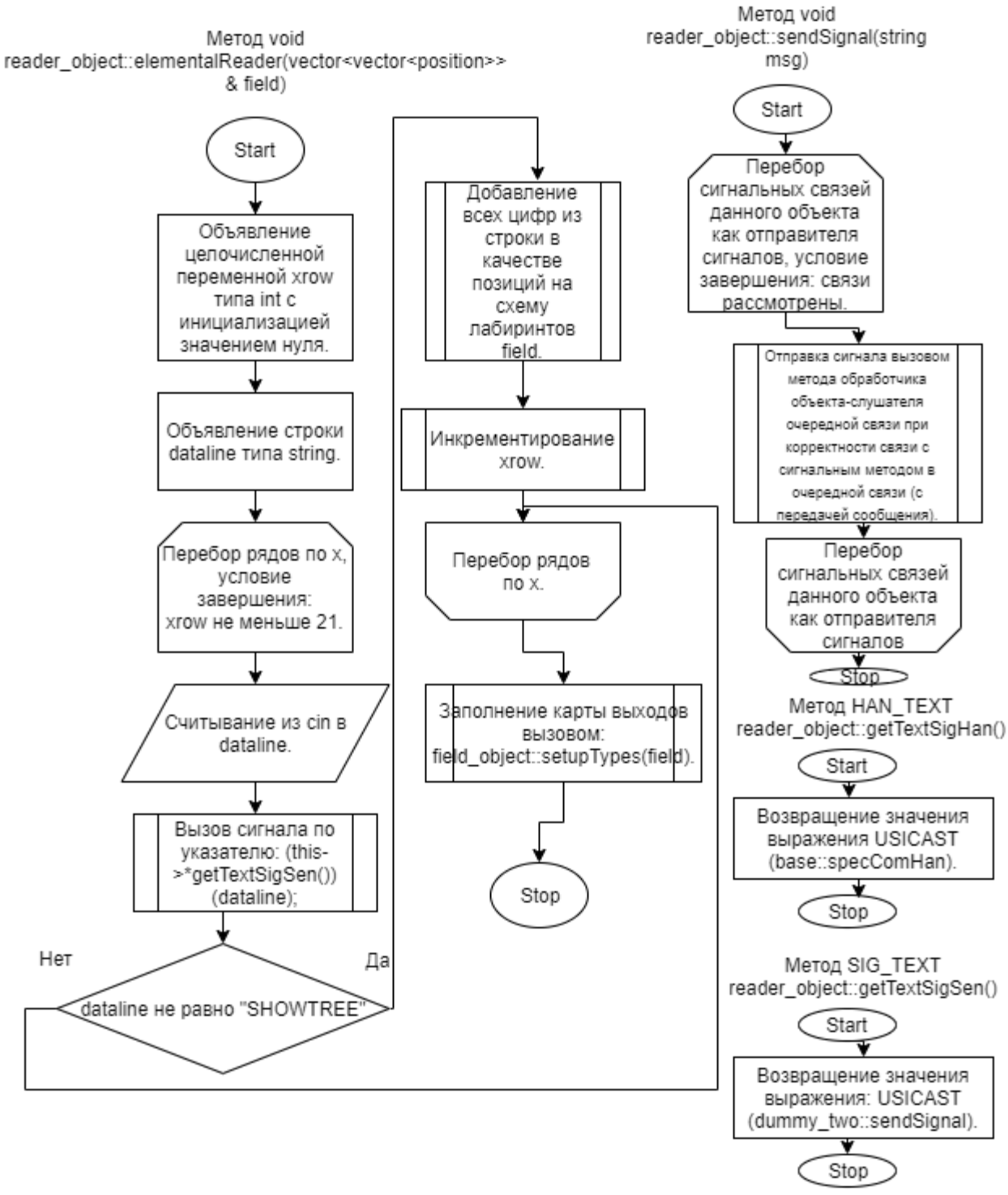

Рисунок 1 – Блок-схема алгоритма

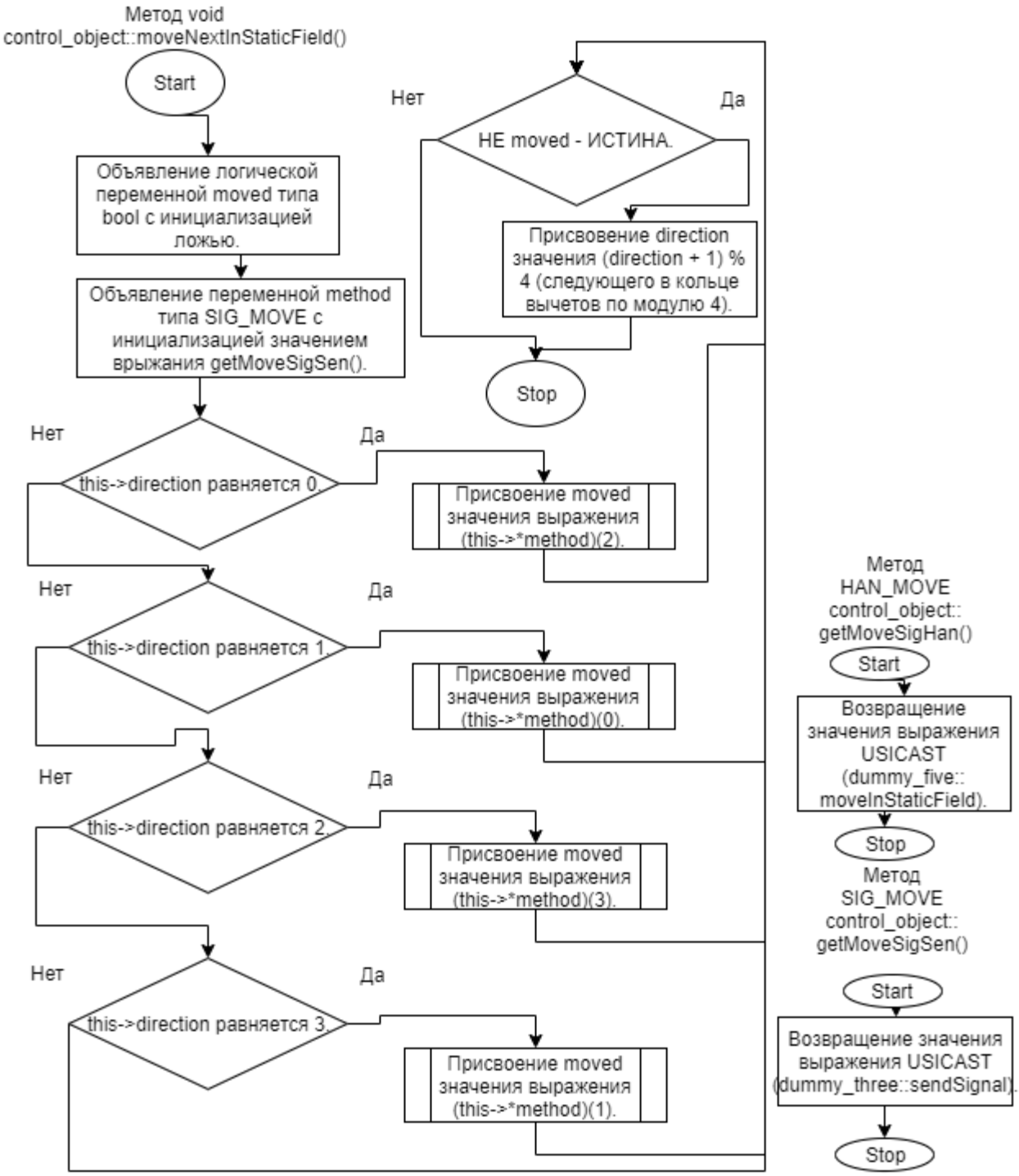

Рисунок 2 – Блок-схема алгоритма

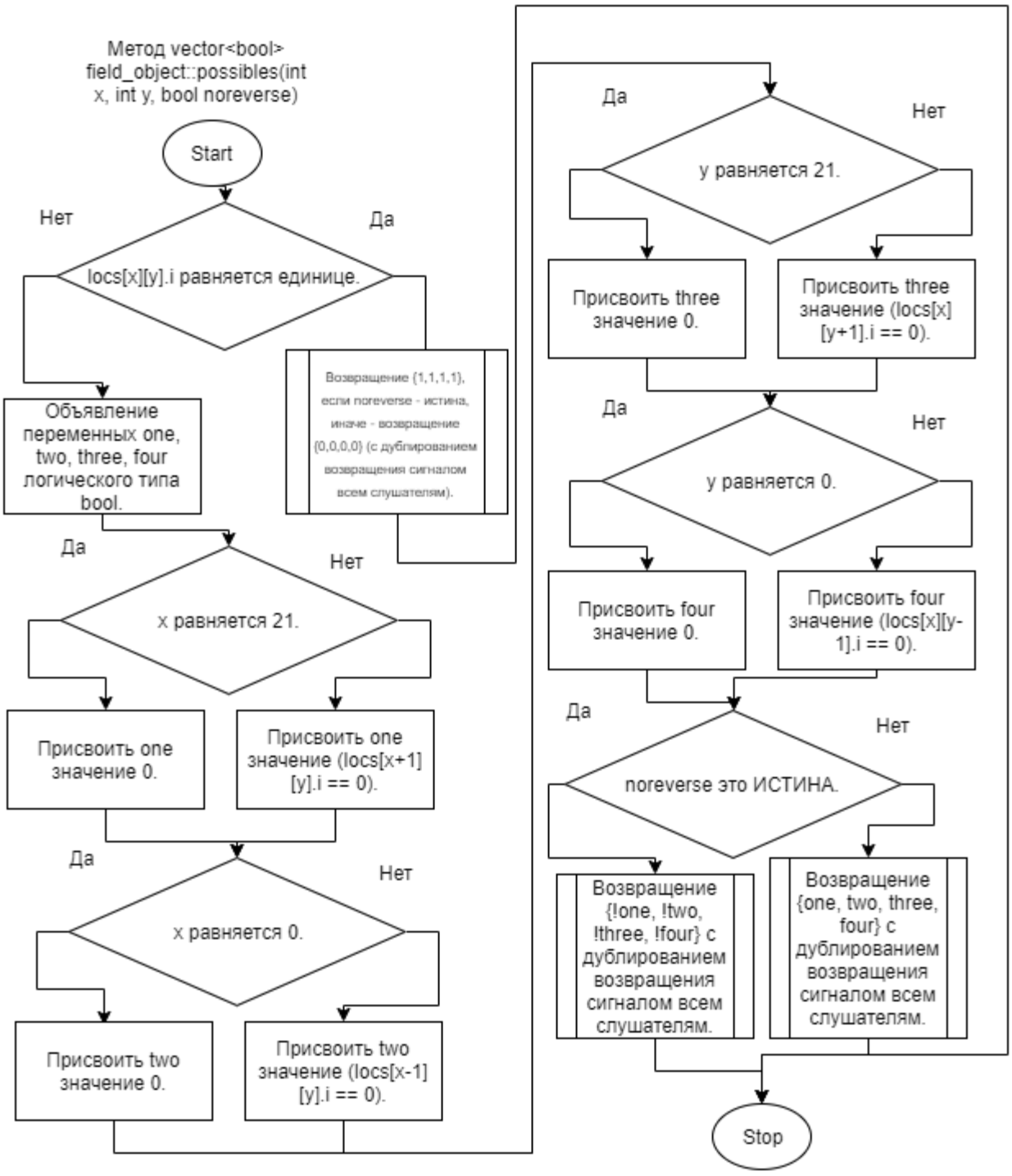

Рисунок 3 – Блок-схема алгоритма

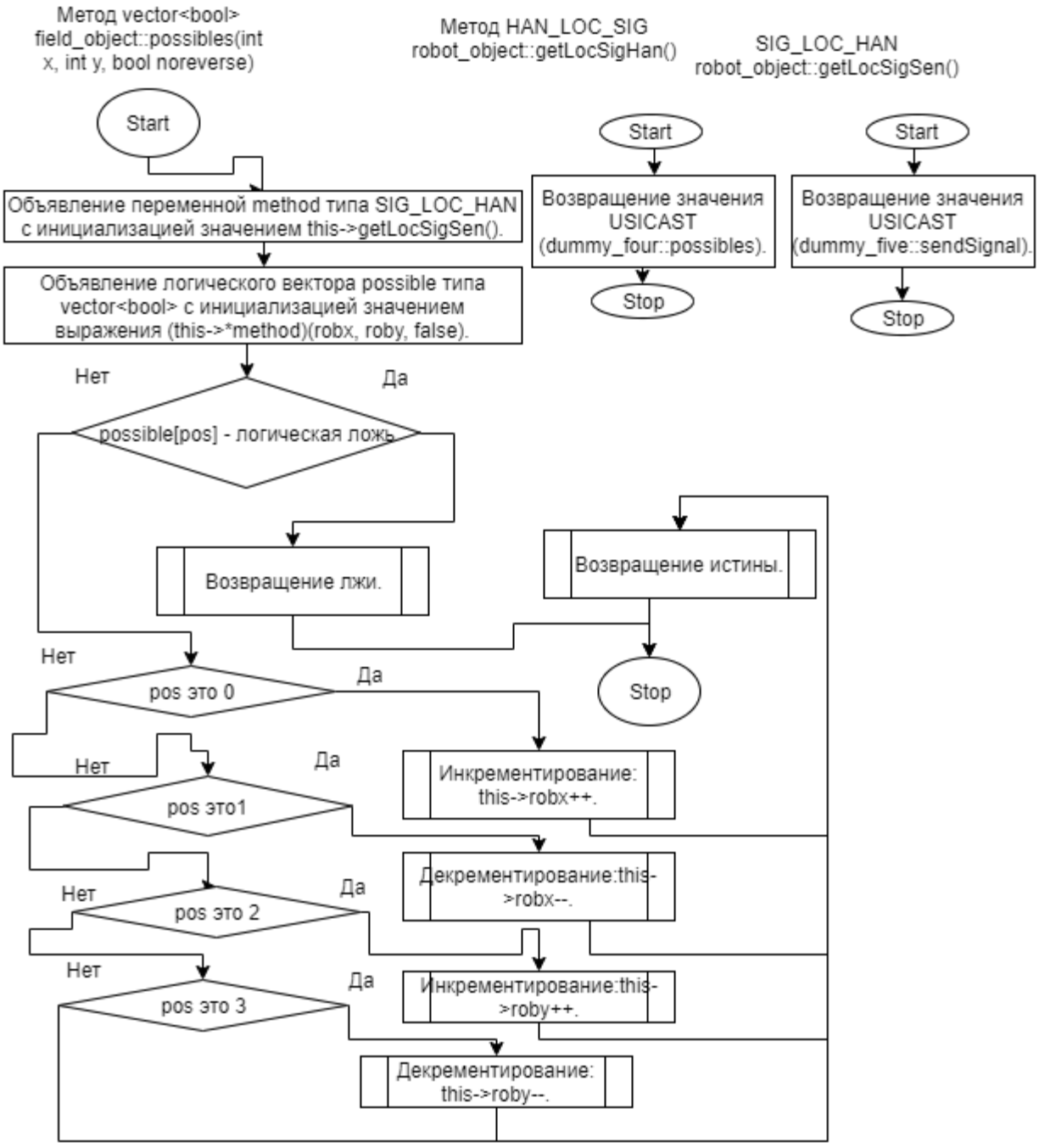

Рисунок 4 – Блок-схема алгоритма

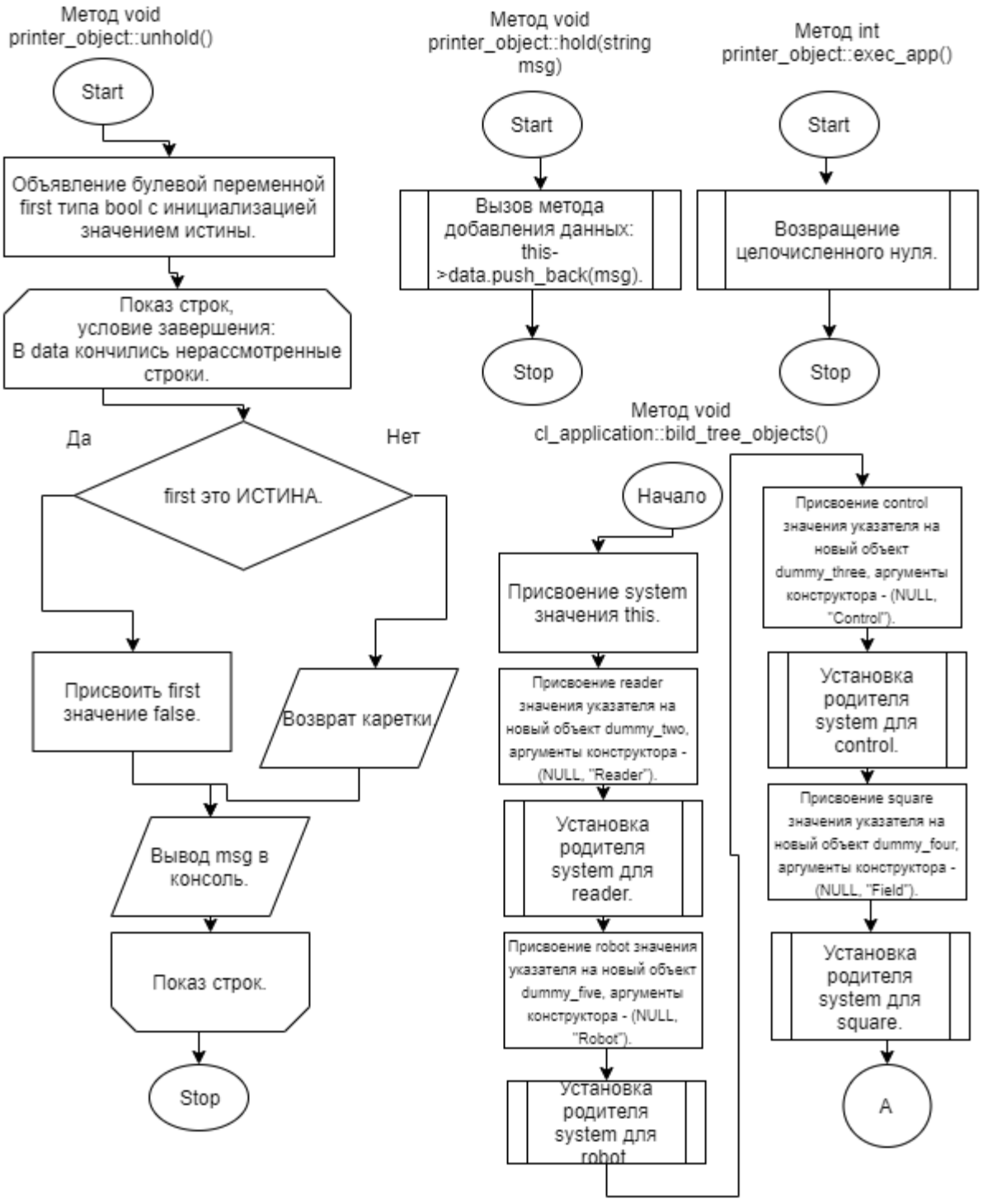

Рисунок 5 – Блок-схема алгоритма

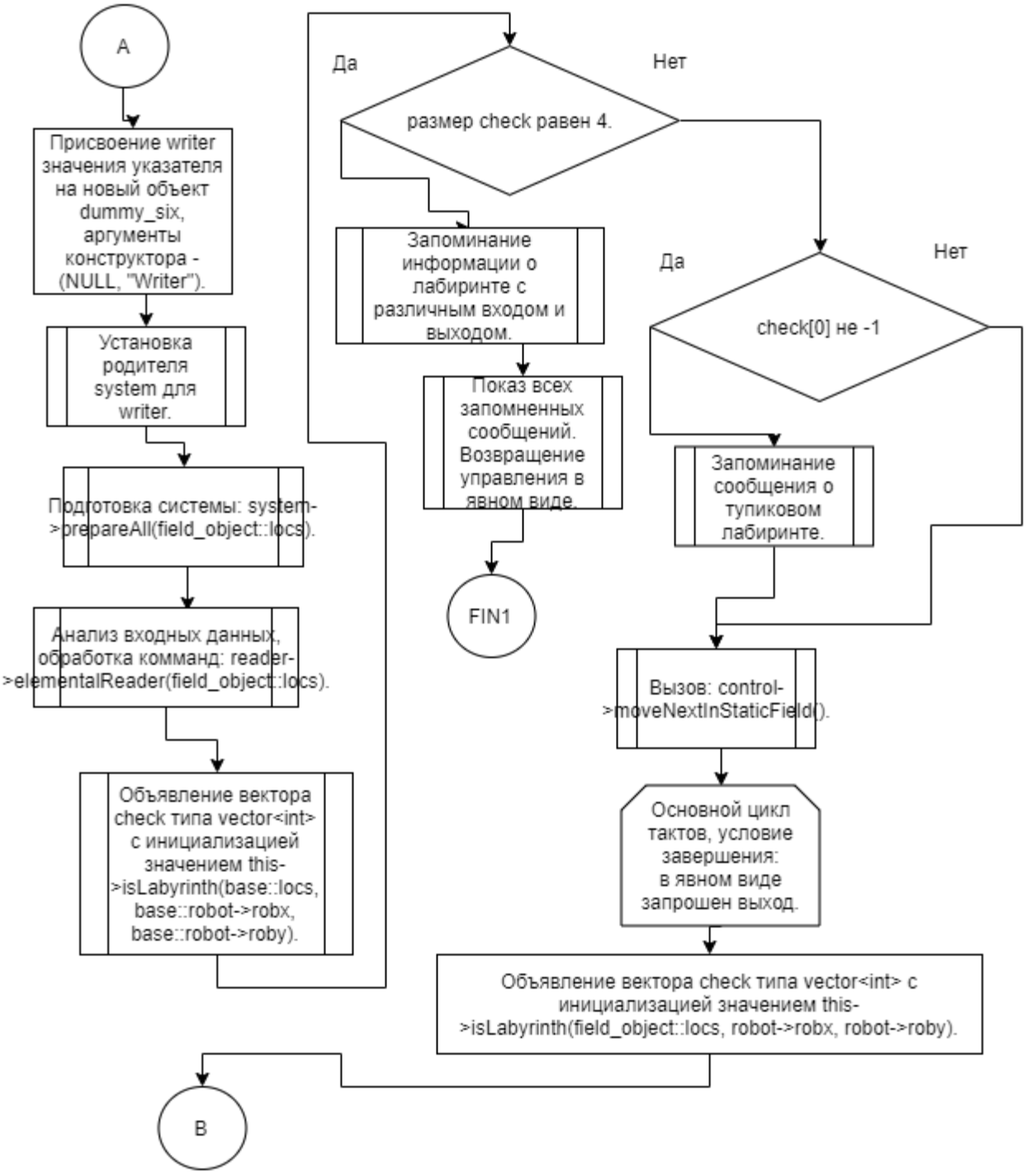

Рисунок 6 – Блок-схема алгоритма

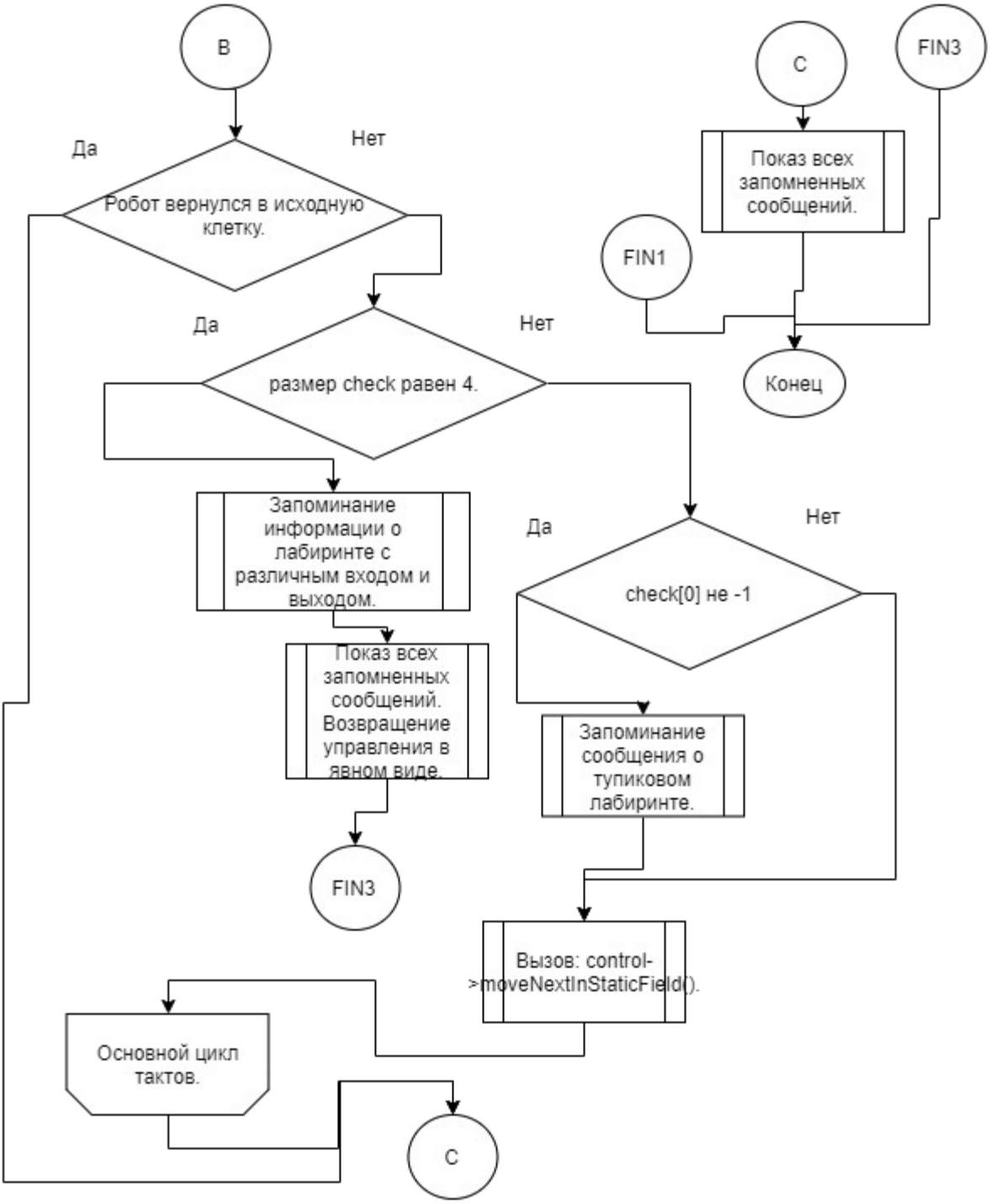

Рисунок 7 – Блок-схема алгоритма

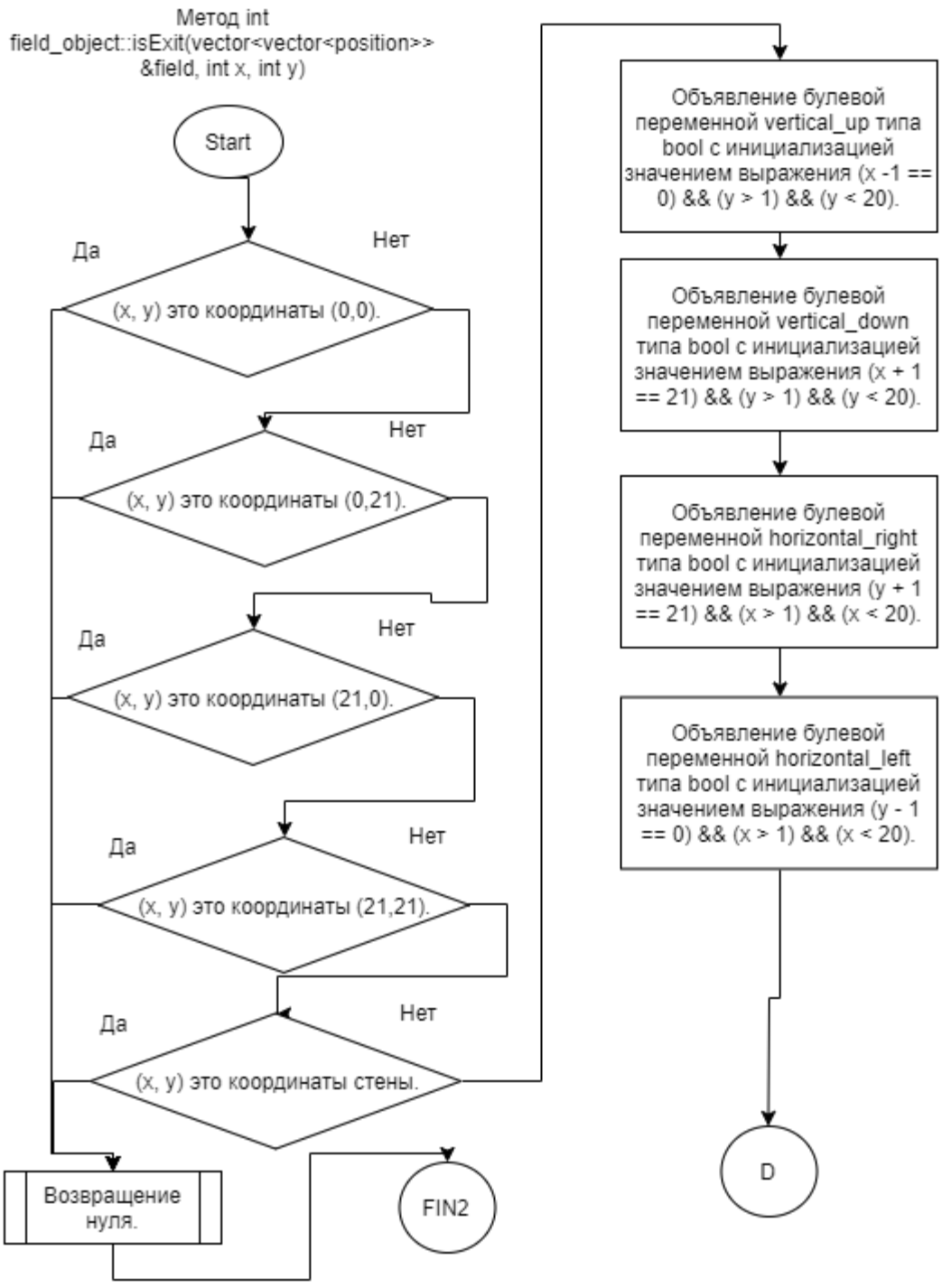

Рисунок 8 – Блок-схема алгоритма

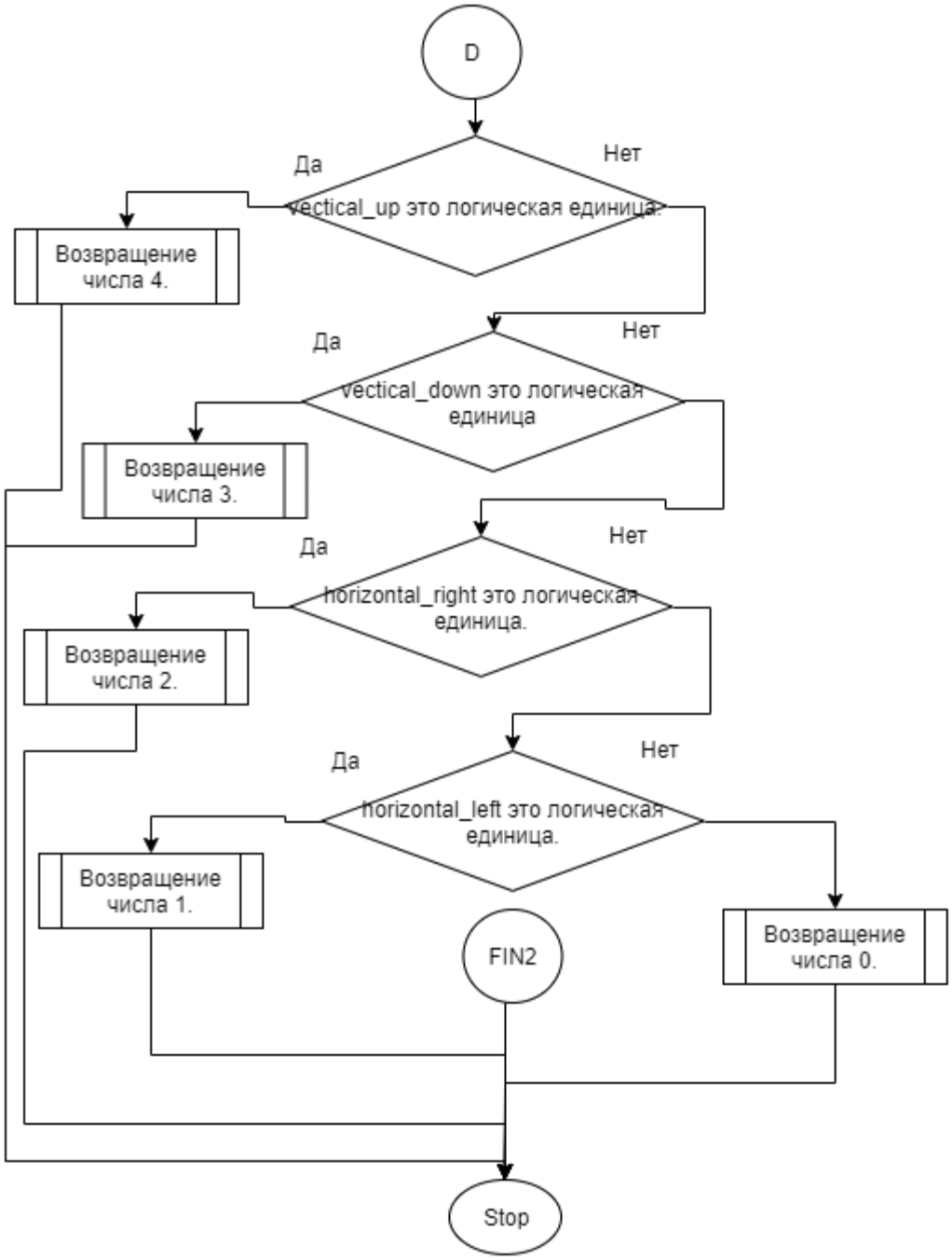

Рисунок 9 – Блок-схема алгоритма

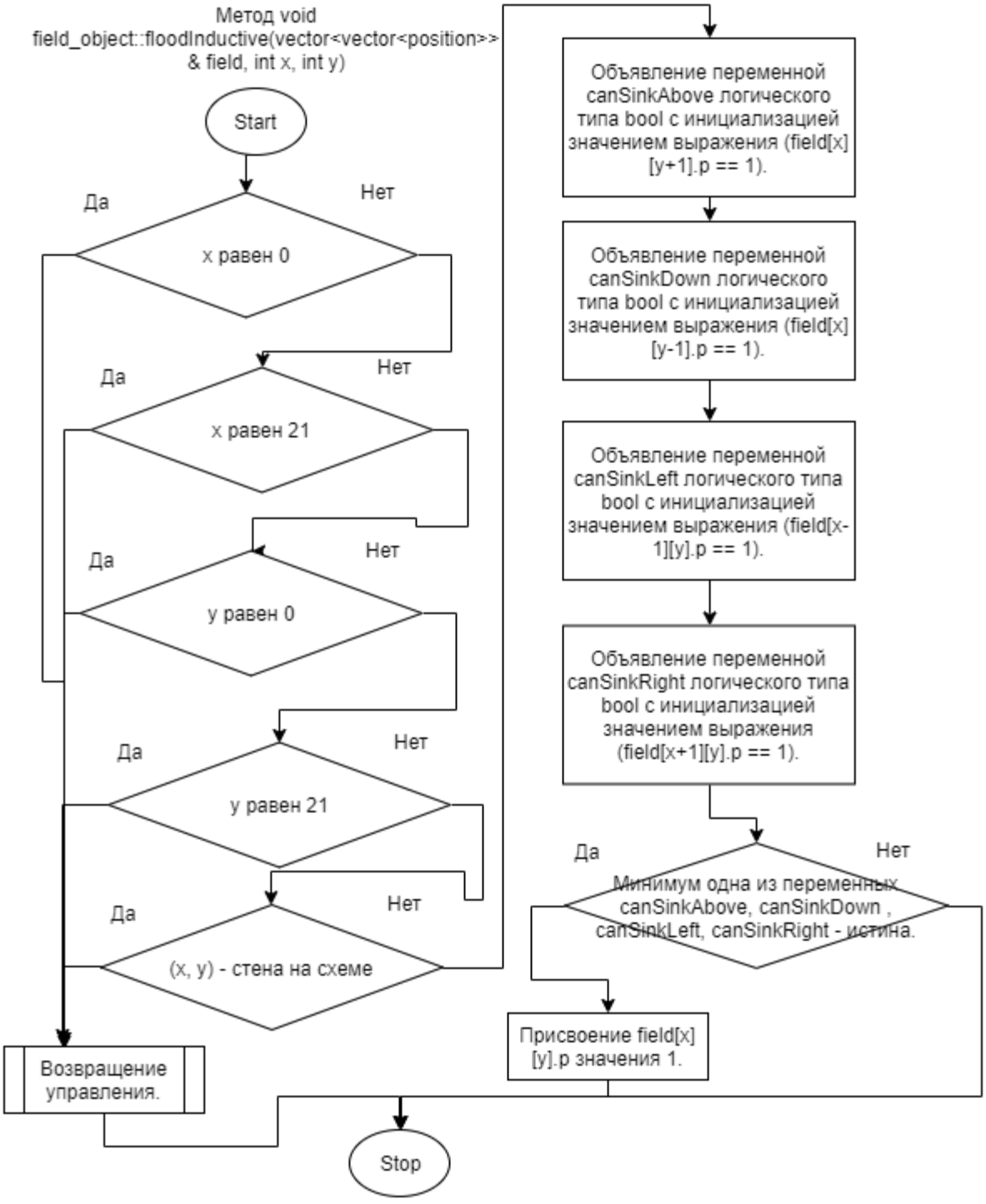

Рисунок 10 – Блок-схема алгоритма

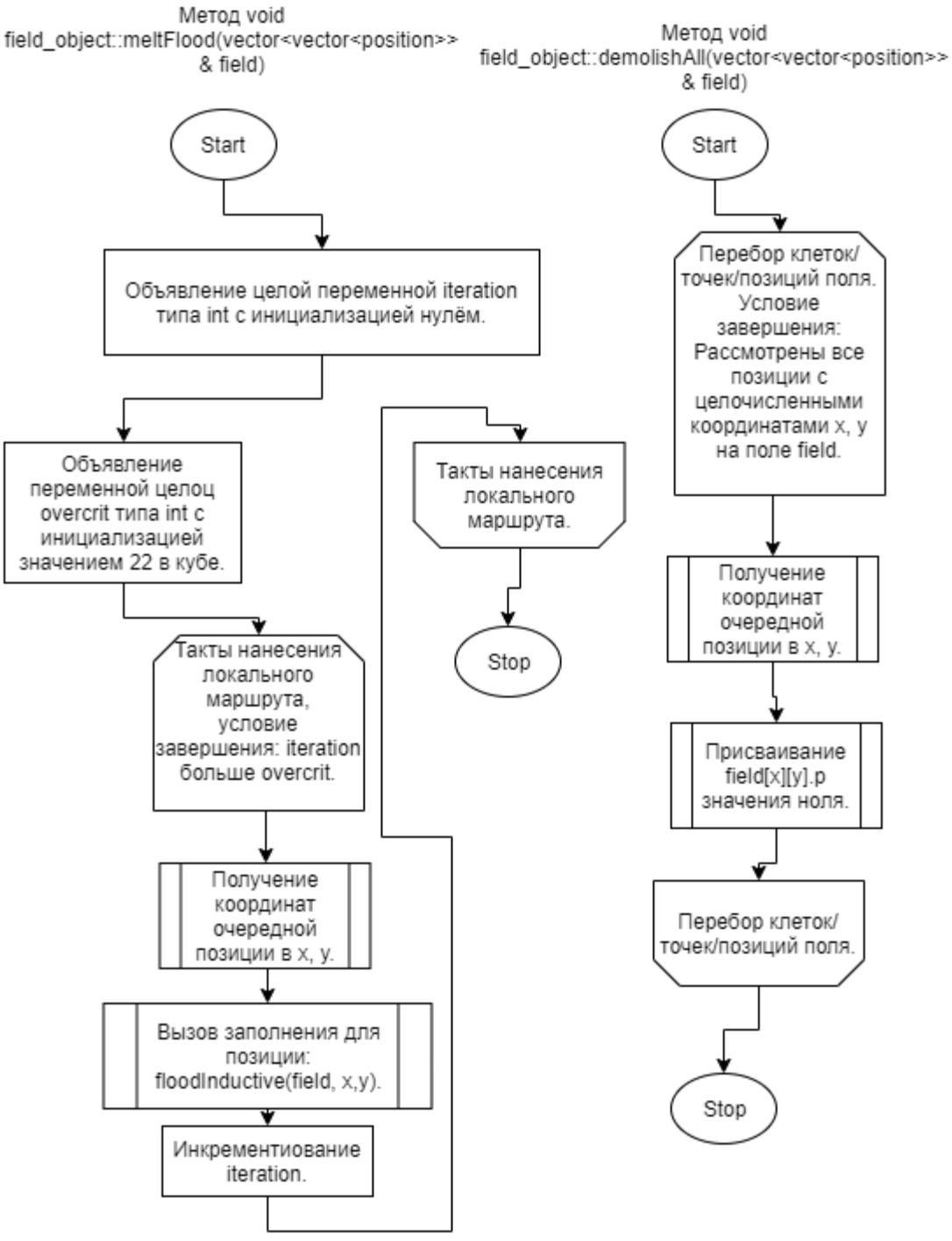

Рисунок 11 – Блок-схема алгоритма

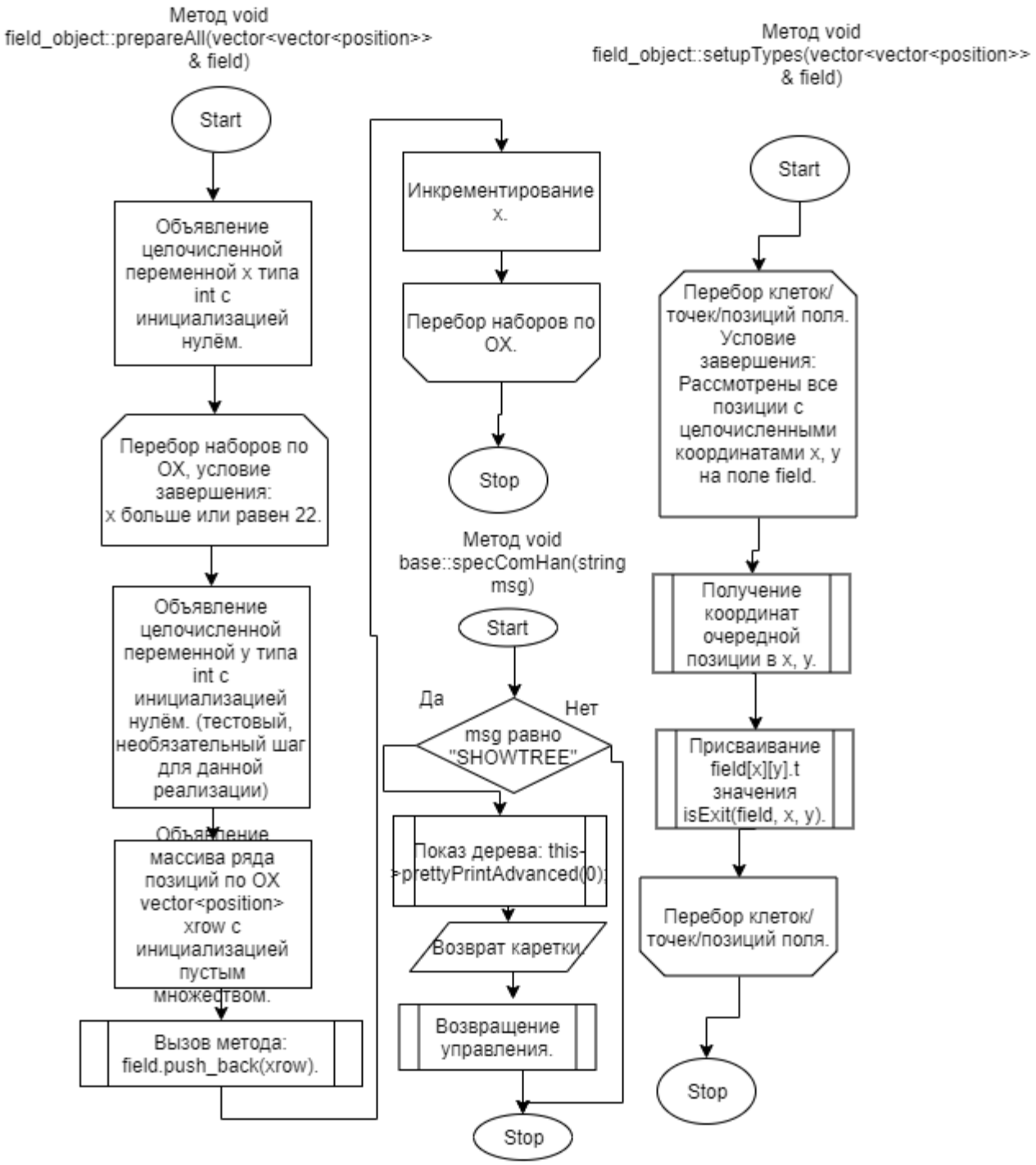

Рисунок 12 – Блок-схема алгоритма

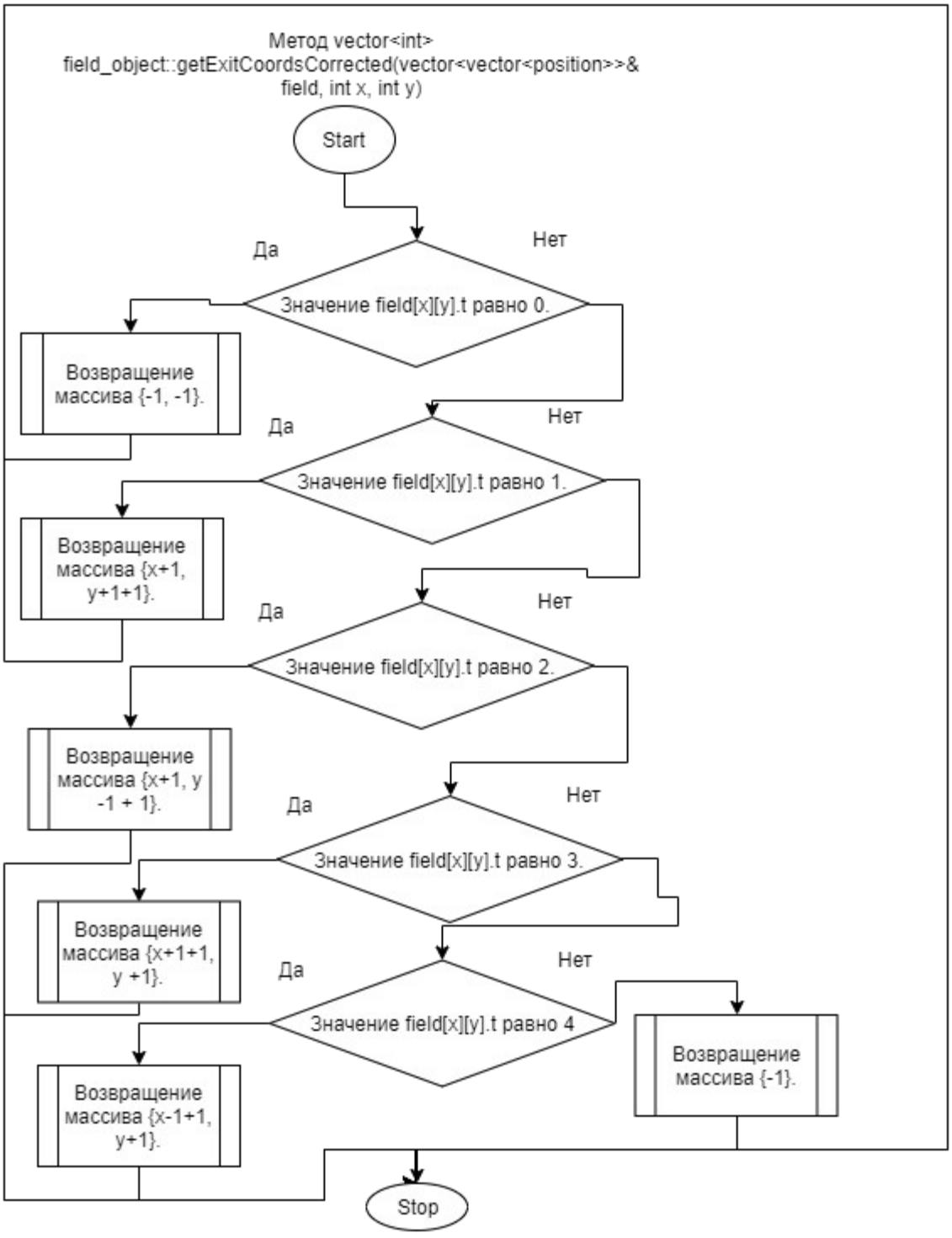

Рисунок 13 – Блок-схема алгоритма

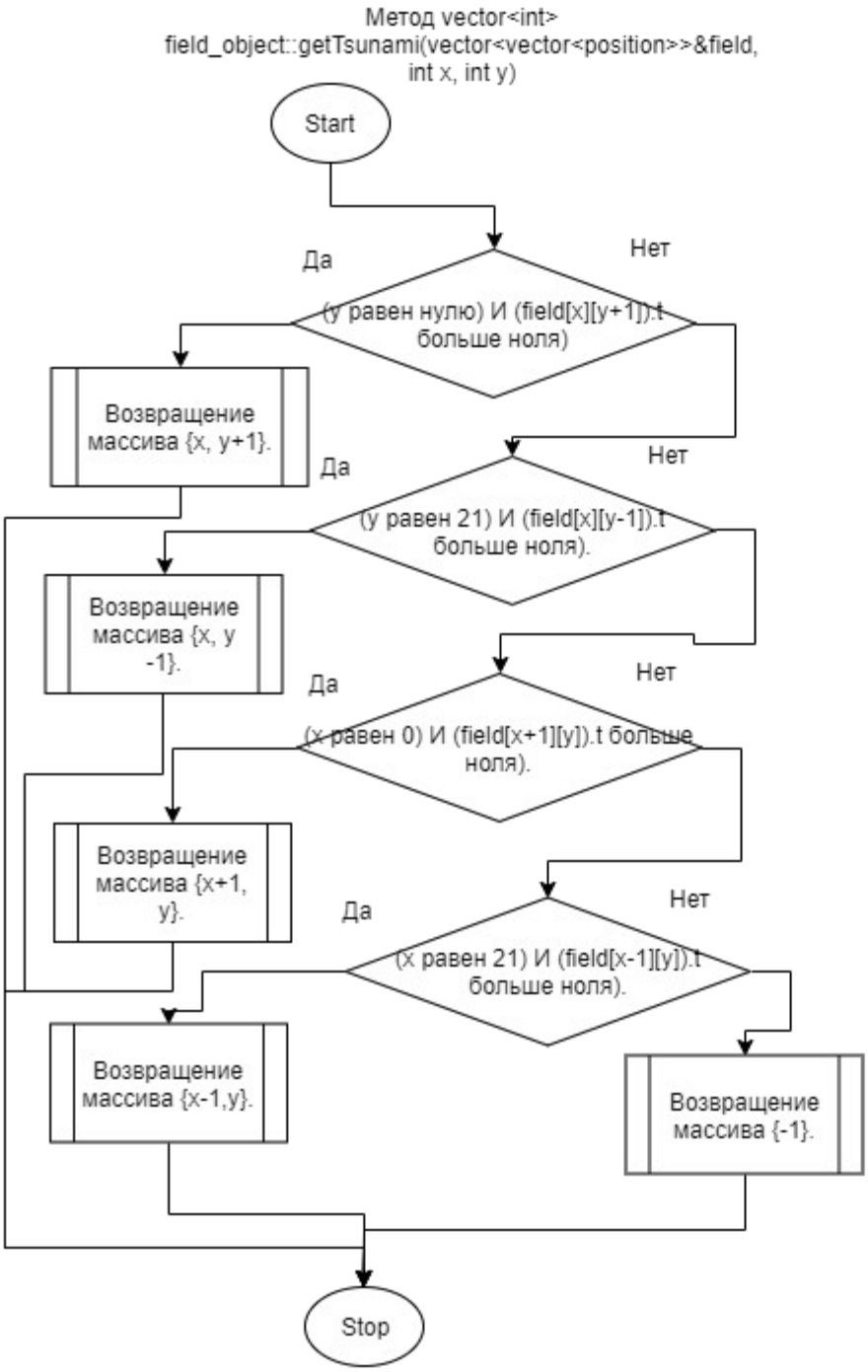

Рисунок 14 – Блок-схема алгоритма

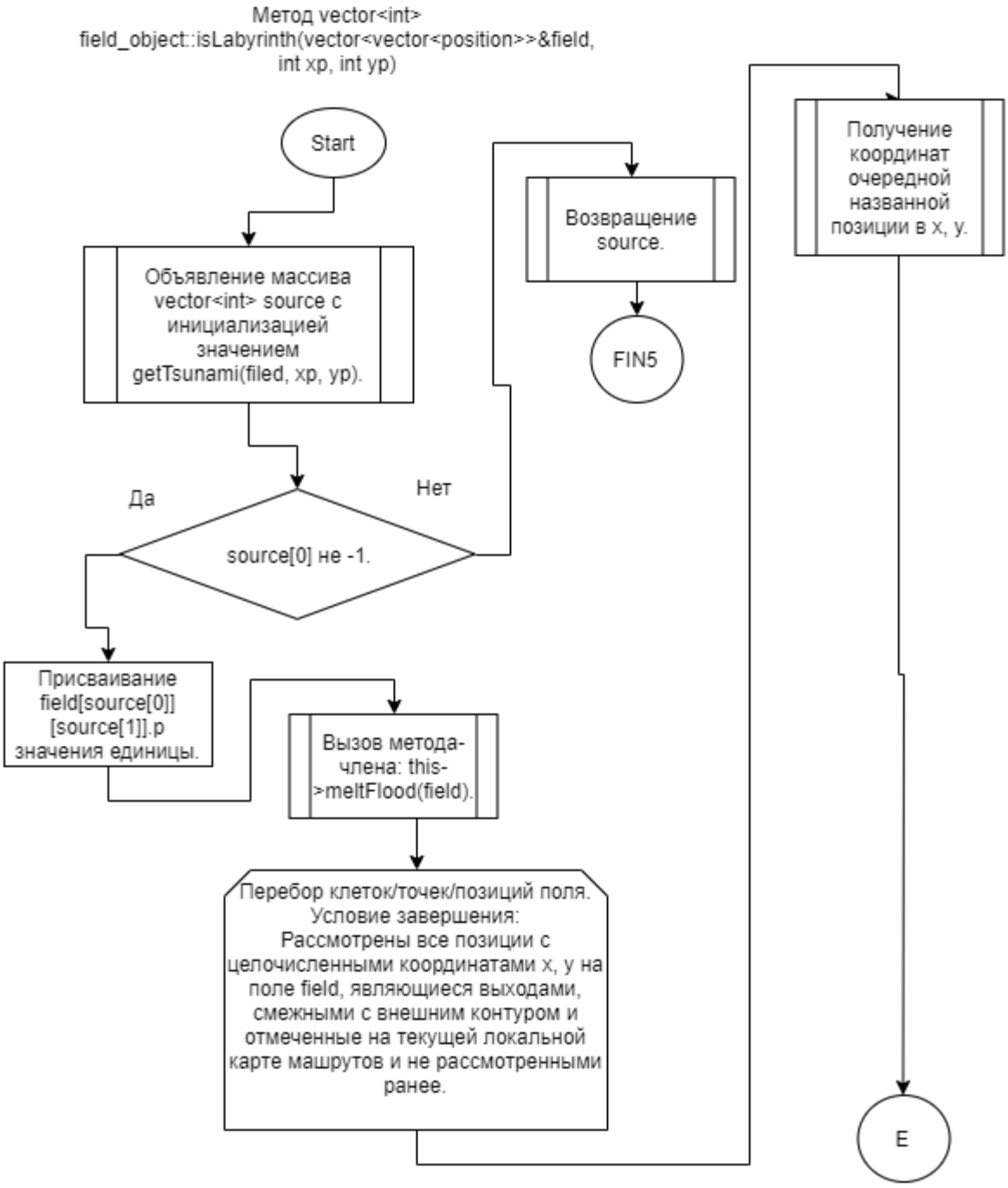

Рисунок 15 – Блок-схема алгоритма

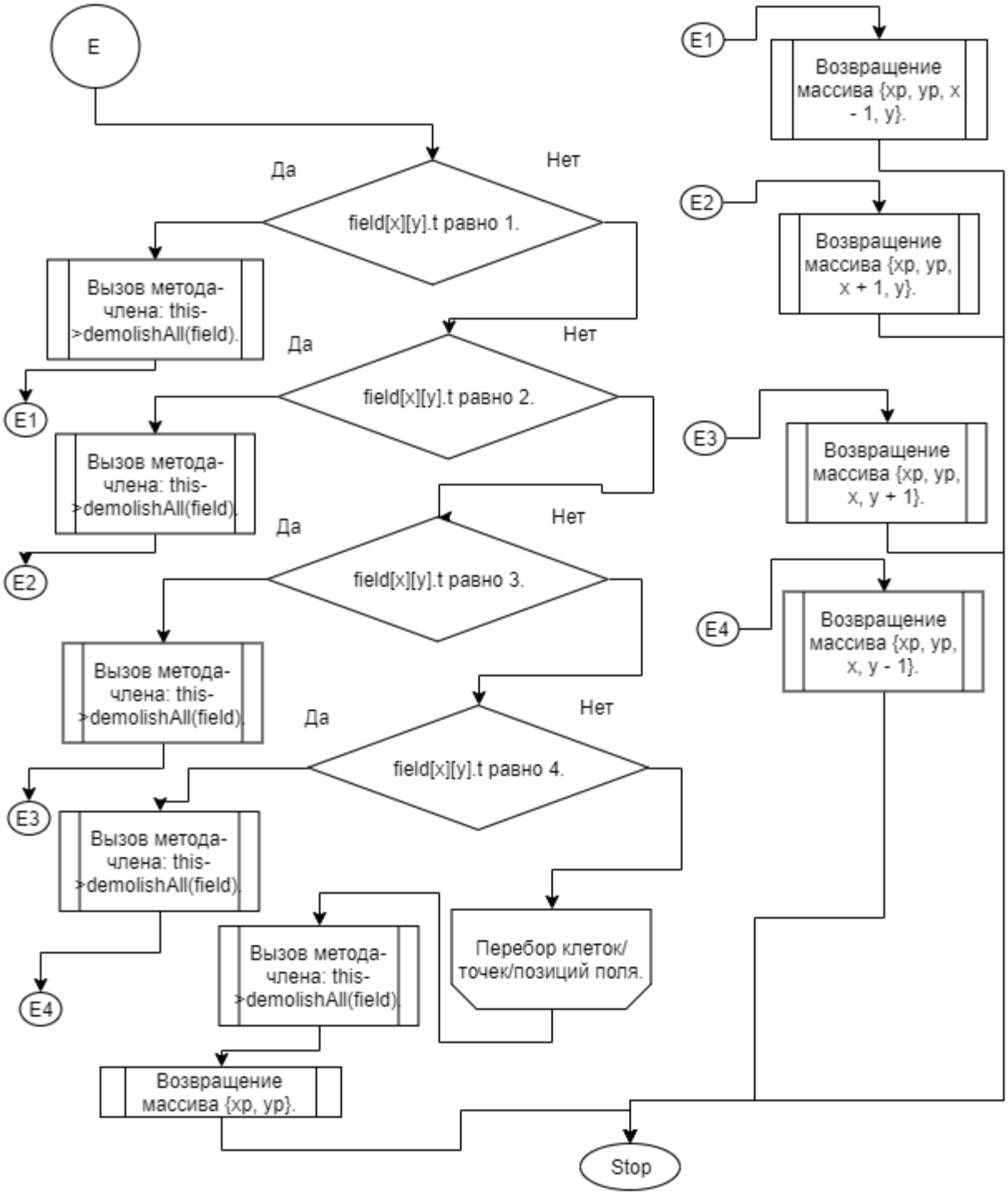

Рисунок 16 – Блок-схема алгоритма

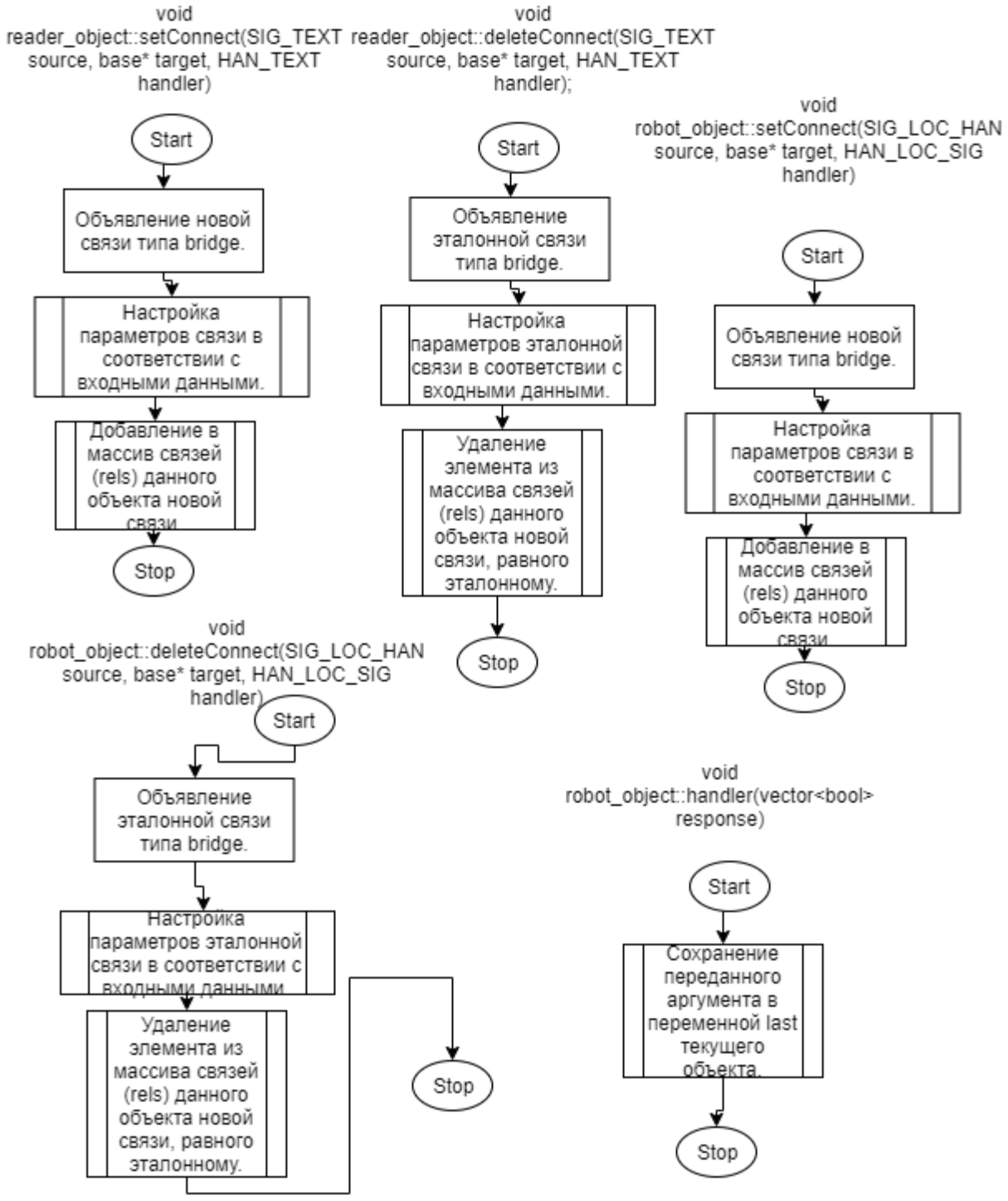

Рисунок 17 – Блок-схема алгоритма

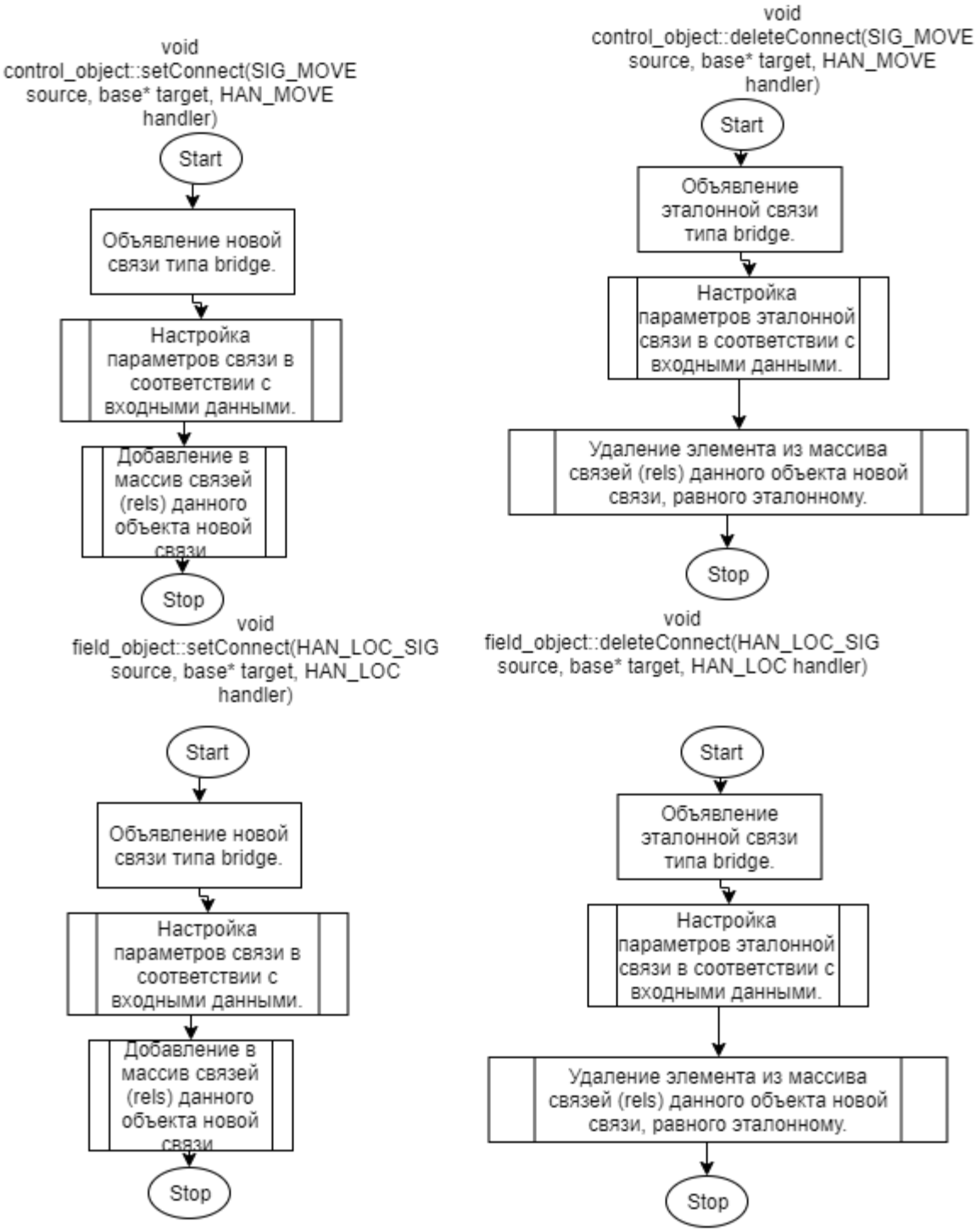

Рисунок 18 – Блок-схема алгоритма

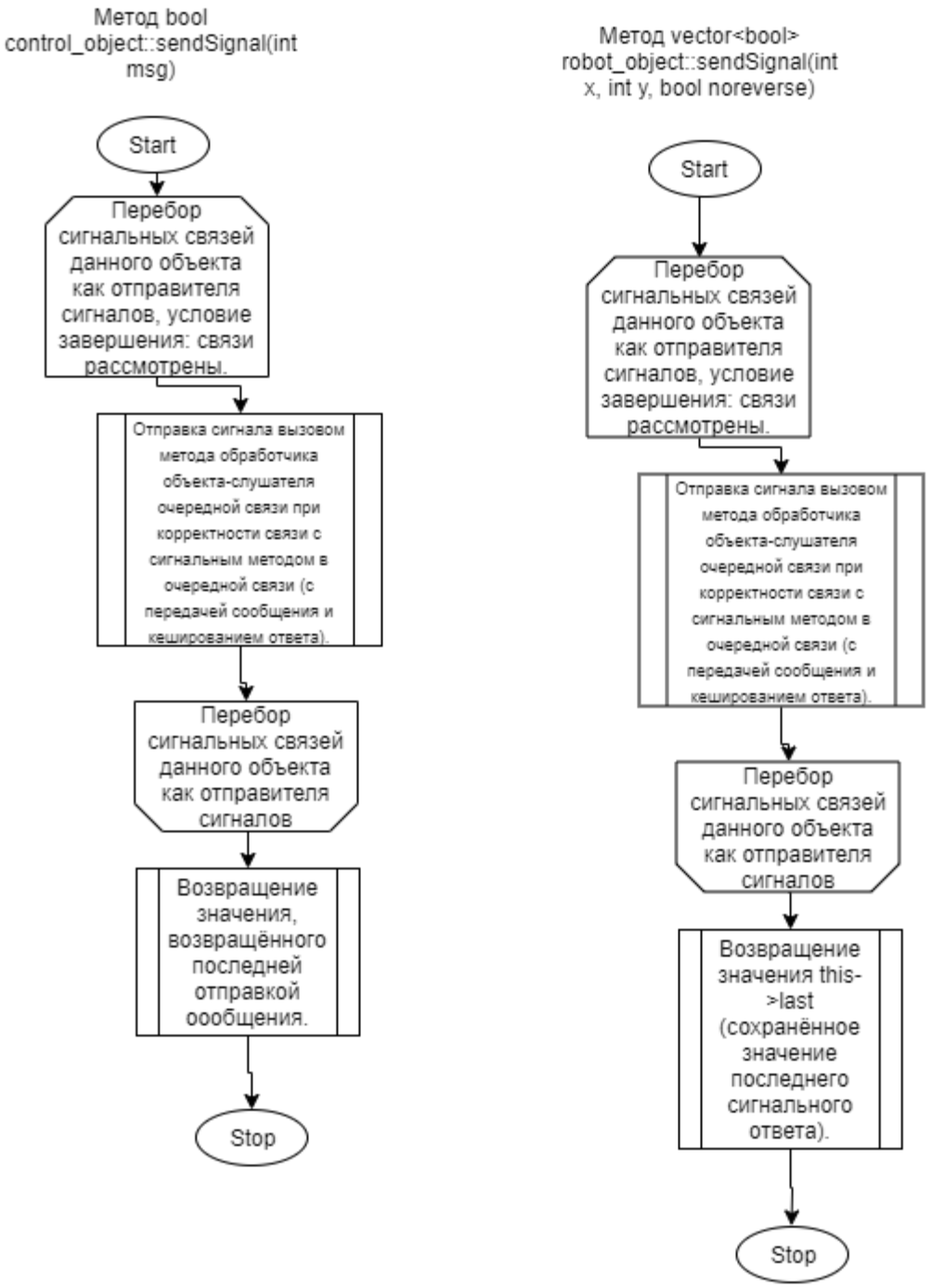

Рисунок 19 – Блок-схема алгоритма

5 КОД ПРОГРАММЫ

Программная реализация алгоритмов для решения задачи представлена ниже.

5.1 Файл base.cpp

Листинг 1 – base.cpp

```
#include "base.h"
#include <iostream>
#include <algorithm>
#include <vector>
#include <regex>
#include "field_object.h"
using namespace std;
//fix
base::base(base* predok, string imya){
	if (predok != NULL){
		this->setParent(predok);
	}
	this->name = imya;
};
string base::getName(){
	return base::name;
};
void base::setName(string imya){
	base::name = imya;
};
void base::setParent(base* predok){
	if (this->roditel != NULL){
		remove(base::roditel->deti.begin(), base::roditel->deti.end(), this);
	}
	this->roditel = predok;
	predok->deti.push_back(this);
};
base* base::getParent(){
	return base::roditel;
};
void base::setWake(int num){
	if (this->getParent() != nullptr){
	this->state = (bool(this->getParent()->isActuallyWake())) ? num : this-
>state;
	} else{
		this->state = num;
	}
	if (this->state == 0){
	for (base* kid : this->deti){
		kid->setWake(0);
	}
```

```
	}
};

bool base::isActuallyWake(){
	if (this->getParent() != nullptr){
	return (this->getParent()->isActuallyWake()) && (this->state != 0);
	} else{
		return (this->state != 0);
	}
};

void base::prettyPrint(int level){
	int clevel = level;
	while (clevel > 0){
		cout << "    ";
		clevel--;
	}
	cout << this->getName();
	for (base* baby : base::deti){
		cout<<endl;
		baby->prettyPrint(level + 1);
	}

};

base* base::findPointerSlow(string ident){
	if (this->name == ident){
		return this;
	} else{
		for (base *kid : this->deti){
			if (kid->findPointerSlow(ident)){
				return kid->findPointerSlow(ident);
			}
		}
	}
	return NULL;
};

void base::prettyPrintAdvanced(int level){
	int clevel = level;
	while (clevel > 0){
		cout << "    ";
		clevel--;
	}
	cout << this->getName() << " is " << ((!this->isActuallyWake())?"not ":"")<<
"ready";
	for (base* baby : base::deti){
		cout<<endl;
		baby->prettyPrintAdvanced(level + 1);
	}

};
base::base(){};
void base::specComHan(string msg){
	if (msg == "SHOWTREE"){
		this->prettyPrintAdvanced(0);
		cout << endl;
```

```
            return;
      }
}
/*
base* base::findPointerLoc(string data, string cur_loc= "/"){
      if (data == "" || data == "//") {
            return NULL;
      }
      base* me = this;
      auto jumpToKerL = [](auto& self, base * me) -> base*{
            if (me->getParent() != NULL){
                  return self(self, me->getParent());
            }
            return me;
      };
      auto preparePathL = [&jumpToKerL, &me](base* target) -> string{
            if (target == jumpToKerL(jumpToKerL, me)){
                  return "/";
            } else{
            auto getFolders = [] (auto &self, map<base*, string> &tofill, base*
begin, string start = "") -> void{
            for (base *kid : begin->deti){
                  tofill[kid] = start;
                  self(self, tofill, kid, start + "/" + kid->name);
            }
            };
            map<base*, string> tofill;
            getFolders(getFolders, tofill, jumpToKerL(jumpToKerL, me));
            return tofill[target] + "/" + target->name;
            }
            };
      if ((data == "/") || (data == "." && cur_loc == "/")){
            return jumpToKerL(jumpToKerL, me);
      }
      string positive = "";
      if ((data.rfind("//", 0) == 0)){
            string pure = "";
            for (char c : data){
                  pure += (string(1,c) != "/") ? string(1,c) : "";
            }
            auto findPointerSlowL = [](auto & self, string name, base* dest) ->
base* {
            if (dest->name == name){
                  return dest;
            } else{
            for (base *kid : dest->deti){
                  if (self(self,name, kid)){
                        return self(self, name, kid);
                  }
            }
            }
            return NULL;
            };
            return findPointerSlowL(findPointerSlowL, pure, jumpToKerL(jumpToKerL,
me));
      } else {
            if (data.rfind(".", 0) == 0){
```

```
                  positive = cur_loc;
            } else if (data.rfind("/", 0) != 0){
                  positive = cur_loc + "/" + data;
            } else{
                  positive = data;
            }
      }
      auto findPointerSlowLocL = [&preparePathL](auto & self, string name, base*
dest) -> base*{
            if (preparePathL(dest) == regex_replace(name, regex("//"), "/")){
                  return dest;
            } else{
            for (base *kid : dest->deti){
                  if (self(self,name, kid)){
                        return self(self, name, kid);
                  }
            }
            }
            return NULL;
            };
      return           findPointerSlowLocL(findPointerSlowLocL,           positive,
jumpToKerL(jumpToKerL, me));
}
string base::myLoc(){
      base* me = this;
      auto jumpToKerL = [](auto& self, base * me) -> base*{
            if (me->getParent() != NULL){
                  return self(self, me->getParent());
            }
            return me;
      };
      if (this == jumpToKerL(jumpToKerL, this)){
            return "/";
      }
      auto preparePathL = [&jumpToKerL, &me](base* target) -> string{
            if (target == jumpToKerL(jumpToKerL, me)){
                  return "/";
            } else{
            auto getFolders = [] (auto &self, map<base*, string> &tofill, base*
begin, string start = "") -> void{
            for (base *kid : begin->deti){
                  tofill[kid] = start;
                  self(self, tofill, kid, start + "/" + kid->name);
            }
            };
            map<base*, string> tofill;
            getFolders(getFolders, tofill, jumpToKerL(jumpToKerL, me));
            return tofill[target] + "/" + target->name;
            }
            };
      return regex_replace(preparePathL(this), regex("//"),"/");
}*/
/*
void base::grabGlobal(string test){
      if (this->isActuallyWake()){
            cout << endl;
            cout << "Signal to " + this->myLoc() + " Text:  " + test;
```

```
	}
}
void base::emitLocal(string &message, pairz connection){
	if (this->isActuallyWake()){
		CONCAST(connection)(message + " (class: " + to_string(this->typenum) +
")");
	}
}
void base::emitGlobal(TYPE_SIG tipsig, string &message){
	if (this->isActuallyWake()){
				cout << endl;
				cout << "Signal from " + this->myLoc();
				}
	for (auto connection : this->relations){
		if (tipsig == connection.zero){
				SIGCAST(connection.zero)(message, connection);
		}
	}
}
void base::setBridge(TYPE_SIG used, base* destination, TYPE_HANDLER handler){
pairz para;
para.zero = used;
para.first = destination;
para.second = handler;
this->relations.push_back(para);
}
void base::killBridge(TYPE_SIG used, base* destination, TYPE_HANDLER handler){
pairz para;
para.zero = used;
para.first = destination;
para.second = handler;
this->relations.erase(remove(relations.begin(),	relations.end(),	para),
relations.end());
}
TYPE_SIG base::getEmitter(){
	return LSIGCAST(base::emitLocal);
}
TYPE_HANDLER base::getHandler(){
	return LCONCAST(base::grabGlobal);
}*/
```

5.2 Файл base.h

Листинг 2 – base.h

```
#ifndef BASE
#define BASE
#include <string>
#include <vector>
#include <utility>
#include <map>
#include <algorithm>
#define USICAST(n) &n
```

```
//External subjective requirement.
#define dummy_two reader_object
#define dummy_three control_object
#define dummy_four field_object
#define dummy_five robot_object
#define dummy_six printer_object
using namespace std;
class base;
class cl_application;
class dummy_two;
class dummy_three;
class dummy_four;
class dummy_five;
class dummy_six;
/*struct pairz;
typedef void(base::* TYPE_HANDLER)(string);
typedef void(base::* TYPE_SIG)(string&, pairz);*/
/*#define SIGCAST(n) (this->*n)
#define CONCAST(n) (n.first->*n.second)
#define LSIGCAST(n) &n
#define LCONCAST(n) &n
struct pairz{
      TYPE_SIG zero;
      base* first;
      TYPE_HANDLER second;
      bool operator==(const pairz& para){
            return ((this->first == para.first) && (this->second == para.second)
&& (this->zero == para.zero));
      }
};*/
struct position{
      int i = 0;
      int t = 0;
      int p = 0;
};
typedef void(dummy_two::* SIG_TEXT)(string msg);//I send text commands.
typedef void(base::* HAN_TEXT)(string msg); //I handle text commands.
typedef bool(dummy_five::* HAN_MOVE)(int pos);//I handle move commands.
typedef bool(dummy_three::* SIG_MOVE)(int pos);//I send move commands.
typedef vector<bool>(dummy_five::*SIG_LOC_HAN)(int x, int y, bool noreverse); //I
send info commands and manage responses.
typedef vector<bool>(dummy_four::*HAN_LOC_SIG)(int x, int y, bool noreverse); //I
handle info commands and send responses back directly (unused) and indirectly.
typedef   void   (robot_object::*HAN_LOC)   (vector<bool>   response);//I   recieve
responses.
struct bridge{
      int bridge_type = 0;
      base* responder = NULL;
      SIG_TEXT call_t = NULL;
      HAN_TEXT han_t = NULL;
      HAN_MOVE han_m = NULL;
      SIG_MOVE sig_m = NULL;
      SIG_LOC_HAN sig_ld = NULL;
      HAN_LOC_SIG han_ld = NULL;
      HAN_LOC han_loc = NULL;
      bool operator==(const bridge& coll){
            return (coll.bridge_type == this->bridge_type) && (coll.han_t == this-
```

```
>han_t) && (coll.han_m == this->han_m) && (coll.sig_m == this->sig_m) &&
(coll.sig_ld == this->sig_ld) && (coll.han_ld == this->han_ld) && (coll.han_loc ==
this->han_loc);
	}
};
class base{
public:
	string name;
	base* roditel = NULL;
	int state = 1;//Now I am on.
	base();
	base(base* predok, string imya = "Object_root");
	void setName(string name);
	string getName();
	void setParent(base* predok);
	base* getParent();
	void prettyPrint(int level);
	void prettyPrintAdvanced(int level);
	void setWake(int num);
	base* findPointerSlow(string ident);
	bool isActuallyWake();
	base* findPointerLoc(string data, string cur_loc);
	vector<base*> deti = {};
	vector<bridge> rels = {};
	/*void emitGlobal(TYPE_SIG tipsig, string &message);
	void grabGlobal(string message);
	void emitLocal(string& message, pairz);
	void setBridge(TYPE_SIG used, base* destination, TYPE_HANDLER handler);
	void killBridge(TYPE_SIG used, base* destination, TYPE_HANDLER handler);
	string myLoc();
	vector<pairz> relations = {};
	TYPE_SIG getEmitter();
	TYPE_HANDLER getHandler();*/
	void specComHan(string msg);
int typenum = 1;
};

#endif
```

5.3 Файл cl_application.cpp

Листинг 3 – cl_application.cpp

```
#include "cl_application.h"
#include "base.h"
#include "control_object.h"
#include "reader_object.h"
#include "field_object.h"
#include "robot_object.h"
#include "printer_object.h"
#include <iostream>
#include <string>
#include <sstream>
```

```
#include <vector>
#include <map>
#include <algorithm>
#include <regex>
using namespace std;
//base * pioneer; //Notice: some deprecated code is hidden from the compilator by
comment through the program.
//Changed not much...
void cl_application::bild_tree_objects(){
      //Preparation.
      system = this;
      reader = new dummy_two(system, "Reader");
      //base::reader->setParent(base::system);
      robot = new dummy_five(system, "Robot");
      //base::robot->setParent(base::system);
      control = new dummy_three(system, "Control");
      //base::control -> setParent(base::system);
      square = new dummy_four(system, "Field");
      //base::square->setParent(base::system);
      writer = new dummy_six(system, "Writer");
      //base::writer->setParent(base::system);
      control->setConnect(control->getMoveSigSen(),     static_cast<base*>(robot),
control->getMoveSigHan());
      reader->setConnect(reader->getTextSigSen(),      static_cast<base*>(system),
reader->getTextSigHan());
      robot->setConnect(robot->getLocSigSen(),  static_cast<base*>(square),  robot-
>getLocSigHan());
      square->setConnect(robot->getLocSigHan(),         static_cast<base*>(robot),
(&robot_object::handler));
      field_object::prepareAll(field_object::locs);
      reader->elementalReader(field_object::locs);
      vector<int>  check  =  square->isLabyrinth(field_object::locs,  robot->robx,
robot->roby);
      if (check.size() == 4){
            writer->hold("Maze  ("  +  to_string(check[0]  +  1)  +  ",  "  +
to_string(check[1]  +  1)  +  ")  ("  +  to_string(check[2]  +  1)  +  ",  "  +
to_string(check[3] + 1) + ")");
            writer->unhold();
            return;
      } else if (check[0] != -1){
            writer->hold("There is no way out of the maze (" + to_string(check[0]
+ 1) + ", " + to_string(check[1] + 1) + ")");
      }
      control->moveNextInStaticField();
      //Start of main checks.
      while (true){
            //cout << robot->robx << " " << robot->roby << endl;
            vector<int>  check  =  square->isLabyrinth(field_object::locs,  robot-
>robx, robot->roby);
            if ((robot->robx == 0) && (robot->roby == 1)){
            //cout << "Made round..." << endl;
            break;
            }
            if (check.size() == 4){
                  //cout << "Found..." << endl;
                  //for (auto ti : check){
                        //cout << ti << endl;
```

```
                  //}

            writer->hold("Maze  ("  +  to_string(check[0]  +  1)  +  ",  "  +
to_string(check[1]  +  1)  +  ")  ("  +  to_string(check[2]  +  1)  +  ",  "  +
to_string(check[3] + 1) + ")");
                  //cout << locs[1][1].t << endl;
            writer->unhold();
            return;
      } else if (check[0] != -1){
            writer->hold("There is no way out of the maze (" + to_string(check[0]
+ 1) + ", " + to_string(check[1] + 1) + ")");
      }
            //cout << "GOing next..." << endl;
            control->moveNextInStaticField();
      }
      writer->unhold();
      //locs[0][6].p=1;
      //meltFlood(locs);
      /*int x = 0;
      while (x < 22){
            int y = 0;
            while (y < 22){
                  cout << locs[x][y].p;
                  y++;
            }
            x++;
      }*/
      /*
      pioneer = this;
      dummy_two reader = dummy_two(pioneer, "Reader");
      base::locs=  {{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},
{}};
      //reader.elementalReader(locs);
      for (vector<position> row : base::locs){
            cout << endl;
            for (position pos : row){
                  cout << pos.i << endl;
            }
      }*/
      /*string root_name;
      getline(cin, root_name);
      pioneer = this;
      pioneer->setName(root_name);
      string current_input;
      bool step = 0;
      string lru_cache = "/";
      while (getline(cin,current_input)){
            stringstream indata;
            indata << current_input;
            vector<string> rawdata = {};
            string input_part;
            while (indata >> input_part){
                  rawdata.push_back(input_part);
            }
            if (current_input == "endtree"){
                  cout << "Object tree" << endl;
                  pioneer -> prettyPrint(0);
```

```
                  step = 1;
            }
            else{
                  if (step == 0){
                  base* guest;
                  base* host = this->findPointerLoc(rawdata[0], lru_cache);
                        switch(stoi(rawdata[2])){
                              case 2:
                                    guest                   =                   new
dummy_two(static_cast<base*>(nullptr), rawdata[1]);
                                    break;
                              case 3:
                                    guest                   =                   new
dummy_three(static_cast<base*>(nullptr), rawdata[1]);
                                    break;
                              case 4:
                                    guest                   =                   new
dummy_four(static_cast<base*>(nullptr), rawdata[1]);
                                    break;
                              case 5:
                                    guest                   =                   new
dummy_five(static_cast<base*>(nullptr), rawdata[1]);
                                    break;
                              case 6:
                                    guest                   =                   new
dummy_six(static_cast<base*>(nullptr), rawdata[1]);
                                    break;
                        };
                        if (host != NULL){
                        guest->setParent(host);
                        } else{
                              cout << "Object tree" << endl;
                              pioneer -> prettyPrint(0);
                              cout << "The head object " + rawdata[0] + " is not
found";
                              break;
                        }
                  } else{
                        if (current_input != "endtree"){
                              if (current_input == "end_of_connections"){
                                    break;
                              } else{
                                    base*    sig    =    findPointerLoc(rawdata[0],
lru_cache);
                                    base*    han    =    findPointerLoc(rawdata[1],
lru_cache);
                                    if (!sig){
                                          cout<<endl;
                                          cout << "Object " + rawdata[0] + " not
found";
                                    }
                                    if (!han){
                                          cout<<endl;
                                          cout << "Handler object " + rawdata[1] +
" not found";
                                    }
                                    if (sig &&  han){
```

```
                                    sig->setBridge(sig->getEmitter(),   han,   han-
>getHandler());
                                    }
                              }
                        }
                  }
            }

      }*/
};
cl_application* cl_application::system = NULL;
dummy_two* cl_application::reader = NULL;
dummy_three* cl_application::control = NULL;
dummy_four* cl_application::square = NULL;
dummy_five* cl_application::robot = NULL;
dummy_six* cl_application::writer = NULL;
int cl_application::exec_app() {
            return 0;
}
      /*
      string current_input;
      string lru_cache = "/";
      while (getline(cin,current_input)){
            stringstream indata;
            indata << current_input;
            vector<string> rawdata = {};
            string input_part;
            string temp = "";
            string quessed = "";
            string message = "";
            while (indata >> input_part){
                  rawdata.push_back(input_part);
            }
            if (rawdata[0] == "END"){
                  break;
            } else if (rawdata[0] == "SET_CONNECT"){
                  base* sig = findPointerLoc(rawdata[1], lru_cache);
                  base* han = findPointerLoc(rawdata[2], lru_cache);
                  if (!sig){
                                    cout<<endl;
                                    cout << "Object " + rawdata[1] + " not
found";
                                    }
                                    if (!han){
                                    cout<<endl;
                                    cout << "Handler object " + rawdata[2] +
" not found";
                                    }
                                    if (sig &&  han){
                  sig->setBridge(sig->getEmitter(), han, han->getHandler());
                                    }
            } else if (rawdata[0] == "DELETE_CONNECT"){
                  base* sig = findPointerLoc(rawdata[1], lru_cache);
                  base* han = findPointerLoc(rawdata[2], lru_cache);
                  if (!sig){
                                    cout<<endl;
                                    cout << "Object " + rawdata[1] + " not
```

```
found";
                                    }
                                    if (!han){
                                          cout<<endl;
                                          cout << "Handler object " + rawdata[2] +
" not found";
                                    }
                                    if (sig &&  han){
                  sig->killBridge(sig->getEmitter(), han, han->getHandler());
                                    }
            } else if (rawdata[0] == "SET_CONDITION"){
                  base* sig = findPointerLoc(rawdata[1], lru_cache);
                  if (!sig){
                        cout<<endl;
                        cout << "Object " + rawdata[1] + " not found";
                  }
                  sig->setWake(stoi(rawdata[2]));
            } else if (rawdata[0] == "EMIT"){
                  quessed = rawdata[0] + " " + rawdata[1] + " ";
                  for (char c : regex_replace(current_input, regex("^ +| +$|( )
+"), "$1")){
                        if (temp != quessed){
                              temp += string(1,c);
                        } else{
                              message += string(1,c);
                        }
                  }
                  base* sig = findPointerLoc(rawdata[1], lru_cache);
                  if (!sig){
                                          cout<<endl;
                                          cout << "Object " + rawdata[1] + " not
found";
                                    }
                                    if (sig){
                  sig->emitGlobal(sig->getEmitter(), message);
                                    }
            }
      }
      //cout<<endl; //Depenends on ambigous requirements.*/
//};
cl_application::cl_application(cl_application*  predok,  string  imya):base(NULL,
imya){
};
```

5.4 Файл cl_application.h

Листинг 4 – cl_application.h

```
#ifndef CL_APPLICATION
#define CL_APPLICATION
#include "base.h"
class cl_application : public base{
      public:
```

```
	cl_application(cl_application* predok, string imya = "System");
	void bild_tree_objects();
	int exec_app();
	static cl_application* system;
	static dummy_two* reader;
	static dummy_three* control;
	static dummy_four* square;
	static dummy_five* robot;
	static dummy_six* writer;
};
#endif
```

5.5 Файл control_object.cpp

Листинг 5 – control_object.cpp

```
#include "control_object.h"
#include "base.h"
#include "cl_application.h"
#include <string>
using namespace std;
int dummy_three::direction = 0;
dummy_three::dummy_three(base* predok, string imya):base(predok, imya){
	this->typenum = 3;
};
HAN_MOVE dummy_three::getMoveSigHan(){
	return USICAST(dummy_five::moveInStaticField);
}
SIG_MOVE control_object::getMoveSigSen(){
	return USICAST(dummy_three::sendSignal);
}
//Changed much...
bool dummy_three::sendSignal(int msg){
	//HAN_MOVE method = this->getMoveSigHan();
	bool temp;
	for (bridge fat : rels){
		if (fat.sig_m == this->getMoveSigSen()){
		temp    =    ((static_cast<robot_object*>(fat.responder)->*(fat.han_m))
(msg));
		}
	}
	return temp;
}
void dummy_three::moveNextInStaticField(){
	bool moved = false;
	SIG_MOVE method = getMoveSigSen();
	switch (this->direction){
		case 0:
			moved = (this->*method)(2);
			break;
		case 1:
			moved = (this->*method)(0);
			break;
```

```
            case 2:
                  moved = (this->*method)(3);
                  break;
            case 3:
                  moved = (this->*method)(1);
                  break;
      }
      if (!moved){
            direction = (direction + 1) % 4;
      }
}
//New.
void control_object::setConnect(SIG_MOVE source, base* target, HAN_MOVE handler){
      bridge fat;
      fat.sig_m = source;
      fat.han_m = handler;
      fat.responder = target;
      fat.bridge_type = 1;
      this->rels.push_back(fat);
}
void  control_object::deleteConnect(SIG_MOVE  source,  base*  target,  HAN_MOVE
handler){
      bridge fat;
      fat.sig_m = source;
      fat.han_m = handler;
      fat.responder = target;
      fat.bridge_type = 1;
      this->rels.erase(remove(this->rels.begin(),  this->rels.end(),  fat),  this-
>rels.end());
}
```

5.6 Файл control_object.h

Листинг 6 – control_object.h

```
#ifndef DUMMY_THREE
#define DUMMY_THREE
#include "base.h"
#include "robot_object.h"
class control_object : public base{
      public:
      dummy_three(base* predok, string imya = "Control");
      static int direction;
      void moveNextInStaticField();
      HAN_MOVE getMoveSigHan();
      SIG_MOVE getMoveSigSen();
      bool sendSignal(int msg);
      //New.
      void setConnect(SIG_MOVE source, base* target, HAN_MOVE handler);
      void deleteConnect(SIG_MOVE source, base* target, HAN_MOVE handler);
};
#endif
```

5.7 Файл field_object.cpp

Листинг 7 – field_object.cpp

```
#include "field_object.h"
#include "robot_object.h"
#include "base.h"
#include <string>
using namespace std;
dummy_four::dummy_four(base* predok, string imya):base(predok, imya){
      this->typenum = 4;
};
vector<vector<position>> field_object::locs = {};
int field_object::isExit(vector<vector<position>> &field, int x, int y){
      if ((x == 0) && (y == 0)) {return 0;};
      if ((x == 0) && (y == 21)) {return 0;};
      if ((x == 21) && (y==0)) {return 0;};
      if ((x == 21) && (y == 21)) {return 0;};
      if (field[x][y].i == 1) {return 0;};
      bool vertical_up = (x -1 == 0) && (y > 1) && (y < 20);
      bool  vertical_down = (x + 1 == 21) && (y > 1) && (y < 20);
      bool  horizontal_right = (y + 1 == 21) && (x > 1) && (x < 20);
      bool horizontal_left = (y - 1 == 0) && (x > 1) && (x < 20);
      if (vertical_up){
            return 1;}
      else if(vertical_down){
            return 2;
      } else if (horizontal_right){
            return 3;}
      else if (horizontal_left){
            return 4;}
      else{
            return 0;
      }
}
//Changed much...
vector<bool> dummy_four::possibles(int x, int y, bool noreverse){
      vector<bool> temp = {};
      if (field_object::locs[x][y].i == 1){
            if (noreverse){
                  temp = {1,1,1,1};
                  for (bridge fat : rels){
            if (fat.han_ld == (&field_object::possibles)){
                  ((static_cast<robot_object*>(fat.responder)->*(fat.han_loc))
(temp));
                  }
            }
                  return {1,1,1,1};}
            else{
                  temp = {0,0,0,0};
                        for (bridge fat : rels){
            if (fat.han_ld == (&field_object::possibles)){
                  ((static_cast<robot_object*>(fat.responder)->*(fat.han_loc))
(temp));
                  }
```

```
            }
                  return {0,0,0,0};
            }
            }
            bool one, two, three, four;
            if (x == 21){
                  one = 0;
            } else {
                  one = (field_object::locs[x+1][y].i == 0);
            }
            if (x == 0){
                  two = 0;
            } else {
                  two = (field_object::locs[x-1][y].i == 0);
            }
            if (y == 21){
                  three = 0;
            } else {
                  three = (field_object::locs[x][y+1].i == 0);
            }
            if (y == 0){
                  four = 0;
            } else {
                  four = (field_object::locs[x][y-1].i == 0);
            }
            if (noreverse){
                  temp = {!one, !two, !three, !four};
                        for (bridge fat : rels){
            if (fat.han_ld == (&field_object::possibles)){
                  ((static_cast<robot_object*>(fat.responder)->*(fat.han_loc))
(temp));
                  }
            }
                  return {!one, !two, !three, !four};
            } else{
                  temp = {one, two, three, four};
                        for (bridge fat : rels){
            if (fat.han_ld == (&field_object::possibles)){
                  ((static_cast<robot_object*>(fat.responder)->*(fat.han_loc))
(temp));
                  }
            }
                  return {one, two, three, four};
            }
      }
void field_object::floodInductive(vector<vector<position>> & field, int x, int y){
      if (x == 0){
            return;}
      if (x == 21){
            return;}
      if (y == 0){
            return;}
      if (y == 21){
            return;}
      if (field[x][y].i == 1){
            return;};
      bool canSinkAbove = (field[x][y+1].p == 1);
```

```
	bool canSinkDown = (field[x][y-1].p == 1);
	bool canSinkLeft = (field[x-1][y].p == 1);
	bool canSinkRight = (field[x+1][y].p == 1);
	if (canSinkAbove || canSinkDown || canSinkLeft || canSinkRight){
		field[x][y].p = 1;
	}
}
void field_object::meltFlood(vector<vector<position>> & field){
	int iteration = 0;
	int overcrit = 22 * 22 * 22;
	while (iteration <= overcrit){
		int x = 0;
		while (x < 22){
			int y = 0;
			while (y < 22){
				floodInductive(field,x,y);
				iteration++;
				y++;
			}
			x++;
}
	}

}
void field_object::demolishAll(vector<vector<position>> & field){
	int x = 0;
	while (x < 22){
		int y = 0;
		while (y < 22){
			field[x][y].p = 0;
			y++;
		}
		x++;
	}
	}
void field_object::prepareAll(vector<vector<position>> & field){
	int x = 0;
	while (x<22){
		int y = 0;
		vector<position> xrow = {};
		field.push_back(xrow);
		x++;
	}
}
void field_object::setupTypes(vector<vector<position>> & field){
	int x  = 0;
	while (x < 22){
		int y = 0;
		while ( y < 22){
			field[x][y].t = field_object::isExit(field,x,y);
			y++;
		}
		x++;
	}
}
vector<int>  field_object::getExitCoordsCorrected(vector<vector<position>>&  field,
int x, int y){
```

```
      if (field[x][y].t == 0){return {-1,-1};};
      if (field[x][y].t == 1){return {x+1,y+1+1};};
      if (field[x][y].t == 2){return {x+1,y-1+1};};
      if (field[x][y].t == 3){return {x+1+1,y+1};};
      if (field[x][y].t == 4){return {x-1+1,y+1};};
      return {-1};
}
vector<int> field_object::getTsunami(vector<vector<position>>&field, int x, int y)
{
      if (y == 0){
            if (field[x][y+1].t > 0){
                  return {x, y + 1};
            }
      }
      if (y == 21){
            if (field[x][y-1].t > 0){
                  return {x, y - 1};
            }
      }
      if (x == 0){
            if (field[x + 1][y].t > 0){
                  return {x + 1, y };
            }
      }
      if (x == 21){
            if (field[x - 1][y].t > 0){
                  return {x - 1, y};
            }
      }
      return {-1};
}
vector<int> field_object::isLabyrinth(vector<vector<position>>&field, int xp, int
yp){
      vector<int> source = getTsunami(field, xp, yp);
      if (source[0] != -1){
            field[source[0]][source[1]].p = 1;
            this->meltFlood(field);
            int x = 0;
            while (x < 22){
                  int y = 0;
                  while (y < 22){
                        if (field[x][y].t > 0 && field[x][y].p == 1){
                              if ((x != xp) && (y != yp)){
                                    switch (field[x][y].t){
                                          case 1:
                                                this->demolishAll(field);
                                                return {xp, yp, x - 1 , y};
                                          case 2:
                                                this->demolishAll(field);
                                                return {xp, yp, x + 1 , y};
                                          case 3:
                                                this->demolishAll(field);
                                                return {xp, yp, x , y + 1};
                                          case 4:
                                                this->demolishAll(field);
                                                return {xp, yp, x , y - 1};
                                    }
```

```
                              }
                        }
                        y++;
                  }
                  x++;
            }
            this->demolishAll(field);
            return {xp, yp};} else{
            return source;
      }
}
//New.
void field_object::setConnect(HAN_LOC_SIG source, base* target, HAN_LOC handler){
      bridge fat;
      fat.han_ld = source;
      fat.han_loc = handler;
      fat.responder = target;
      fat.bridge_type = 4;
      this->rels.push_back(fat);
}
void  field_object::deleteConnect(HAN_LOC_SIG  source,  base*  target,  HAN_LOC
handler){
      bridge fat;
      fat.han_ld = source;
      fat.han_loc = handler;
      fat.responder = target;
      fat.bridge_type = 4;
      this->rels.erase(remove(this->rels.begin(),  this->rels.end(),  fat),  this-
>rels.end());
}
```

5.8 Файл field_object.h

Листинг 8 – field_object.h

```
#ifndef DUMMY_FOUR
#define DUMMY_FOUR
#include "base.h"
class field_object : public base{
      public:
      dummy_four(base* predok, string imya = "Field");
      vector<bool> possibles(int x, int y, bool noreverse = false);
      static vector<vector<position>> locs;
      static int isExit(vector<vector<position>> & field, int x, int y);
      void floodInductive(vector<vector<position>> & field, int x, int y);
      void meltFlood(vector<vector<position>> & field);
      void demolishAll(vector<vector<position>> & field);
      static void prepareAll(vector<vector<position>> & field);
      static void setupTypes(vector<vector<position>> & field);
      vector<int> getExitCoordsCorrected(vector<vector<position>> & field, int x,
int y);
      vector<int> getTsunami(vector<vector<position>> & field, int x, int y);
      vector<int> isLabyrinth(vector<vector<position>> & field, int xp, int yp);
```

```
	//New.
	void setConnect(HAN_LOC_SIG source, base* target, HAN_LOC handler);
	void deleteConnect(HAN_LOC_SIG source, base* target, HAN_LOC handler);
};
#endif
```

5.9 Файл main.cpp

Листинг 9 – main.cpp

```
#include <stdlib.h>
#include <stdio.h>
#include "cl_application.h"
#include "base.h"
#include "control_object.h"
#include "reader_object.h"
#include <iostream>
using namespace std;
// program here
int main()
{
	cl_application ob_cl_application = new cl_application(nullptr);
	ob_cl_application.bild_tree_objects();
	return ob_cl_application.exec_app();
}
```

5.10 Файл printer_object.cpp

Листинг 10 – printer_object.cpp

```
#include "printer_object.h"
#include "base.h"
#include <string>
#include <iostream>
using namespace std;
dummy_six::dummy_six(base* predok, string imya):base(predok, imya){
	this->typenum = 6;
};
void dummy_six::unhold(){
	bool first = true;
	for (string msg : data){
		if (first){
			first = false;}
		else{
			cout << endl;}
		cout << msg;
	}
}
void dummy_six::hold(string msg){
```

```
	this->data.push_back(msg);
}
```

5.11 Файл printer_object.h

Листинг 11 – printer_object.h

```
#ifndef DUMMY_SIX
#define DUMMY_SIX
#include "base.h"
class printer_object : public base{
	public:
	dummy_six(base* predok, string imya = "Writer");
	vector<string> data;
	void unhold();
	void hold(string msg);
};
#endif
```

5.12 Файл reader_object.cpp

Листинг 12 – reader_object.cpp

```
#include "reader_object.h"
#include "base.h"
#include "cl_application.h"
#include "field_object.h"
#include <string>
using namespace std;
dummy_two::dummy_two(base* predok, string imya):base(predok, imya){
	this->typenum = 2;
};
void dummy_two::elementalReader(vector<vector<position>> & field){
	int xrow = 0;
	string dataline;
	while (xrow < 22){
		cin >> dataline;
		(this->*getTextSigSen())(dataline);
		if (dataline != "SHOWTREE"){
			for (char c : dataline){
				position loc;
				loc.i = stoi(string(1,c));
				field[xrow].push_back(loc);
			}
			xrow++;
		}
	}
	field_object::setupTypes(field);
};
```

```
HAN_TEXT dummy_two::getTextSigHan(){
	return USICAST(base::specComHan);
}
SIG_TEXT dummy_two::getTextSigSen(){
	return USICAST(dummy_two::sendSignal);
}
//Changed much..,
void dummy_two::sendSignal(string msg){
	//HAN_TEXT method = this->getTextSigHan();
	//(cl_application::system->*method)(msg);
	for (bridge fat : rels){
		if (fat.call_t == this->getTextSigSen()){
		((static_cast<base*>(fat.responder)->*(fat.han_t))(msg));
		}
	}
}
//New.
void reader_object::setConnect(SIG_TEXT source, base* target, HAN_TEXT handler){
	bridge fat;
	fat.call_t = source;
	fat.han_t = handler;
	fat.responder = target;
	fat.bridge_type = 2;
	this->rels.push_back(fat);
}
void reader_object::deleteConnect(SIG_TEXT source, base* target, HAN_TEXT handler)
{
	bridge fat;
	fat.call_t = source;
	fat.han_t = handler;
	fat.responder = target;
	fat.bridge_type = 2;
	this->rels.erase(remove(this->rels.begin(),  this->rels.end(),  fat),  this-
>rels.end());
}
```

5.13 Файл reader_object.h

Листинг 13 – reader_object.h

```
#ifndef DUMMY_TWO
#define DUMMY_TWO
#include "base.h"
#include <iostream>
class reader_object : public base{
	public:
	dummy_two(base* predok, string imya = "Reader");
	void elementalReader(vector<vector<position>> &field);
	void sendSignal(string pofd);
	HAN_TEXT getTextSigHan();
	SIG_TEXT getTextSigSen();
	//New.
	void setConnect(SIG_TEXT source, base* target, HAN_TEXT handler);
```

```
      void deleteConnect(SIG_TEXT source, base* target, HAN_TEXT handler);
};
#endif
```

5.14 Файл robot_object.cpp

Листинг 14 – robot_object.cpp

```
#include "robot_object.h"
#include "field_object.h"
#include "cl_application.h"
#include "base.h"
#include <string>
using namespace std;
dummy_five::dummy_five(base* predok, string imya):base(predok, imya){
      this->typenum = 5;
};
bool dummy_five::moveInStaticField(int pos){
      SIG_LOC_HAN method = this->getLocSigSen();
      vector<bool> possible = (this->*method)(robx, roby, false);
      if (possible[pos]){
            switch (pos){
                  case 0:
                        this->robx++;
                        break;
                  case 1:
                        this->robx--;
                        break;
                  case 2:
                        this->roby++;
                        break;
                  case 3:
                        this->roby--;
                        break;
            }
                  return true;
            }
      return false;
}
HAN_LOC_SIG dummy_five::getLocSigHan(){
      return USICAST(dummy_four::possibles);
}
SIG_LOC_HAN dummy_five::getLocSigSen(){
      return USICAST(dummy_five::sendSignal);
}
//Changed much...
vector<bool> dummy_five::sendSignal(int x, int y, bool noreverse){
      //HAN_LOC_SIG method = this->getLocSigHan();
      //return (cl_application::square->*method)(x,y,noreverse);
      for (bridge fat : rels){
            if (fat.sig_ld == this->getLocSigSen()){
            ((static_cast<field_object*>(fat.responder)->*(fat.han_ld))(x,      y,
noreverse));
```

```
            }
      }
      return this->last;
}
//New.
void robot_object::handler(vector<bool> response){
      this->last = response;
}
//New.
void  robot_object::setConnect(SIG_LOC_HAN  source,  base*  target,  HAN_LOC_SIG
handler){
      bridge fat;
      fat.sig_ld = source;
      fat.han_ld = handler;
      fat.responder = target;
      fat.bridge_type = 3;
      this->rels.push_back(fat);
}
void  robot_object::deleteConnect(SIG_LOC_HAN  source,  base*  target,  HAN_LOC_SIG
handler){
      bridge fat;
      fat.sig_ld = source;
      fat.han_ld = handler;
      fat.responder = target;
      fat.bridge_type = 3;
      this->rels.erase(remove(this->rels.begin(),  this->rels.end(),  fat),  this-
>rels.end());
}
```

5.15 Файл robot_object.h

Листинг 15 – robot_object.h

```
#ifndef DUMMY_FIVE
#define DUMMY_FIVE
#include "base.h"
class robot_object : public base{
      public:
      dummy_five(base* predok, string imya = "Robot");
      int robx = 0;
      int roby = 1;
      bool moveInStaticField(int pos);
      HAN_LOC_SIG getLocSigHan();
      SIG_LOC_HAN getLocSigSen();
      vector<bool> sendSignal(int x, int y, bool noreverse = false);
      //New.
      void setConnect(SIG_LOC_HAN source, base* target, HAN_LOC_SIG handler);
      void deleteConnect(SIG_LOC_HAN source, base* target, HAN_LOC_SIG handler);
      vector<bool> last = {};
      void handler(vector<bool> response);
};
#endif
```

6 ТЕСТИРОВАНИЕ

Результат тестирования программы представлен в таблице 37.

Таблица 37 – Результат тестирования программы

Входные данные	Ожидаемые выходные данные	Фактические выходные данные
00000000000000000000 01011101111011111111 01000111000000011111 01011111111111100000 01000111111011000011 01111111111000011111 01111111111111111111 01111111111111111111 01111111111111111111 01110111111111111111 01110111111111110110 00000000001111111110 01101111101111111110 01101100000110000000 01111111110000111110 01111111111111100110 01010110111111101111 01010110111111101111 00000000111111111110 01111111111111111110 01111111111111111110 00000000000000000000	There is no way out of the maze (1, 3) There is no way out of the maze (1, 7) There is no way out of the maze (1, 12) There is no way out of the maze (4, 22) Maze (14, 22) (12, 1)	There is no way out of the maze (1, 3) There is no way out of the maze (1, 7) There is no way out of the maze (1, 12) There is no way out of the maze (4, 22) Maze (14, 22) (12, 1)
SHOWTREE 00000000000000000000 01011101111011111111 01000111000000011111 01011111111111100000 01000111111011000011 01111111111000011111 01111111111111111111 01111111111111111111 01110111111111111111 01110111111111110110 00000000001111111110 01101111101111111110 01101100000110000000 01111111110000111110 01111111111111100110 01010110111111101111 01010110111111101111 00000000111111111110 01111111111111111110 01111111111111111110 00000000000000000000	System is ready Reader is ready Robot is ready Control is ready Field is ready Writer is ready There is no way out of the maze (1, 3) There is no way out of the maze (1, 7) There is no way out of the maze (1, 12) There is no way out of the maze (4, 22) Maze (14, 22) (12, 1)	System is ready Reader is ready Robot is ready Control is ready Field is ready Writer is ready There is no way out of the maze (1, 3) There is no way out of the maze (1, 7) There is no way out of the maze (1, 12) There is no way out of the maze (4, 22) Maze (14, 22) (12, 1)
00000000000000000000	There is no way out of the	There is no way out of the

Входные данные	Ожидаемые выходные данные	Фактические выходные данные
0101110111101111111110 0100011100000001111110 0101111111111111100000 0100011111101100001110 0111111111100001111110 0111111111111111111110 0111111111111111111110 0111111111111111111110 0111011111111111111110 0111011111111111110110 0000000000111111110110 0110111110111111110110 0110110000011000000010 0111111111000011110110 0111111111111110000110 0101011011111110111110 0101011011111110111110 0000000011111111111110 0111111111111111111110 0111111111111111111110 0000000000000000000000	maze (1, 3) There is no way out of the maze (1, 7) There is no way out of the maze (1, 12) There is no way out of the maze (4, 22) There is no way out of the maze (19, 1) There is no way out of the maze (12, 1)	maze (1, 3) There is no way out of the maze (1, 7) There is no way out of the maze (1, 12) There is no way out of the maze (4, 22) There is no way out of the maze (19, 1) There is no way out of the maze (12, 1)
0000000000000000000000 0111111111111111111110 0100011100000001111110 0101111111111111100010 0100011111101100001110 0111111111100001111110 0111111111111111111110 0111111111111111111110 0111111111111111111110 0111011111111111111110 0111011111111111110110 0100000000111111110110 0110111110111111110110 0110110000011000000010 0111111111000011110110 0111111111111110000110 0101011011111110111110 0101011011111110111110 0100000011111111111110 0111111111111111111110 0111111111111111111110 0000000000000000000000		There is no way out of the maze (1, 3) There is no way out of the maze (1, 7) There is no way out of the maze (1, 12) There is no way out of the maze (4, 22) Maze (14, 22) (12, 1)
0000000000000000000000 0111111111111111111110 0100011100000001111110 0101111111111111100010 0100011111101100001110 0111111111100001111110 0111111111111111111110 0111111111111111111110 0111111111111111111110 0111011111111111111110 0111011111111111110110	Maze (22, 4) (20, 1)	Maze (22, 4) (20, 1)

Входные данные	Ожидаемые выходные данные	Фактические выходные данные
01000000001111111110110 01101111101111111110110 01101100000110000000010 01111111110000111110110 01111111111111110000110 01010110111111110111110 01010110111111110111110 01110000111111111111110 00001111111111111111110 01101111111111111111110 00000000000000000000000		

ЗАКЛЮЧЕНИЕ

Курсовая работа на основе задания К7 системы "Аврора" выполнена, подготовлен подробный отчёт по соответствующему заданию.

Осуществлено моделирование работы системы особой конструкции, использующей следующие объекты: "система", объект для чтения данных, объект пульта управления, объект, моделирующий поле размещения лабиринтов, "робот", объект для вывода результатов. Реализован весь необходимый фунционал программы, все упомянутые сигналы, команды и обработчики. Корректность решения подтверждена тестами программы, произведёнными системой "Аврора", а также приведёнными непосредственно в тексте отчёта по выполненному заданию.

В ходе выполнения работы освоены навыки объектно-ориентированного программирования и разработки на языке C++ в целом, а также усвоен соответствующий теоретический материал, в частности, на практике применены указатели на методы-члены классов. Получен опыт составления документации программного кода, оформления блок-схем в соответствии с российским государственным стандартом.

В связи с вышеназванным можно сделать вывод о том, что цели курсовой работы работы по предмету "Объектно-ориентированное программирование" достигнуты, её задачи выполнены в полном объёме.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Васильев А.Н. Объектно-ориентированное программирование на C++. Издательство: Наука и Техника. Санкт-Петербург, 2016г. 543 стр.

2. Шилдт Г. C++: базовый курс. 3-е изд. Пер. с англ.. — М.: Вильямс, 2017. — 624 с.

3. Методическое пособие для проведения практических заданий, контрольных и курсовых работ по дисциплине «Объектно-ориентированное программирование» [Электронный ресурс] – URL: https://mirea.aco-avrora.ru/student/files/methodichescoe_posobie_dlya_laboratorny h_rabot_3.pdf (дата обращения 05.05.2021).

4. Приложение к методическому пособию студента по выполнению заданий в рамках курса «Объектно-ориентированное программирование» [Электронный ресурс]. URL: https://mirea.aco-avrora.ru/student/files/Prilozheniye_k_methodichke.pdf (дата обращения 05.05.2021).

5. Видео лекции по курсу «Объектно-ориентированное программирование» [Электронный ресурс]. АСО «Аврора».

6. Антик М.И. Дискретная математика [Электронный ресурс]: Учебное пособие /Антик М.И., Казанцева Л.В. — М.: МИРЭА — Российский технологический университет, 2018 — 1 электрон. опт. диск (CD-ROM).