МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт информационных технологий
Кафедра вычислительной техники

КУРСОВАЯ РАБОТА

По дисциплине «Объектно-ориентированное программирование»
(наименование дисциплины)

Тема курсовой работы К_3 Моделирование работы инженерного арифметического калькулятора
(наименование темы)

Студент группы ИКБО-72-23 Шатохин Богдан Александрович
(учебная группа) *(Фамилия Имя Отчество)* *(подпись студента)*

Руководитель курсовой работы доцент Ингтем Ж.Г.
(Должность, звание, ученая степень) *(подпись руководителя)*

Консультант ст.преп. Данилович Е.С.
(Должность, звание, ученая степень) *(подпись консультанта)*

Работа представлена к защите «31» мая 2024 г.

Допущен к защите «31» мая 2024 г.

Хорошо (4)

Москва 2024 г.

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт информационных технологий
Кафедра вычислительной техники

Утверждаю

Заведующий кафедрой _____
Подпись

Платонова О.В.
ФИО

«21» февраля 2024г.

ЗАДАНИЕ
На выполнение курсовой работы
по дисциплине **«Объектно-ориентированное программирование»**

Студент Шатохин Богдан Александрович Группа ИКБО-72-23

Тема К_3 Моделирование работы инженерного арифметического калькулятора

Исходные данные:

1. Описания исходной иерархии дерева объектов.
2. Описание схемы взаимодействия объектов.
3. Множество команд для управления функционированием моделируемой системы.

Перечень вопросов, подлежащих разработке, и обязательного графического материала:

1. Построение версий программ.
2. Построение и работа с деревом иерархии объектов.
3. Взаимодействия объектов посредством интерфейса сигналов и обработчиков.
4. Блок-схемы алгоритмов.
5. Управление функционированием моделируемой системы

Срок представления к защите курсовой работы: до «31» мая 2024 г.

Задание на курсовую работу выдал _____ (Ингтем Ж.Г.)
Подпись *ФИО консультанта*
«21» февраля 2024 г.

Задание на курсовую работу получил _____ (Шатохин Б.А.)
Подпись *ФИО исполнителя*
«21» февраля 2024 г.

Москва 2024 г.

ОТЗЫВ

на курсовую работу

по дисциплине «Объектно-ориентированное программирование»

Студент **Шатохин Богдан Александрович** группа ИКБО-72-23
(ФИО студента) *(Группа)*

Характеристика курсовой работы

Критерий	Да	Нет	Не полностью
1. Соответствие содержания курсовой работы указанной теме			✓
2. Соответствие курсовой работы заданию	✓		
3. Соответствие рекомендациям по оформлению текста, таблиц, рисунков и пр.	✓		
4. Полнота выполнения всех пунктов задания			✓
5. Логичность и системность содержания курсовой работы	✓		
6. Отсутствие фактических грубых ошибок	✓		

Замечаний: Выполнено 4 задания из 5

Рекомендуемая оценка: хорошо (4)

_____ доцент Ингтем Ж.Г.
(Подпись руководителя) *(ФИО руководителя)*

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Настоящая курсовая работа выполнена в соответствии с требованиями ГОСТ Единой системы программной документации (ЕСПД) [1]. Все этапы решения задач курсовой работы фиксированы, соответствуют требованиям, приведенным в методическом пособии для выполнения практических заданий, контрольных и курсовых работ по дисциплине «Объектно-ориентированное программирование» [2-3] и методике разработки объектно-ориентированных программ [4-6].

Объектно-ориентированное программирвание - это методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определённого класса, а классы в свою очередб образуют иерархию классов. ООП строиться на трёх принципах, трёх "китах" ООП: инкапсуляция, наследовательность, полиморфизм. Данная курсовая работа, в большей своей части, направлена на демонстрацию принципа наследовательности.

1 ПОСТАНОВКА ЗАДАЧИ

Реализовать механизм взаимодействия объектов с использованием сигналов и обработчиков, с передачей вместе сигналом текстового сообщения (строковой переменной).

Для организации взаимосвязи по механизму сигналов и обработчиков в базовый класс добавить три метода:

- установления связи между сигналом текущего объекта и обработчиком целевого объекта;
- удаления (разрыва) связи между сигналом текущего объекта и обработчиком целевого объекта;
- выдачи сигнала от текущего объекта с передачей строковой переменной. Включенный объект может выдать или обработать сигнал.

Методу установки связи передать указатель на метод сигнала текущего объекта, указатель на целевой объект и указатель на метод обработчика целевого объекта.

Методу удаления (разрыва) связи передать указатель на метод сигнала текущего объекта, указатель на целевой объект и указатель на метод обработчика целевого объекта.

Методу выдачи сигнала передать указатель на метод сигнала и строковую переменную. В данном методе реализовать алгоритм:

1. Если текущий объект отключен, то выход, иначе к пункту 2.
2. Вызов метода сигнала с передачей строковой переменной по ссылке.
3. Цикл по всем связям сигнал-обработчик текущего объекта:

    3.1. Если в очередной связи сигнал-обработчик участвует метод сигнала, переданный по параметру, то проверить готовность целевого объекта. Если целевой объект готов, то вызвать метод обработчика

целевого объекта указанной в связи и передать в качестве аргумента строковую переменную по значению.

4. Конец цикла.

Для приведения указателя на метод сигнала и на метод обработчика использовать параметризированное макроопределение препроцессора.

В базовый класс добавить метод определения абсолютной пути до текущего объекта. Этот метод возвращает абсолютный путь текущего объекта.

Состав и иерархия объектов строится посредством ввода исходных данных. Ввод организован как в версии № 3 курсовой работы. Если при построении дерева иерархии возникает ситуация дубляжа имен среди починенных у текущего головного объекта, то новый объект не создается.

Система содержит объекты шести классов с номерами: 1, 2, 3, 4, 5, 6. Классу корневого объекта соответствует номер 1. В каждом производном классе реализовать один метод сигнала и один метод обработчика.

Каждый метод сигнала с новой строки выводит:

```
Signal from «абсолютная координата объекта»
```

Каждый метод сигнала добавляет переданной по параметру строке текста номер класса принадлежности текущего объекта по форме:

```
«пробел»(class: «номер класса»)
```

Каждый метод обработчика с новой строки выводит:

```
Signal to «абсолютная координата объекта»   Text: «переданная строка»
```

Моделировать работу системы, которая выполняет следующие команды с параметрами:

- EMIT «координата объекта» «текст» – выдает сигнал от заданного по координате объекта;
- SET_CONNECT «координата объекта выдающего сигнал» «координата

целевого объекта» – устанавливает связь;

- DELETE_CONNECT «координата объекта выдающего сигнал» «координата целевого объекта» – удаляет связь;
- SET_CONDITION «координата объекта» «значение состояния» – устанавливает состояние объекта.
- END – завершает функционирование системы (выполнение программы).

Реализовать алгоритм работы системы:

- в методе построения системы:
  - построение дерева иерархии объектов согласно вводу;
  - ввод и построение множества связей сигнал-обработчик для заданных пар объектов.
- в методе отработки системы:
  - привести все объекты в состоянии готовности;
  - цикл до признака завершения ввода:
    - ввод наименования объекта и текста сообщения;
    - вызов сигнала заданного объекта и передача в качестве аргумента строковой переменной, содержащей текст сообщения.
  - конец цикла.

Допускаем, что все входные данные вводятся синтаксически корректно. Контроль корректности входных данных можно реализовать для самоконтроля работы программы. Не оговоренные, но необходимые функции и элементы классов добавляются разработчиком.

1.1 Описание входных данных

В методе построения системы.

Множество объектов, их характеристики и расположение на дереве

иерархии. Структура данных для ввода согласно изложенному в версии № 3 курсовой работы.

После ввода состава дерева иерархии построчно вводится:

```
«координата объекта выдающего сигнал» «координата целевого объекта»
```

Ввод информации для построения связей завершается строкой, которая содержит:

```
«end_of_connections»
```

В методе запуска (отработки) системы построчно вводятся множество команд в производном порядке:

- EMIT «координата объекта» «текст» – выдать сигнал от заданного по координате объекта;
- SET_CONNECT «координата объекта выдающего сигнал» «координата целевого объекта» – установка связи;
- DELETE_CONNECT «координата объекта выдающего сигнал» «координата целевого объекта» – удаление связи;
- SET_CONDITION «координата объекта» «значение состояния» – установка состояния объекта.
- END – завершить функционирование системы (выполнение программы).

Команда END присутствует обязательно.

Если координата объекта задана некорректно, то соответствующая операция не выполняется и с новой строки выдается сообщение об ошибке.

Если не найден объект по координате:

```
Object «координата объекта» not found
```

Если не найден целевой объект по координате:

```
Handler object «координата целевого объекта» not found
```

Пример ввода:

```
appls_root
/ object_s1 3
/ object_s2 2
/object_s2 object_s4 4
/ object_s13 5
/object_s2 object_s6 6
/object_s1 object_s7 2
endtree
/object_s2/object_s4 /object_s2/object_s6
/object_s2 /object_s1/object_s7
/ /object_s2/object_s4
/object_s2/object_s4 /
end_of_connections
EMIT /object_s2/object_s4 Send message 1
EMIT /object_s2/object_s4 Send message 2
EMIT /object_s2/object_s4 Send message 3
EMIT /object_s1 Send message 4
END
```

1.2 Описание выходных данных

Первая строка:

```
Object tree
```

Со второй строки вывести иерархию построенного дерева.

Далее, построчно, если отработал метод сигнала:

```
Signal from «абсолютная координата объекта»
```

Если отработал метод обработчика:

```
Signal to «абсолютная координата объекта»   Text: «переданная строка»
```

Пример вывода:

```
Object tree
appls_root
    object_s1
        object_s7
    object_s2
        object_s4
        object_s6
    object_s13
Signal from /object_s2/object_s4
Signal to /object_s2/object_s6 Text:  Send message 1 (class: 4)
Signal to / Text:  Send message 1 (class: 4)
Signal from /object_s2/object_s4
```

```
Signal to /object_s2/object_s6 Text:  Send message 2 (class: 4)
Signal to / Text:  Send message 2 (class: 4)
Signal from /object_s2/object_s4
Signal to /object_s2/object_s6 Text:  Send message 3 (class: 4)
Signal to / Text:  Send message 3 (class: 4)
Signal from /object_s1
```

2 МЕТОД РЕШЕНИЯ

Для решения задачи используется:

- объекты структуры str_connect, количество задаётся пользователем;
- структура o_sh.

Класс papa_class:

- свойства/поля:
    - поле поле номер класса:
        - наименование — class_id;
        - тип — int;
        - модификатор доступа — private;
    - поле вектор объектов структуры str_connect:
        - наименование — connections;
        - тип — vector <o_sh*>;
        - модификатор доступа — private;
- функционал:
    - метод set_connections — метод установления связи между сигналом текущего объекта и обработчиком;
    - метод emit_signal — метод выдачи сигнала от текущего объекта с передачей стрококвой переменной;
    - метод delete_connect — метод удаления (разрыва) связи между сигналом текущего объекта и обработчиком целивого объекта;
    - метод get_absolute_path — метод получения абсолютного пути объекта;
    - метод get_class_id — метод получения номера класса.

Класс mama_class:

- функционал:

- метод signal_f — метод сигнала 1 класса;
- метод handler_f — метод обработчика 1 класса;
- метод get_method — метод получения указателя на метод сигнала класса по номеру;
- метод get_handler — метод получения указателя на метод обработчика класса по номеру;
- метод build_tree — метод построения исходного дерева иерархии объектов;
- метод exec_app — метод запуск приложения.

Класс wine:

- функционал:
  - метод signal_f — метод сигнала 2 класса;
  - метод handler_f — метод обработчика 2 класса.

Класс wine2:

- функционал:
  - метод signal_f — метод сигнала 3 класса;
  - метод handler_f — метод обработчика 3 класса.

Класс wine3:

- функционал:
  - метод signal_f — метод сигнала 4 класса;
  - метод handler_f — метод сигнала 4 класса.

Класс wine4:

- функционал:
  - метод signal_f — метод сигнала 5 класса;
  - метод handler_f — метод сигнала 5 класса.

Класс wine5:

- функционал:

- метод signal_f — метод сигнала 6 класса;
- метод handler_f — метод сигнала 6 класса.

Таблица 1 – Иерархия наследования классов

№	Имя класса	Классы-наследники	Модификатор доступа при наследовании	Описание	Номер
1	papa_class			базовый класс, содержащий основное методы для постройки дерева иерархии	
		mama_class	public		2
		wine	public		3
		wine2	public		4
		wine3	public		5
		wine4	public		6
		wine5	public		7
2	mama_class			наследуемый класс от базового, необходимый для запуска приложения	
3	wine			вспомогательный класс, необходимый для запуска приложения	
4	wine2			вспомогательный класс, необходимый для запуска приложения	
5	wine3			вспомогательный класс, необходимый для запуска приложения	
6	wine4			вспомогательный класс, необходимый для запуска приложения	
7	wine5			вспомогательный класс, необходимый для запуска приложения	

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 set_connections класса papa_class

Функционал: метод установления связи между сигналом текущего объекта и обработчиком.

Параметры: TYPE_SIGNAL signal - указатель на метод сигнала; papa_class* p_targert - указатель на целевой объект; TYPE_HANDLER p_handler - указатель на метод обработчика.

Возвращаемое значение: нет.

Алгоритм метода представлен в таблице 2.

Таблица 2 – Алгоритм метода set_connections класса papa_class

№	Предикат	Действия	№ перехода
1		объявление указателя p_value на объект структуры o_sh	2
2		инициализация целочисленной переменной i значением 0	3
3	i < колличество элементов connect текущего объекта		4
			6
4	поле p_signal i-го объекта вектора connects текущего объекта равно параметру signal и поле p_target i-го объекта вектора connects		∅

№	Предикат	Действия	№ перехода
	текущего объекта равно параметру p_obj и поле p_handler i-го объекта вектора connects текущего объекта равно параметру p_handler текущего объекта		
			5
5		увелчение значения i на 1	6
6		создание нового объекта структуры str_connect при помози функции new и присвоение указателя p_value на адресс этого объекта	7
7		присвоение пол. p_signal значение параметра signal объекту по указателю p_value	8
8		присвоение пол. p_target значение параметра p_target объекту по указателю p_value	9
9		присвоение пол. p_handler значение параметра p_handler объекту по указателю p_value	10
10		добавление элемнета p_value в вектор connects текущего объекта при помощи метода push_back	∅

3.2 Алгоритм метода emit_signal класса papa_class

Функционал: метод выдачи сигнала от текущего объекта с передачей стрококвой переменной.

Параметры: TYPE_SIGNAL signal - указатель на метод сигнала; string& s_command - передаваемое сообщение.

Возвращаемое значение: нет.

Алгоритм метода представлен в таблице 3.

Таблица 3 – Алгоритм метода emit_signal класса papa_class

№	Предикат	Действия	№ перехода
1		вызов метода сигнала по указателю p_signal от текущего объекта с параметорм message	2
2		инициализация переменной i типа int = 0	3
3	i < размера вектора connects для текущего объекта		4
			∅
4	поле p_signal i-го объекта вектора connects текущего объекта равно параметру siganl		5
			8
5		присвоение указателю p_target значение поля p_target i-го объекта вектора connects текущего объекта	6
6		присвоение указателю handler значение поля p_handler i-го объекта вектора connects текущего объекта	7
7		вызов метода обрабочтика по указателю p_handler от текущего объекта с параметром message	8
8		увеличение значения переменной i на 1	3

3.3 Алгоритм метода delete_connect класса papa_class

Функционал: метод удаления (разрыва) связи между сигналом текущего объекта и обработчиком целивого объекта.

Параметры: TYPE_SIGNAL signal - указатель на метод сигнала; papa_class* p_target - указатель на целевой объект; TYPE_HANDLER p_handler - указатель на метод обработчика.

Возвращаемое значение: нет.

Алгоритм метода представлен в таблице 4.

Таблица 4 – Алгоритм метода delete_connect класса papa_class

№	Предикат	Действия	№ перехода
1		инициализация целочисленной переменной i значением 0	2
2	i < колличества в векторе connects текущего объекта		3
			∅
3	поле p_signal i-го объекта вектора connects текущего объекта равно параметру signal и поле p_target i-го объекта вектора connects текущего объекта равно параметру p_obj и поле p_handler i-го объекта вектора connects текущего объекта равно параметру p_handler	удаление i-го элемента из поля вектора connects текущего объекта	∅
			4
4		увеличение значения переменной i на 1	2

3.4 Алгоритм метода get_absolute_path класса papa_class

Функционал: метод получения абсолютного пути объекта.

Параметры: нет.

Возвращаемое значение: string - абсолютный путь.

Алгоритм метода представлен в таблице 5.

Таблица 5 – Алгоритм метода get_absolute_path класса papa_class

№	Предикат	Действия	№ перехода
1		объявление переменной строковой типа path	2
2		инициализация указателя p_obj на объект класса papa_class и приравнивание значение текущего объекта	3
3	указатель на головной объект, объекта по указателю p_obj не равен nullptr		4
			6
4		присваивание path "/" + имя объекта по указателю p_obj + path	5
5		присваивание указателю p_obj указателя на головнйо объект объекта по укахателю p_obj	3
6	path равен ""	присваивание переменной path значения "/"	7
			7
7		возврат path	∅

3.5 Алгоритм метода get_class_id класса papa_class

Функционал: метод получения номера класса.

Параметры: нет.

Возвращаемое значение: нет.

Алгоритм метода представлен в таблице 6.

Таблица 6 – Алгоритм метода get_class_id класса papa_class

№	Предикат	Действия	№ перехода
1		возврат значения поля class_id у текущего объекта	∅

3.6 Алгоритм метода signal_f класса mama_class

Функционал: метод сигнала 1 класса.

Параметры: string text - текст сигнала.

Возвращаемое значение: нет.

Алгоритм метода представлен в таблице 7.

Таблица 7 – Алгоритм метода signal_f класса mama_class

№	Предикат	Действия	№ перехода
1		присваивания параметру text: text + (class: 1)	2
2		вывод символа перехода на новую строку, "Signal from " и вывод возвращенного значения от вызова метода get_absolute_path от текущего объекта	∅

3.7 Алгоритм метода handler_f класса mama_class

Функционал: метод обработчика 1 класса.

Параметры: string text - сообщение обработчику.

Возвращаемое значение: нет.

Алгоритм метода представлен в таблице 8.

Таблица 8 – Алгоритм метода handler_f класса mama_class

№	Предикат	Действия	№ перехода
1		вывод символа на новую строку, "Signal to ", вывод возвращенного значения от вызова метода get_absolute_path от текущего объекта, "Text: ", text	∅

3.8 Алгоритм метода get_method класса mama_class

Функционал: метод получения указателя на метод сигнала класса по номеру.

Параметры: int id - номер класса.

Возвращаемое значение: TYPE_SIGNAL - тип указателя сигнала.

Алгоритм метода представлен в таблице 9.

Таблица 9 – Алгоритм метода get_method класса mama_class

№	Предикат	Действия	№ перехода
1	id = 2	возврат указателя на метод сигнала класса wine преобразованный к типу сигнала при помощи SIGNAL_D	∅
	id = 3	возврат указателя на метод сигнала класса wine2 преобразованный к типу сигнала при помощи SIGNAL_D	∅
	id = 4	возвращаем указатель на метод сигнала класса wine3 преобразованный к типу сигнала при помощи SIGNAL_D	∅
	id = 5	возвращаем указатель на метод сигнала класса wine4 преобразованный к типу сигнала при помощи SIGNAL_D	∅
	id = 6	возвращаем указатель на метод сигнала класса wine5 преобразованный к типу сигнала при помощи SIGNAL_D	∅
		возвращаем указатель на метод сигнала класса mama_class преобразованный к типу сигнала при помощи SIGNAL_D	∅

3.9 Алгоритм метода get_handler класса mama_class

Функционал: метод получения указателя на метод обработчика класса по номеру.

Параметры: TYPE_HANDLER - тип указателя обработчика.

Возвращаемое значение: int id - номера класса.

Алгоритм метода представлен в таблице 10.

Таблица 10 – Алгоритм метода get_handler класса mama_class

№	Предикат	Действия	№ перехода
1	id = 2	возвращаем указатель на метод обработчика класса wine преобразованный к типу обработчика при помощи SIGNAL_D	∅
	id = 3	возвращаем указатель на метод обработчика класса wine2 преобразованный к типу обработчика при помощи SIGNAL_D	∅
	id = 4	возвращаем указатель на метод обработчика класса wine3 преобразованный к типу обработчика при помощи SIGNAL_D	∅
	id = 5	возвращаем указатель на метод обработчика класса wine4 преобразованный к типу обработчика при помощи SIGNAL_D	∅
	id = 6	возвращаем указатель на метод обработчика класса wine5 преобразованный к типу обработчика при помощи SIGNAL_D	∅
		возвращаем указатель на метод обработчика класса mama_class преобразованный к типу обработчика при помощи SIGNAL_D	∅

3.10 Алгоритм метода build_tree класса mama_class

Функционал: метод построения исходного дерева иерархии объектов.

Параметры: нет.

Возвращаемое значение: нет.

Алгоритм метода представлен в таблице 11.

Таблица 11 – Алгоритм метода build_tree класса mama_class

№	Предикат	Действия	№ перехода
1		объявление целочисленной переменной i_state	2
2		инициализация указателя p_heag на объект класса papa_class равному адресу текущего объекта	3
3		инициализация указателя ob на объект класса papa_class	4
4		объявлнеие переменных стрококвого типа path, to_path, from_path и object_name	5
5		ввод значаения object_name	6
6		вызов метода set_name от текущего объекта	7
7		ввод значений в переменную path	8
8	path равен "endtree"		17
		ввод значения в переменную object_name и i_state	9
9	в object_name содержиться "/" и "."		7
		присвоение указателю p_heag значения вызова метода coord_find от объекта по указателю p_obj с параметром path	10
10	p_heag не равен 0		11
			13
11	у объекта по адрессу p_heag есть подчинённый объект, поле которого равно object_name		7
			12
12	i_class = 2	при помощи оператора функции new создаётся новый объект класса wine с использованием параметризированного конструктора с параметром p_heag и object_name	6

№	Предикат	Действия	№ перехода
	i_class = 3	при помощи оператора функции new создаётся новый объект класса wine2 с использованием параметризированного конструктора с параметром p_heag и object_name	6
	i_class = 4	при помощи оператора функции new создаётся новый объект класса wine3 с использованием параметризированного конструктора с параметром p_heag и object_name	6
	i_class = 5	при помощи оператора функции new создаётся новый объект класса wine4 с использованием параметризированного конструктора с параметром p_heag и object_name	6
	i _class = 6	при помощи оператора функции new создаётся новый объект класса wine5 с использованием параметризированного конструктора с параметром p_heag и object_name	6
			6
13		вывод "Object_tree"	14
14		вызов метода print_tree от текущего объекта	15
15		вызов символа перехода на новую строку, "The head object", path, " not found"	16
16		выход из всей программы при помощи функции exit с параметром 1	∅
17		ввод значений в переменную from_path	18
18	from_path равен "end_of_connections"		∅
		ввод значений в переменную to_path	19
19		присваивание указателя p_heag значений вызова метода find_using_path от объекта по указателю	20

№	Предикат	Действия	№ перехода
		p_heag с параметром from_path	
20		вызов метода set_connect от объекта по указателю p_heag с параметрами указателя на метод сигнала объекта по указаетлю p_heag, указателя на объект по пути to_path и с указателем на метод обработчика объекта по пути to_path	17

3.11 Алгоритм метода signal_f класса wine

Функционал: метод сигнала 2 класса.

Параметры: string text - текст сигнала .

Возвращаемое значение: нет.

Алгоритм метода представлен в таблице 12.

Таблица 12 – Алгоритм метода signal_f класса wine

№	Предикат	Действия	№ перехода
1		присваивание параметру text : text + (class: 2)	2
2		вывод символа перехода на новую строку, "Signal from " и вывод возвращенного значения метода get_absolute_path от текущего объекта	∅

3.12 Алгоритм метода handler_f класса wine

Функционал: метод обработчика 2 класса.

Параметры: string text - сообщение обработчику.

Возвращаемое значение: нет.

Алгоритм метода представлен в таблице 13.

Таблица 13 – Алгоритм метода handler_f класса wine

№	Предикат	Действия	№ перехода
1		вывод сообщения "Signal to ", вывод результата метода get_absolute_path от текущего объекта и "Text"	∅

3.13 Алгоритм метода signal_f класса wine2

Функционал: метод сигнала 3 класса.

Параметры: string text - текст сигнала .

Возвращаемое значение: нет.

Алгоритм метода представлен в таблице 14.

Таблица 14 – Алгоритм метода signal_f класса wine2

№	Предикат	Действия	№ перехода
1		присваивание параметру text : text + (class: 3)	2
2		вывод символа перехода на новую строку, "Signal from " и вывод возвращенного значения метода get_absolute_path от текущего объекта	∅

3.14 Алгоритм метода handler_f класса wine2

Функционал: метод обработчика 3 класса.

Параметры: string text - сообщение обработчику.

Возвращаемое значение: нет.

Алгоритм метода представлен в таблице 15.

Таблица 15 – Алгоритм метода handler_f класса wine2

№	Предикат	Действия	№ перехода
1		вывод сообщения "Signal to ", вывод результата метода get_absolute_path от текущего объекта и "Text"	∅

3.15 Алгоритм метода signal_f класса wine3

Функционал: метод сигнала 4 класса.

Параметры: нет.

Возвращаемое значение: string text - текст сигнала.

Алгоритм метода представлен в таблице 16.

Таблица 16 – Алгоритм метода signal_f класса wine3

№	Предикат	Действия	№ перехода
1		присваивание параметру text : text + (class: 4)	2
2		вывод символа перехода на новую строку, "Signal from " и вывод возвращенного значения метода get_absolute_path от текущего объекта	∅

3.16 Алгоритм метода handler_f класса wine3

Функционал: метод сигнала 4 класса.

Параметры: string text - сообщение обработчику.

Возвращаемое значение: нет.

Алгоритм метода представлен в таблице 17.

Таблица 17 – Алгоритм метода handler_f класса wine3

№	Предикат	Действия	№ перехода
1		вывод сообщения "Signal to ", вывод результата метода get_absolute_path от текущего объекта и "Text"	∅

3.17 Алгоритм метода signal_f класса wine4

Функционал: метод сигнала 5 класса.

Параметры: string text - текст сигнала .

Возвращаемое значение: нет.

Алгоритм метода представлен в таблице 18.

Таблица 18 – Алгоритм метода signal_f класса wine4

№	Предикат	Действия	№ перехода
1		присваивание параметру text : text + (class: 5)	2
2		вывод символа перехода на новую строку, "Signal from " и вывод возвращенного значения метода get_absolute_path от текущего объекта	∅

3.18 Алгоритм метода handler_f класса wine4

Функционал: метод сигнала 5 класса.

Параметры: string text - сообщение обработчику.

Возвращаемое значение: нет.

Алгоритм метода представлен в таблице 19.

Таблица 19 – Алгоритм метода handler_f класса wine4

№	Предикат	Действия	№ перехода
1		вывод сообщения "Signal to ", вывод результата метода get_absolute_path от текущего объекта и "Text"	∅

3.19 Алгоритм метода signal_f класса wine5

Функционал: метод сигнала 6 класса.

Параметры: string text - текст сигнала .

Возвращаемое значение: нет.

Алгоритм метода представлен в таблице 20.

Таблица 20 – Алгоритм метода signal_f класса wine5

№	Предикат	Действия	№ перехода
1		присваивание параметру text : text + (class: 6)	∅

3.20 Алгоритм метода handler_f класса wine5

Функционал: метод сигнала 6 класса.

Параметры: string text - сообщение обработчику.

Возвращаемое значение: нет.

Алгоритм метода представлен в таблице 21.

Таблица 21 – Алгоритм метода handler_f класса wine5

№	Предикат	Действия	№ перехода
1		вывод сообщения "Signal to ", вывод результата метода get_absolute_path от текущего объекта и "Text"	∅

3.21 Алгоритм метода exec_app класса mama_class

Функционал: метод запуск приложения.

Параметры: нет.

Возвращаемое значение: int - индикатор успешности выполнения прогаммы.

Алгоритм метода представлен в таблице 22.

Таблица 22 – Алгоритм метода exec_app класса mama_class

№	Предикат	Действия	№ перехода
1		объявление целочисленной переменной status	2
2		инициализация указателя p_obj на объект класса papa_class и приравнивание ему адреса текущего объекта	3
3		объявление указателя path_obj на объект класса	4

№	Предикат	Действия	№ перехода
		papa_class	
4		объявление строковых переменных command, path, delname, text, from_path, to_path	5
5		вывод "Object tree"	6
6		вывод метода print_tree от текущего объекта	7
7		ввод значений в переменную command	8
8	значение переменной command = "END"		39
			9
9	значение переменной command равно "SET"	ввод значения переменной path	10
			12
10	p_obj не равно нулю	p_obj = значение вызова метода coord_find от объекта по адресу p_obj с параметром path	11
		вывод символа перехода на новую строчку "The object was not found at the specified coordinate: " path	7
11		вывод символа перехода на новую строчку "Object is set " поле имени объекта по указателю p_obj	7
12	значаение переменной command равно "MOVE"	ввод значения в переменную path	13
			17
13	p_obj равен nullptr	вывод символа перехода на новую строку path " Head object is not found"	7
			14
14	значение вызова метода coord_find от текузего объекта не равен нулю	вывод символа перехода на новую строчку path "Redefining the head object failed"	7
			15

№	Предикат	Действия	№ перехода
15	головной объект с координатой path имеет подчинеённый объект поле имя которого равно полю имени объекта по указателю p_obj	вывод символа перехода на новую строку path " Dubbing the names of subordinate objects"	7
			16
16	значение вызова метода change_papa от объекта по указателю p_obj с параметорм значения вызова метода coord_find от объекта по указателю p_obj = true	вывод символа перехода на новую строку "New head object" поле имя объекта по указателю значения вызова метода coord_find от объекта по указателю p_obj	7
		вывод символа перехода на новую строчку path " Redefining the head object failed"	7
17	значение переменной command = "FIND"	ввод значения в переменную path	18
			19
18	значение вызова метода coord_find от текущего объекта по адресу p_obj не равен нулю	вывод символа перехода на новую строку path "Object name: " поле имя объекта по указателю p_obj значение вызова метода coord_find от объекта по указателю p_obj	7
		вывод символа перехода на новую строку path " Object is not found"	7
19	значение переменной command = "DELETE"	ввод значения в переменную path	20
			22
20	объект по указателю p_obj в подчинённых объектах имеет такой, что его имя =	вывод символа перехода на новую строку "The object" path "/" delname "has been deleted"	21

№	Предикат	Действия	№ перехода
	delname		
			7
21		вызов метода delete_object от объекта по указателю p_obj	7
22	значение переменной command равно "EMIT"	ввод значения в переменную path	23
			26
23		ввод всей строки в переменную text	24
24		path_obj приравнивается значение вызова метода coord_find от объекта по адрессу path_obj	25
25	path_obj равен 0	вывод символа перехода на новую строку "Object " path " not found"	7
		вызов метода emit_signal от объекта по указателю path_object с параметрами указателя на метод сигнал объекта по указателю path_obj	7
26	значение переменной s_command равно "SET_CONNECT"	ввод значения в переменную from_path	27
			31
27		ввод значения в переменную to_path	28
28		path_obj приравнивается значение вызова метода coord_find от объекта по адресу path_obj с параметром path	29
29	значение вызова метода coord_find от объекта по адресу p_obj с параметром from_path равно nullptr	вывод символа перехода на новую строку "Object " from path "not found"	7
			30
30	значение вызова метода coord_find от объекта по	вывод символа перехода на новую строчку	7

№	Предикат	Действия	№ перехода
	адресу p_obj с параметром to_path равно nullptr	"Handler object " from_path "not found"	
		вызов метода set_connection от объекта по указателю path_obj с параметрами указателя на метод сигнала объекта по указателю path_obj, указателя на объект и пути to_path и с указателем на метод обработчик объекта по пути to_path	7
31	значение переменной command равно "DELETE_CONNECTION"	вывод значений в переменную from_path	32
			36
32		ввод значений в переменную to_path	33
33		path_obj приравнивается значение вызовас метода coord_find от объекта по адресу path_obj с параметром path	34
34	значение вызова метода coord_find от объекта по адресу p_obj с параметром from_path равно nullptr	вывод символа перехода на новую строку "Object " from_path "not found"	7
			35
35	значение вызова метода coord_find от объекта по адресу p_obj с параметром to_path равно nullptr	вывод символа перехода на новую строку "Handler object " from_path "not_found"	7
		вызов метода delete_connect от объекта по указателю path_obj с параметроми указателя на метода сигнала объекта по указателю path_obj, указателя на объект по пути to_path и с указателем на метод обработчик объекта по пути to_path	7
36	значение переменной	ввод значений в переменную path	37

№	Предикат	Действия	№ перехода
	command равно "SET_CONDITION"		
			∅
37		ввод значений в переменную status	38
38	значение вызова метода coord_find от объекта по адресу p_obj с параметром path равно nullptr	вывод символа перехода на новую строку "Object " from_path " not found"	7
		вызов метода set_status с параметром RON от объекта по указателю значения вызова метода coord_find от объекта по адресу p_obj с параметром path	7
39		возврат 0	∅

4 БЛОК-СХЕМЫ АЛГОРИТМОВ

Представим описание алгоритмов в графическом виде на рисунках 1-26.

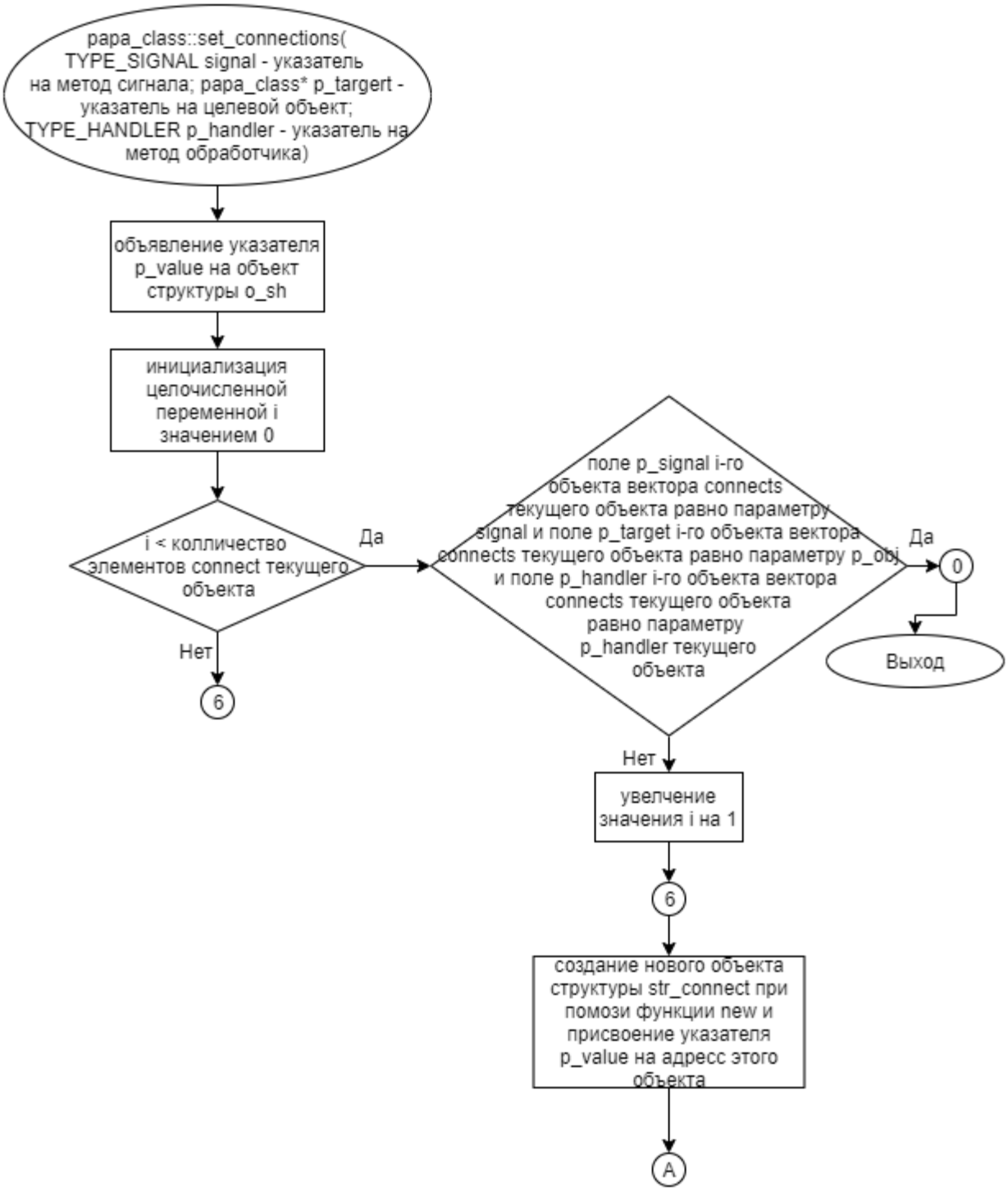

Рисунок 1 – Блок-схема алгоритма

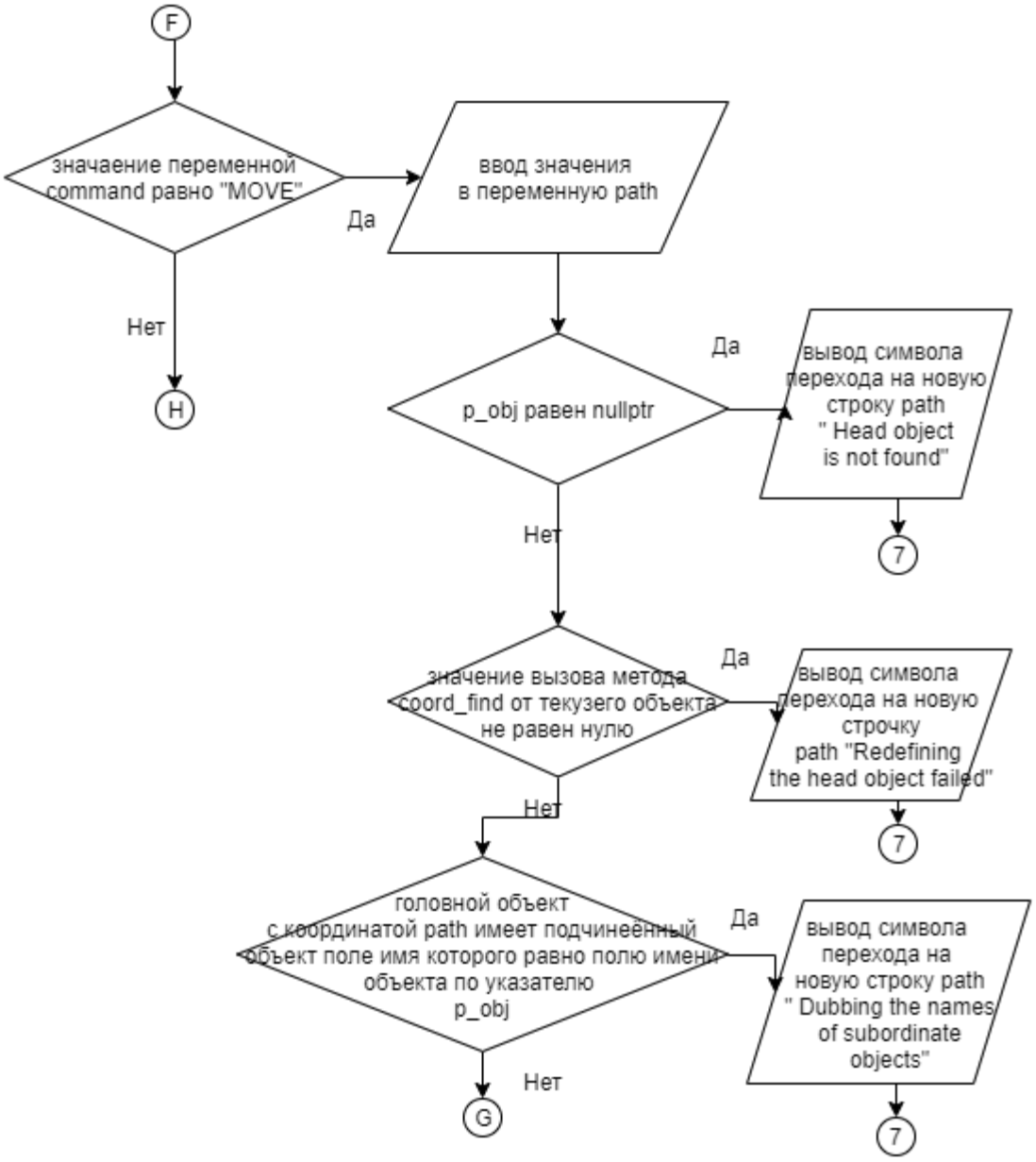

Рисунок 2 – Блок-схема алгоритма

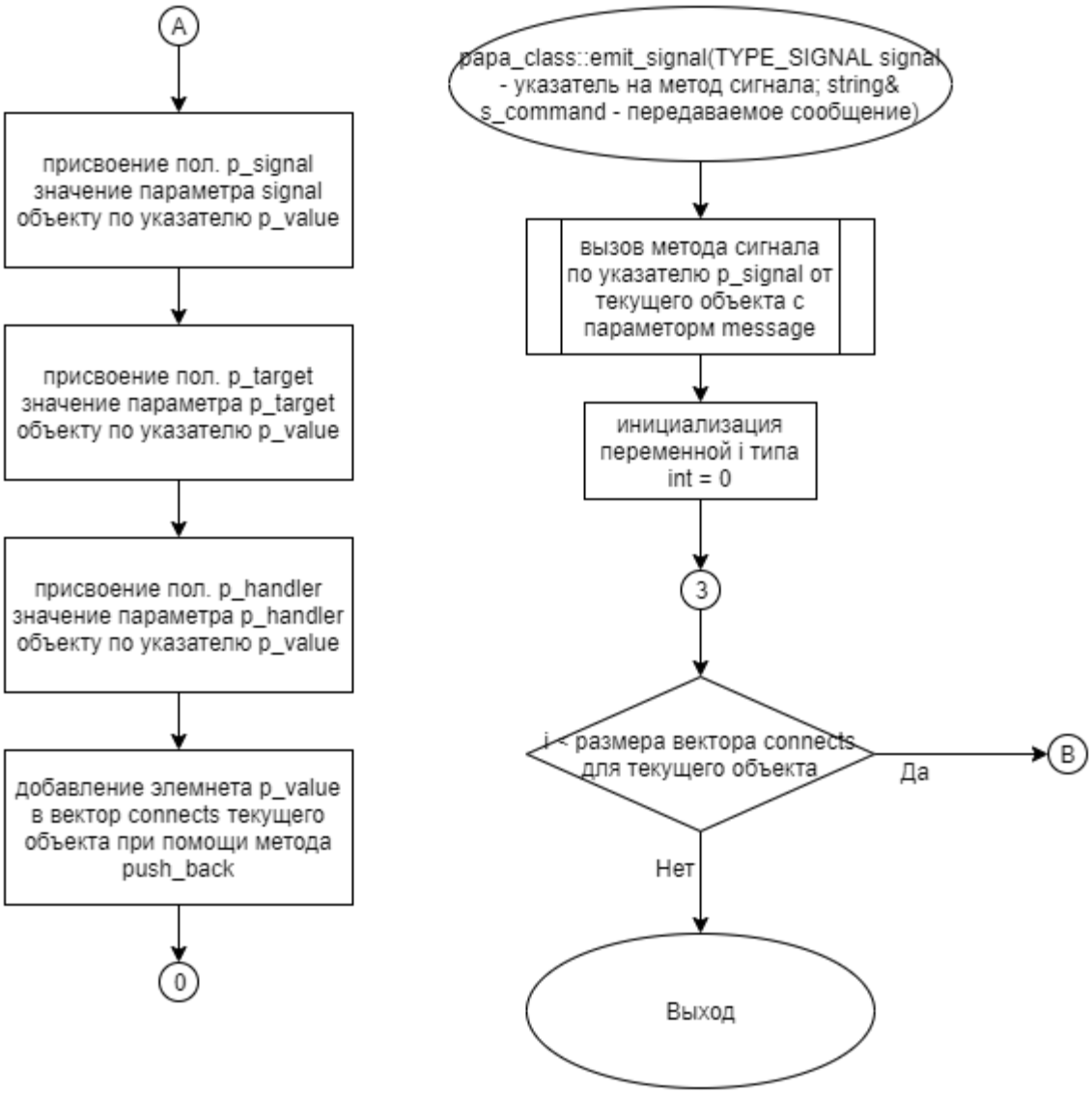

Рисунок 3 – Блок-схема алгоритма

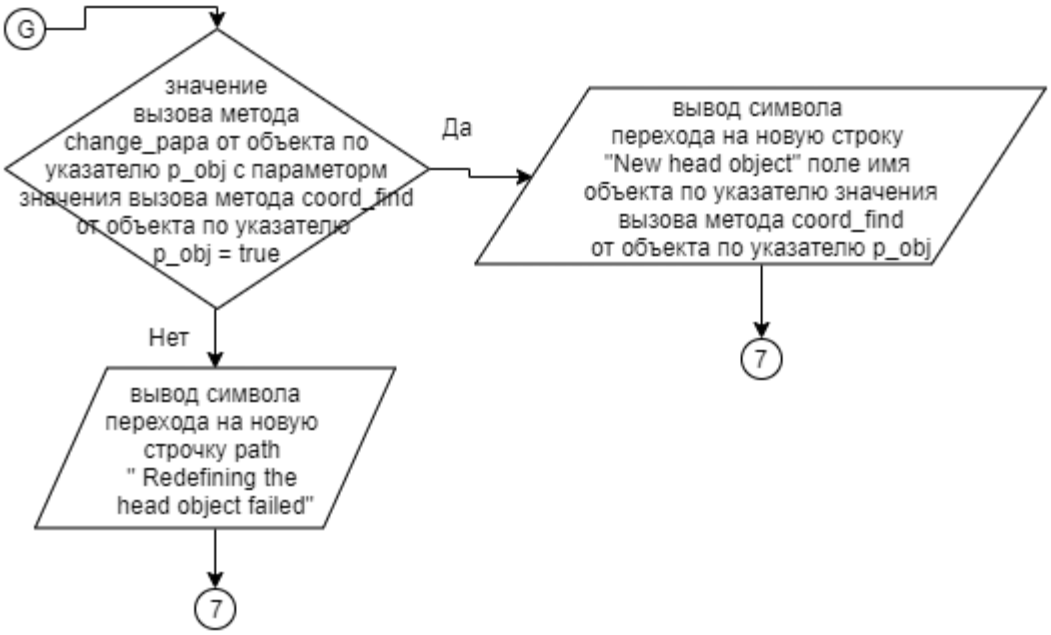

Рисунок 4 – Блок-схема алгоритма

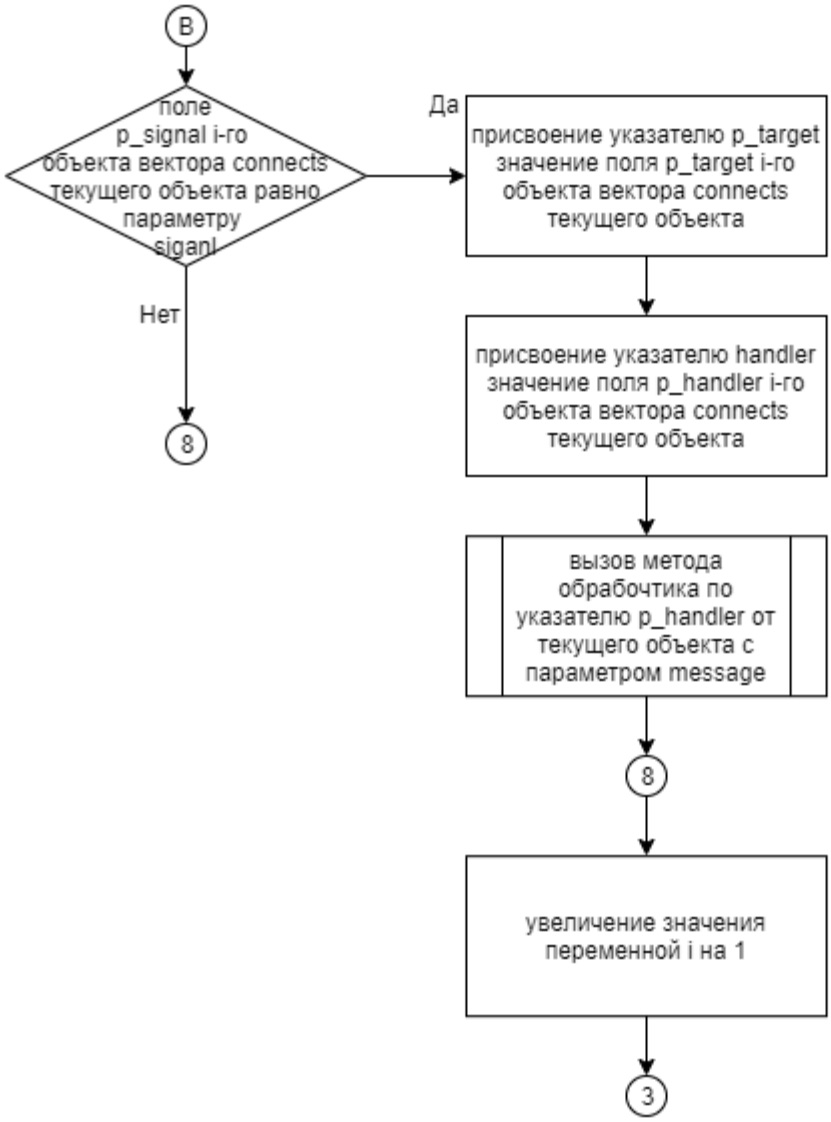

Рисунок 5 – Блок-схема алгоритма

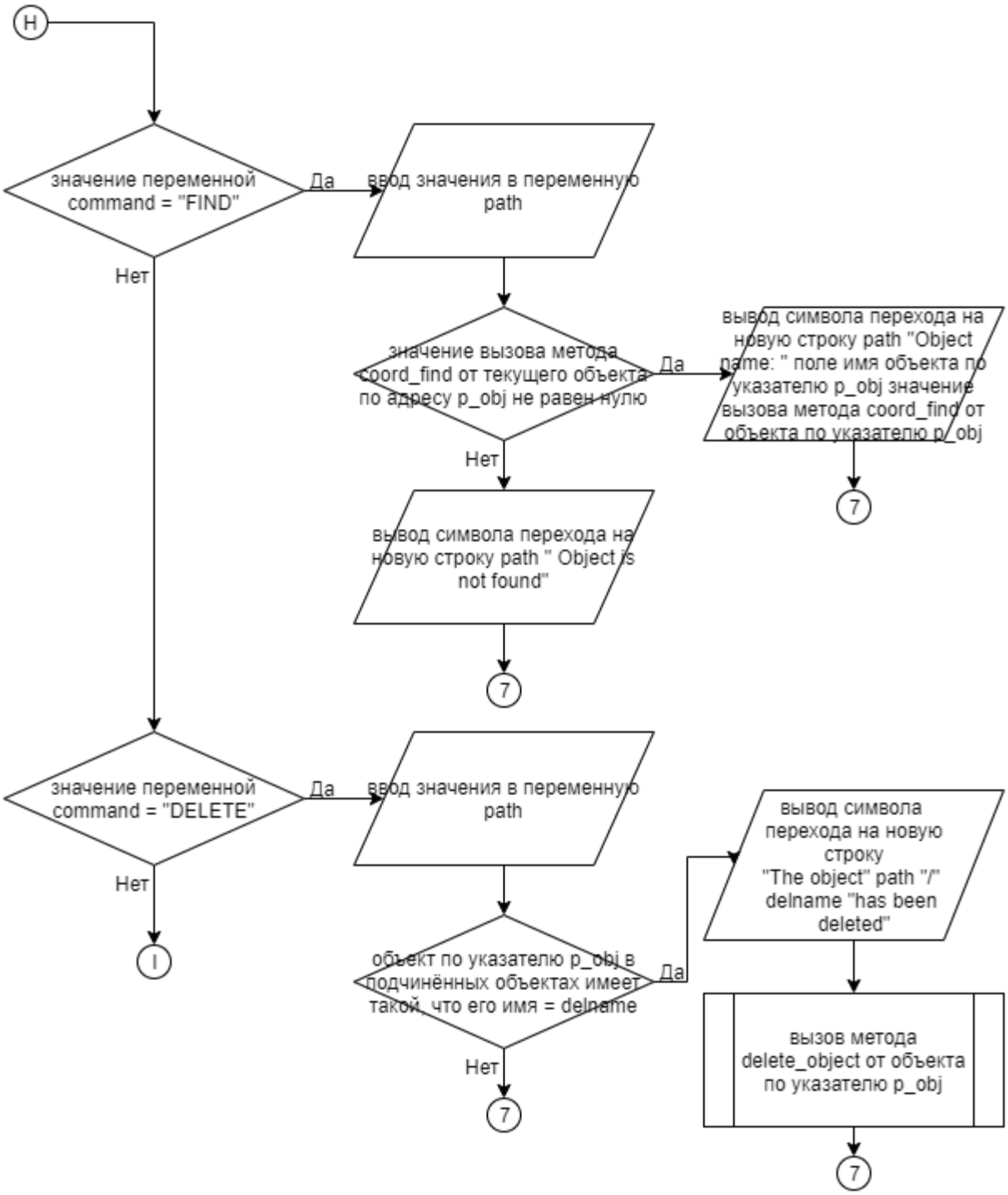

Рисунок 6 – Блок-схема алгоритма

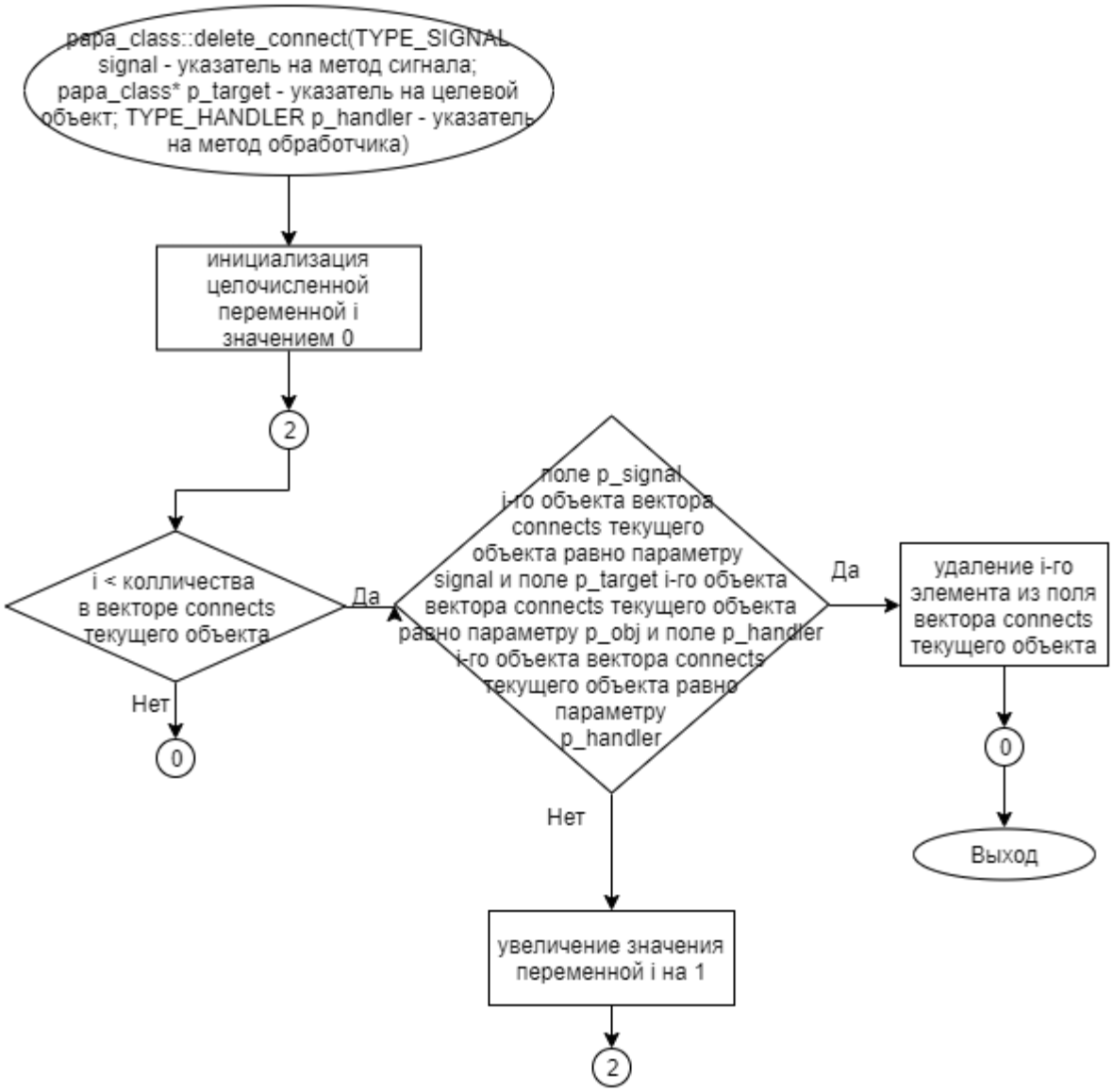

Рисунок 7 – Блок-схема алгоритма

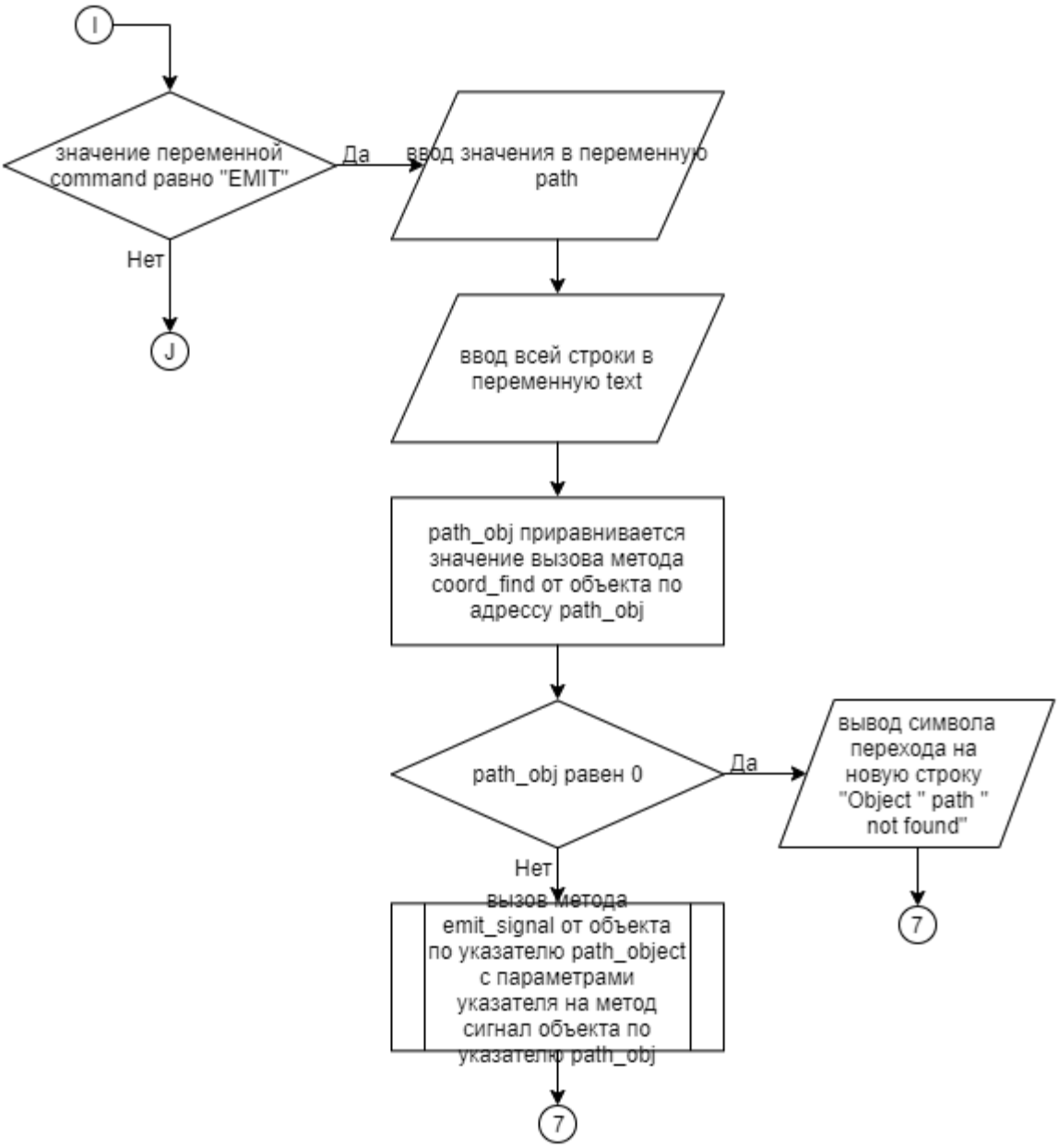

Рисунок 8 – Блок-схема алгоритма

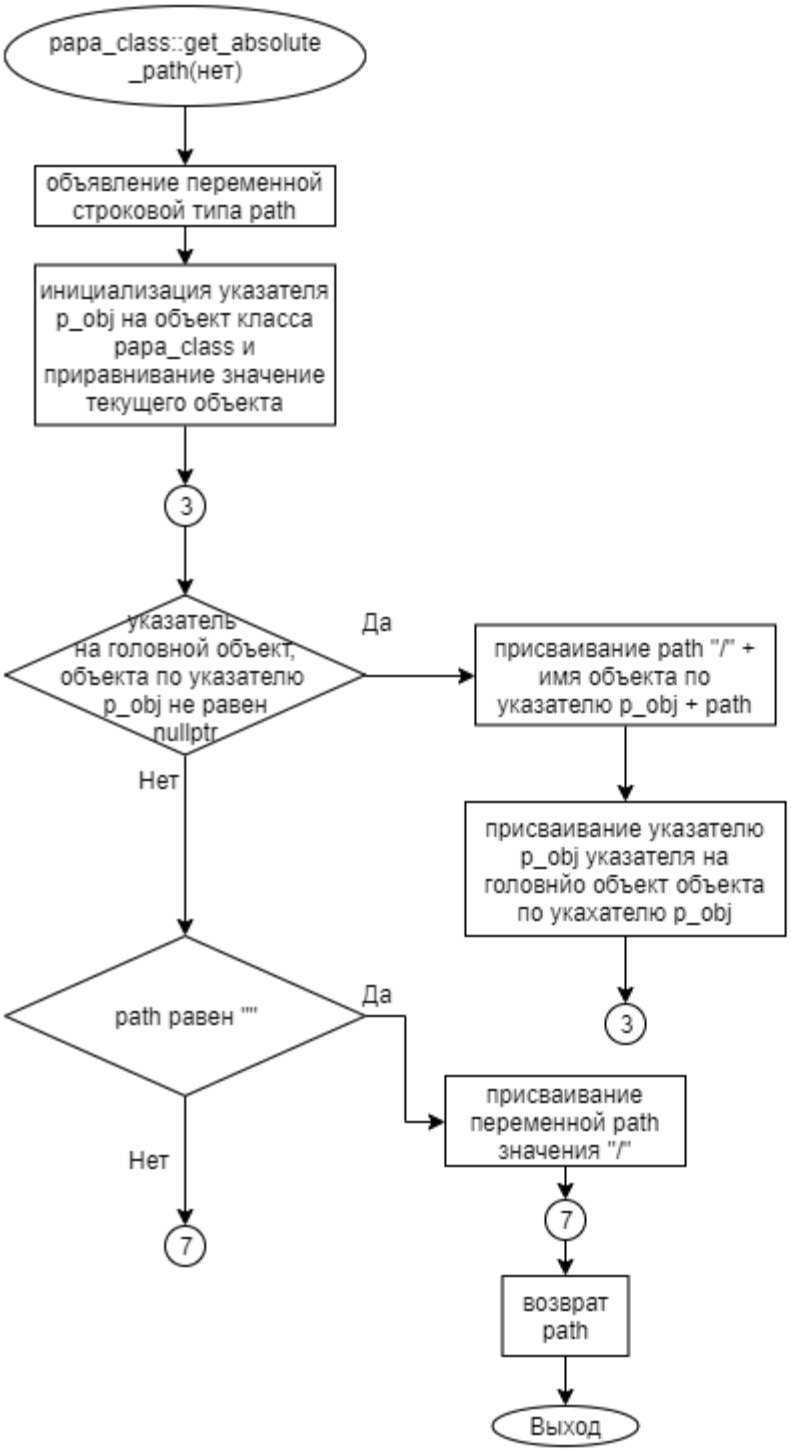

Рисунок 9 – Блок-схема алгоритма

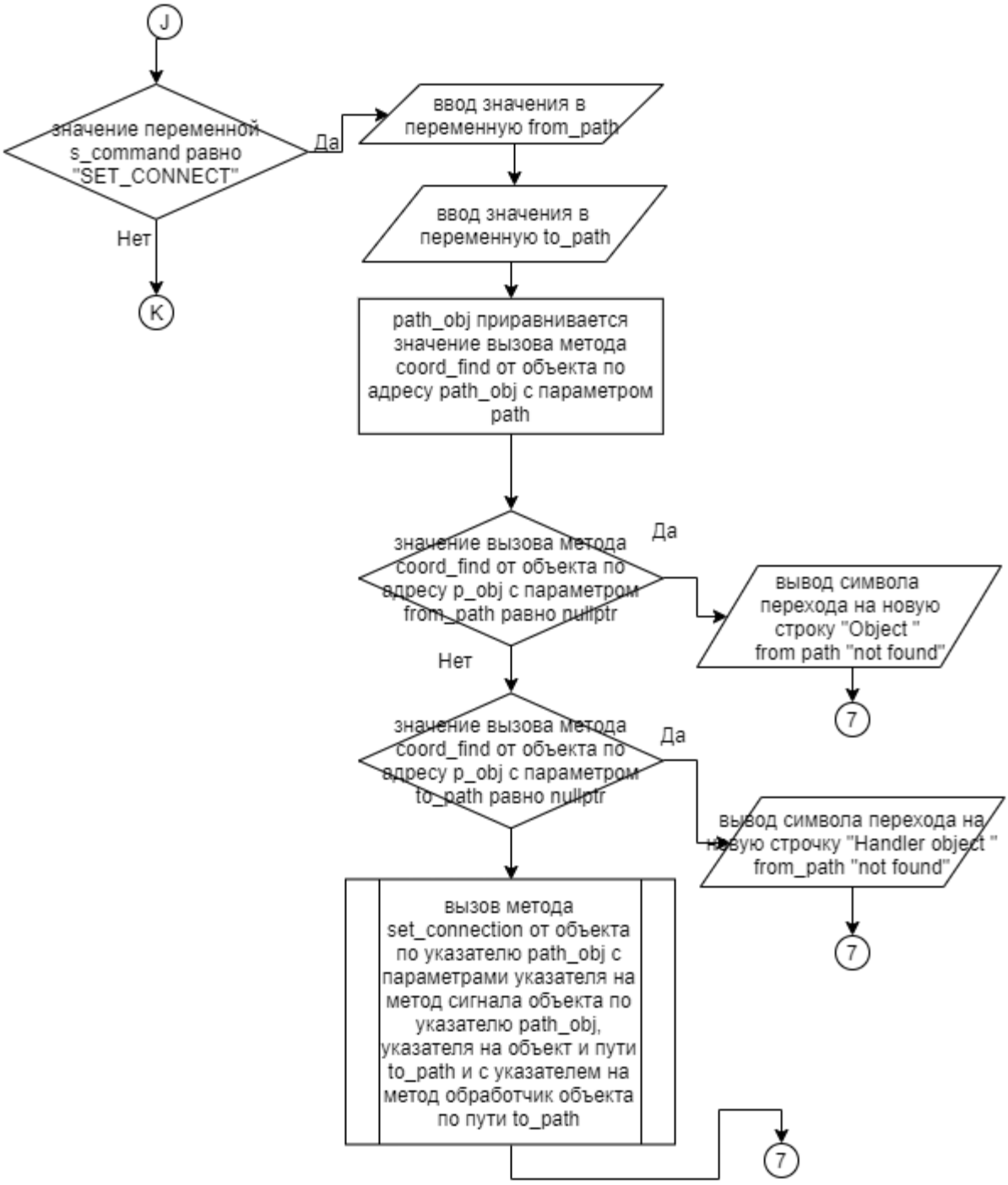

Рисунок 10 – Блок-схема алгоритма

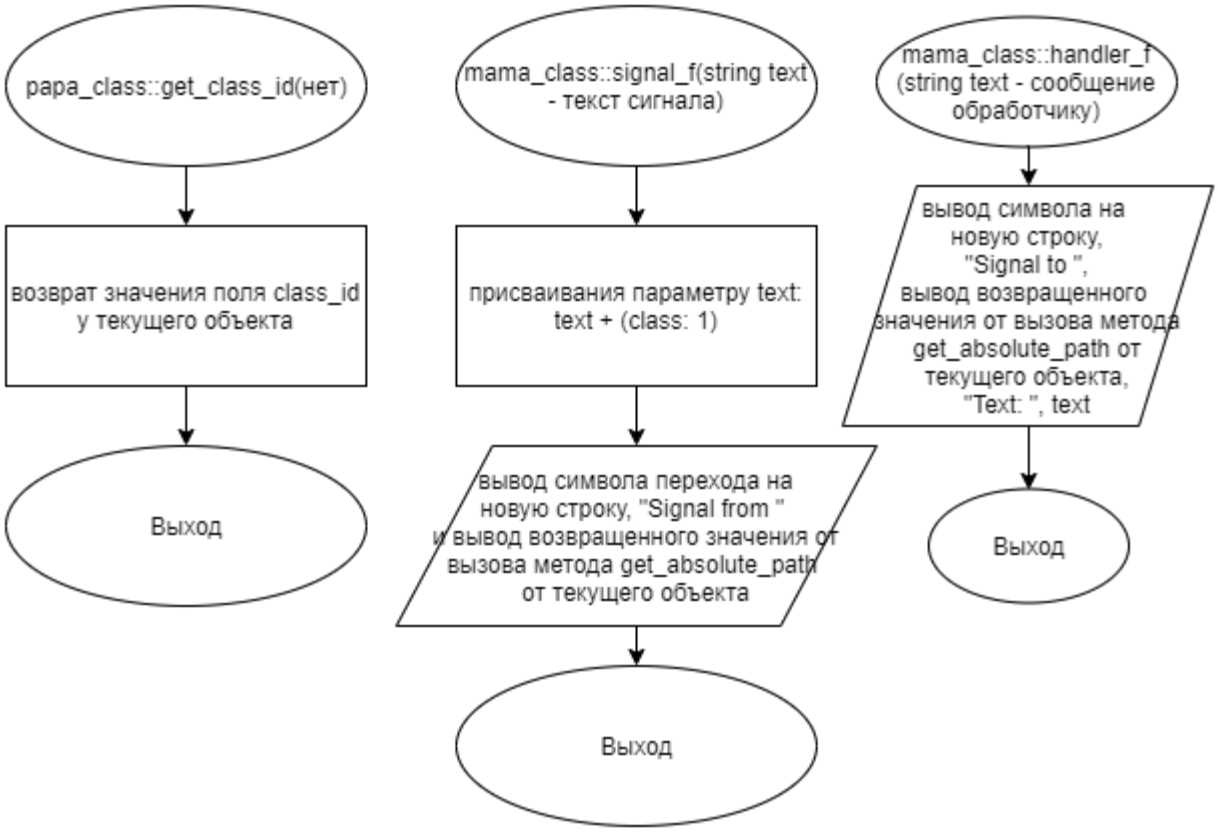

Рисунок 11 – Блок-схема алгоритма

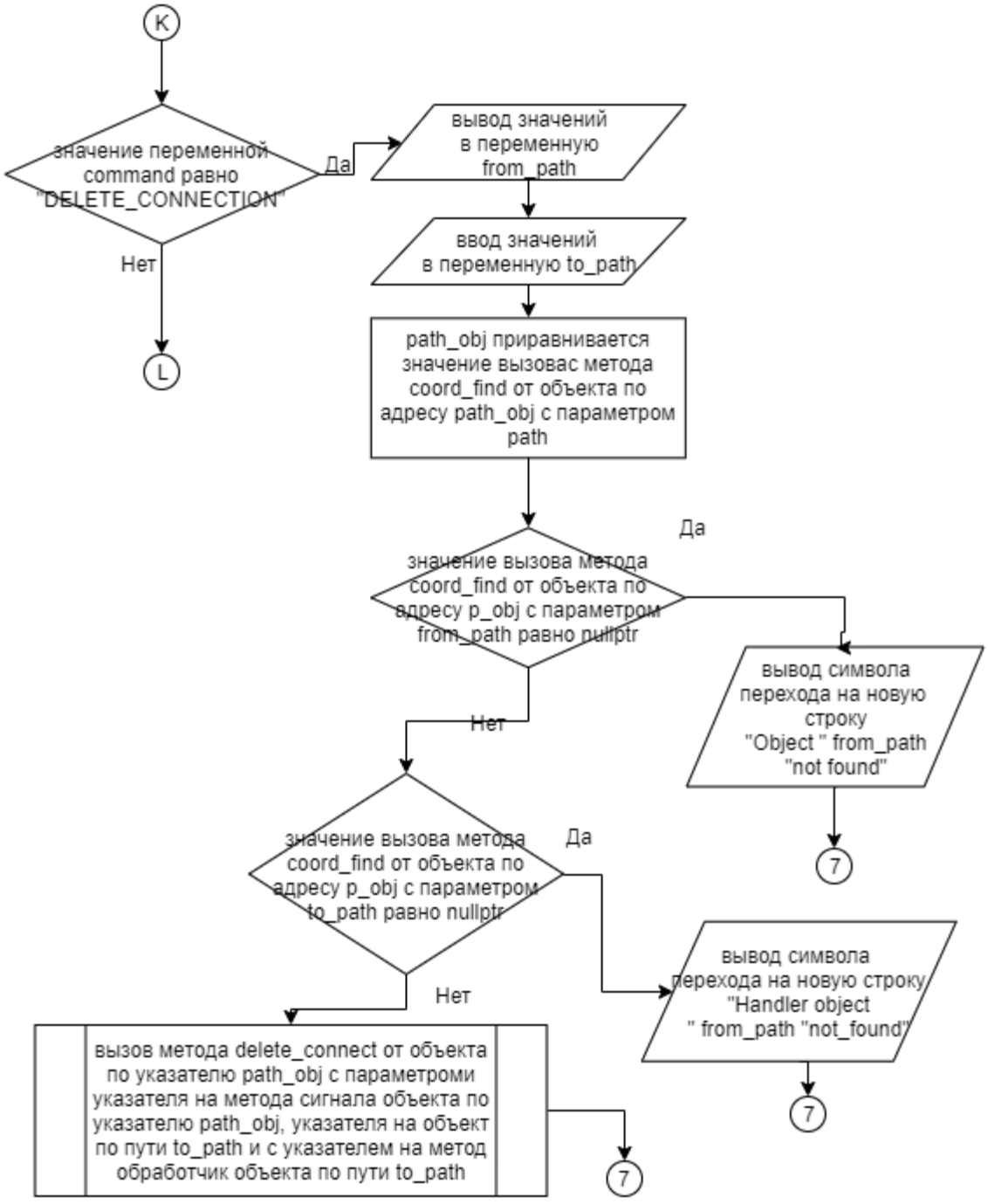

Рисунок 12 – Блок-схема алгоритма

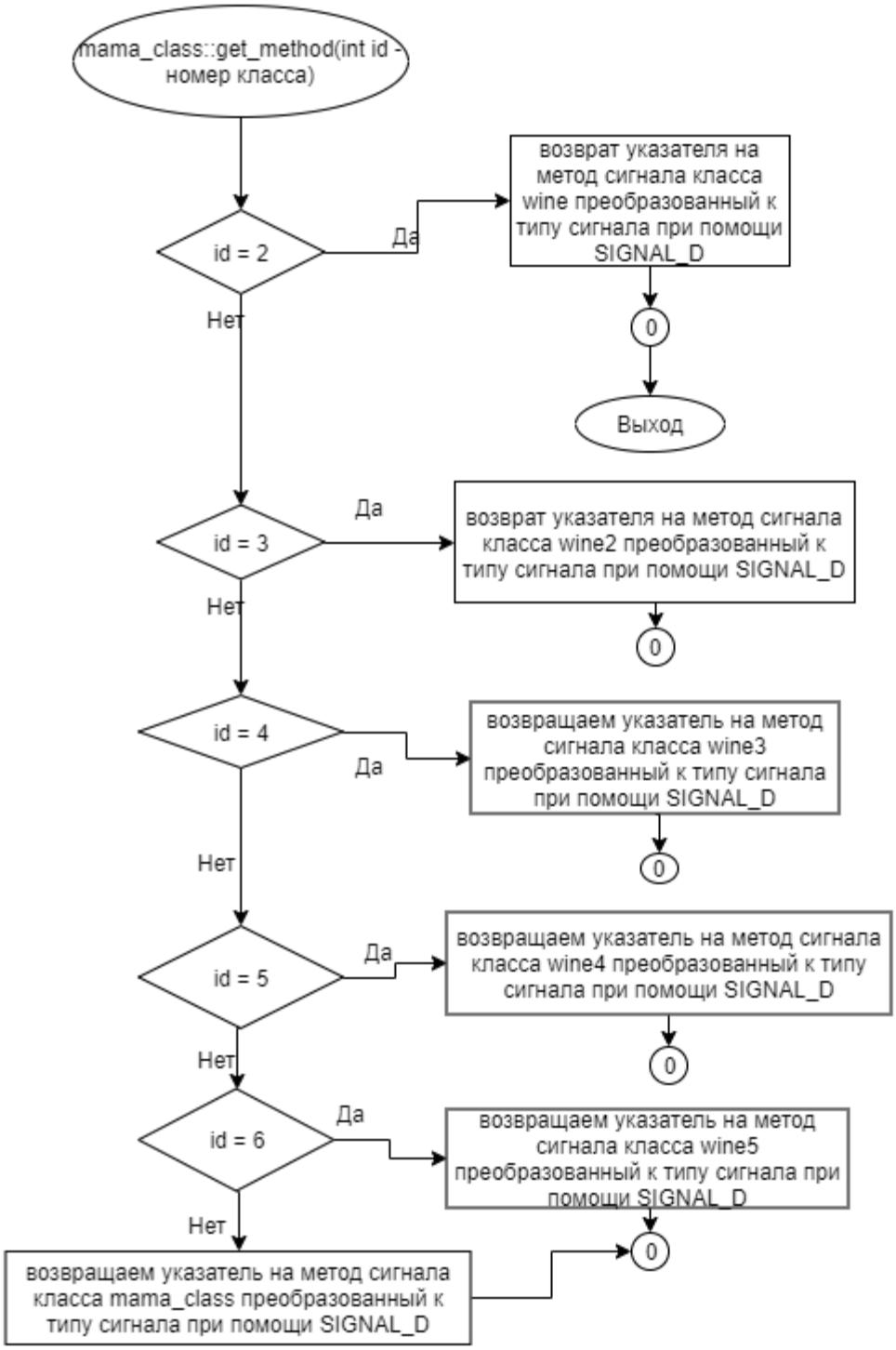

Рисунок 13 – Блок-схема алгоритма

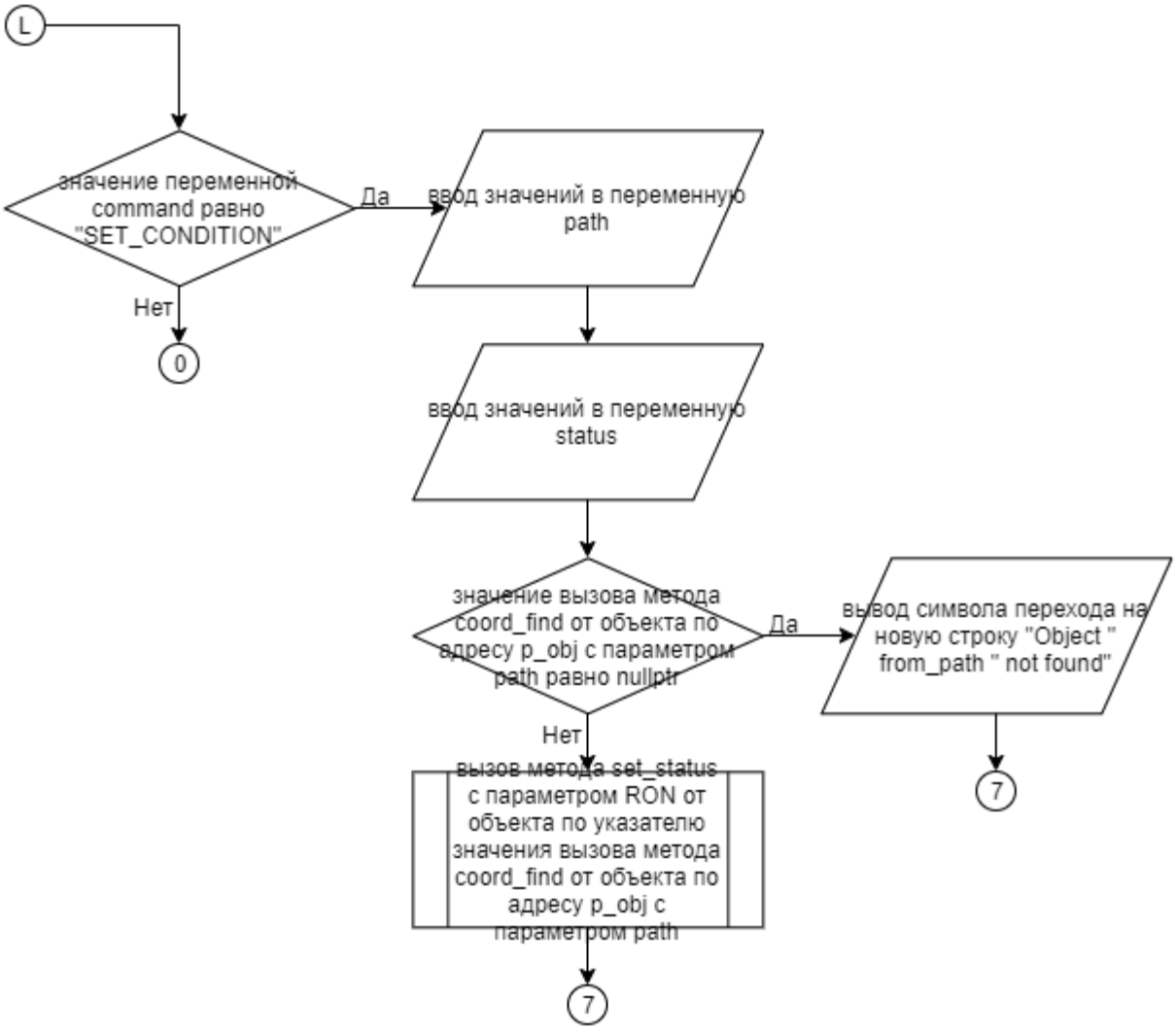

Рисунок 14 – Блок-схема алгоритма

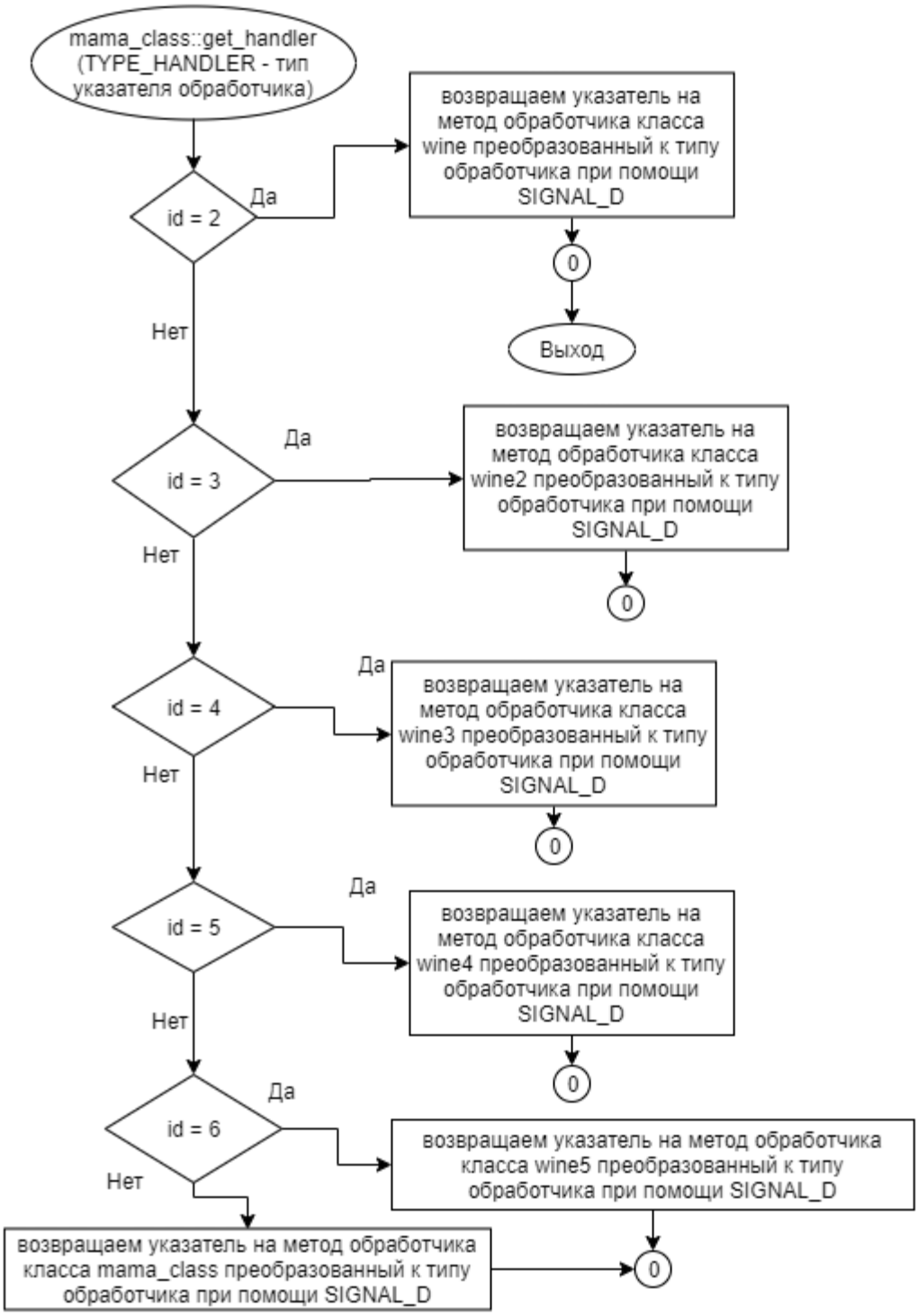

Рисунок 15 – Блок-схема алгоритма

Рисунок 16 – Блок-схема алгоритма

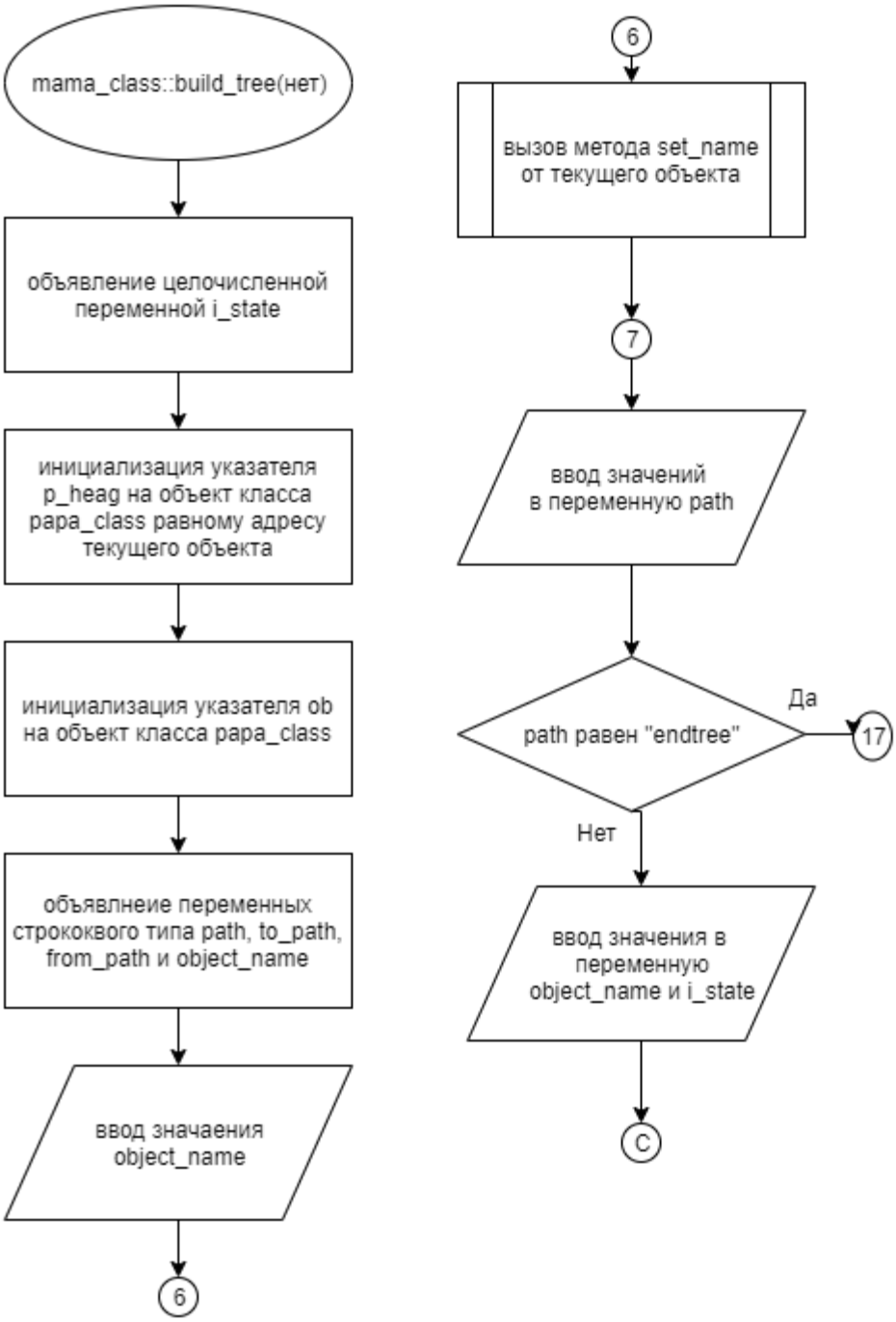

Рисунок 17 – Блок-схема алгоритма

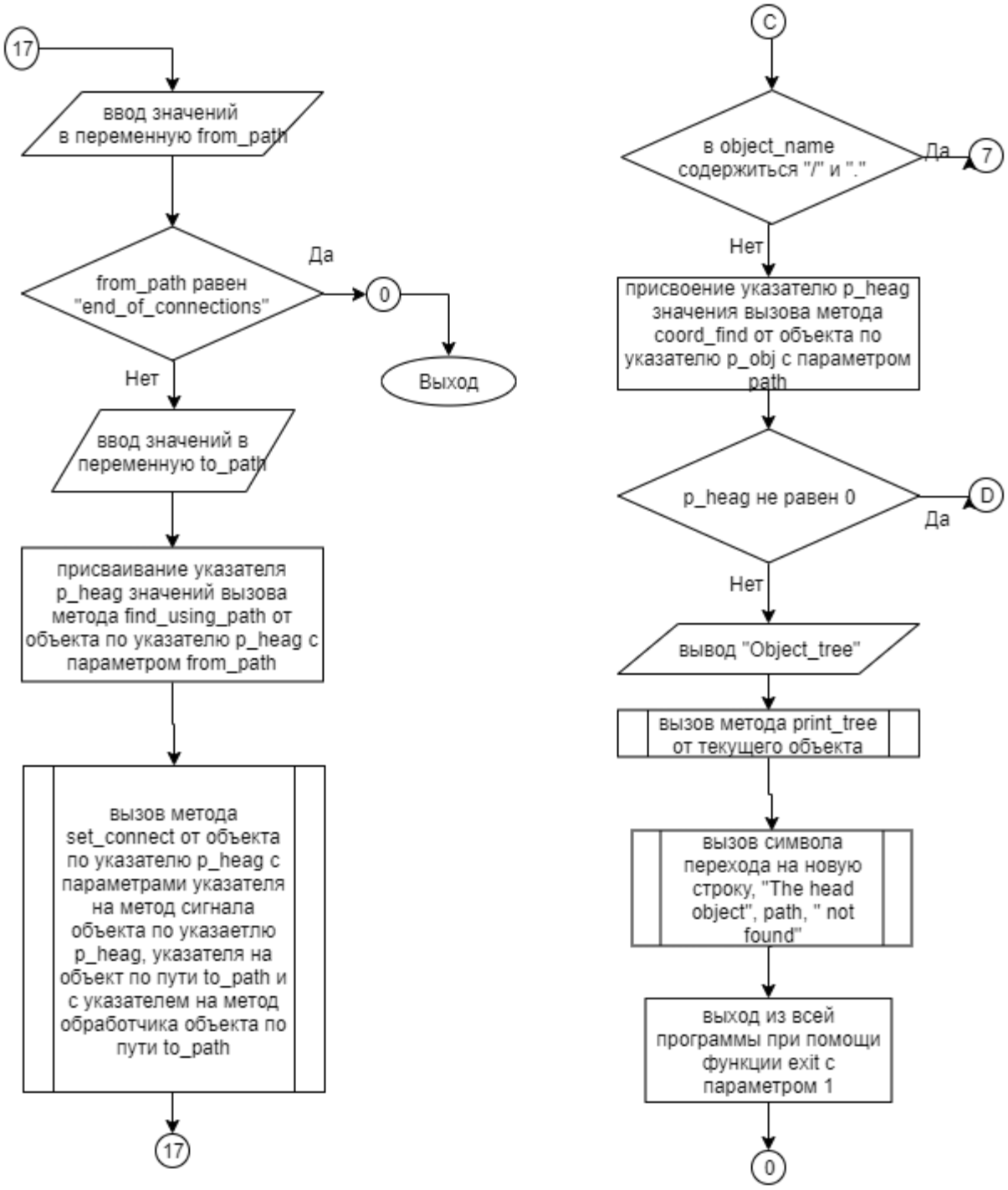

Рисунок 18 – Блок-схема алгоритма

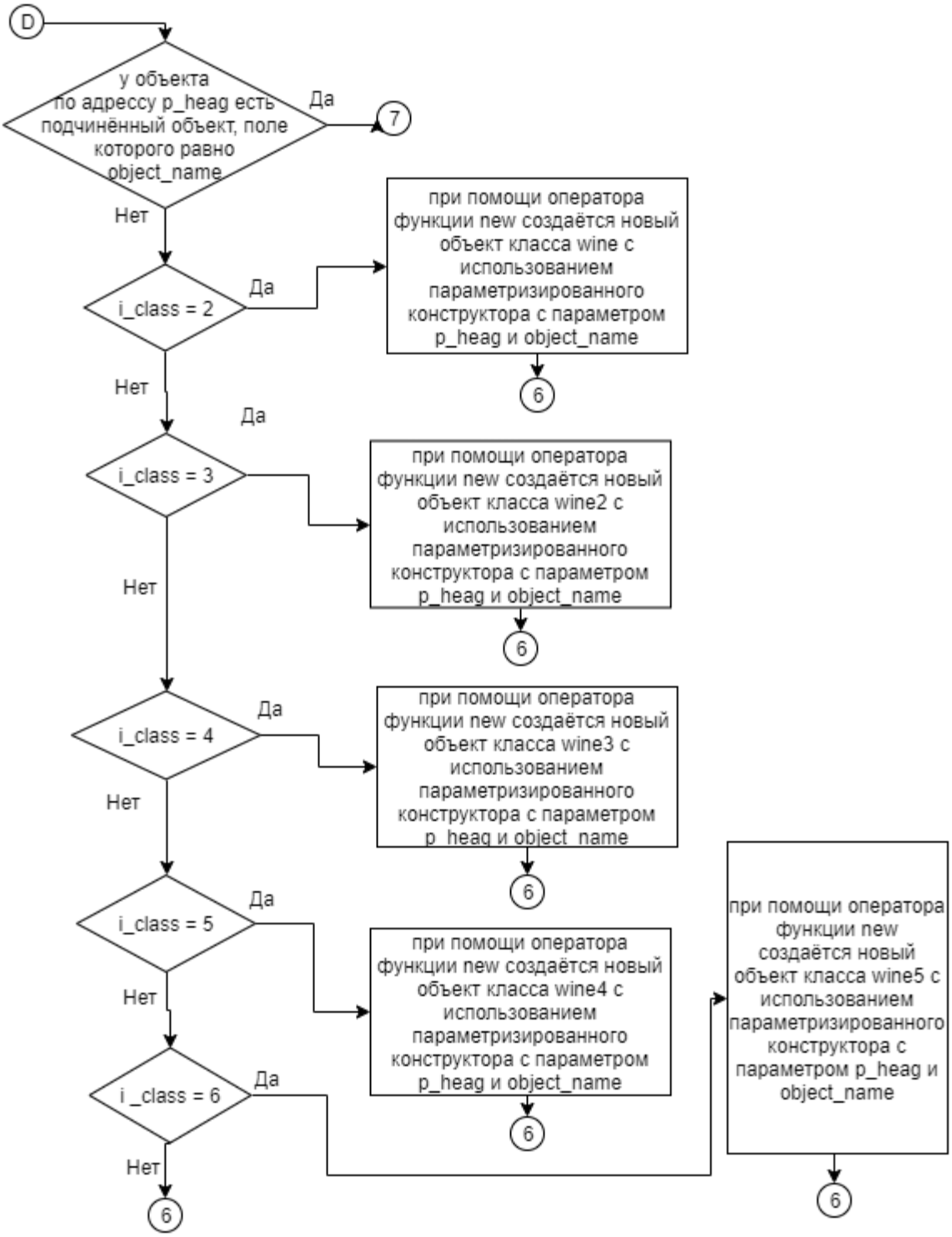

Рисунок 19 – Блок-схема алгоритма

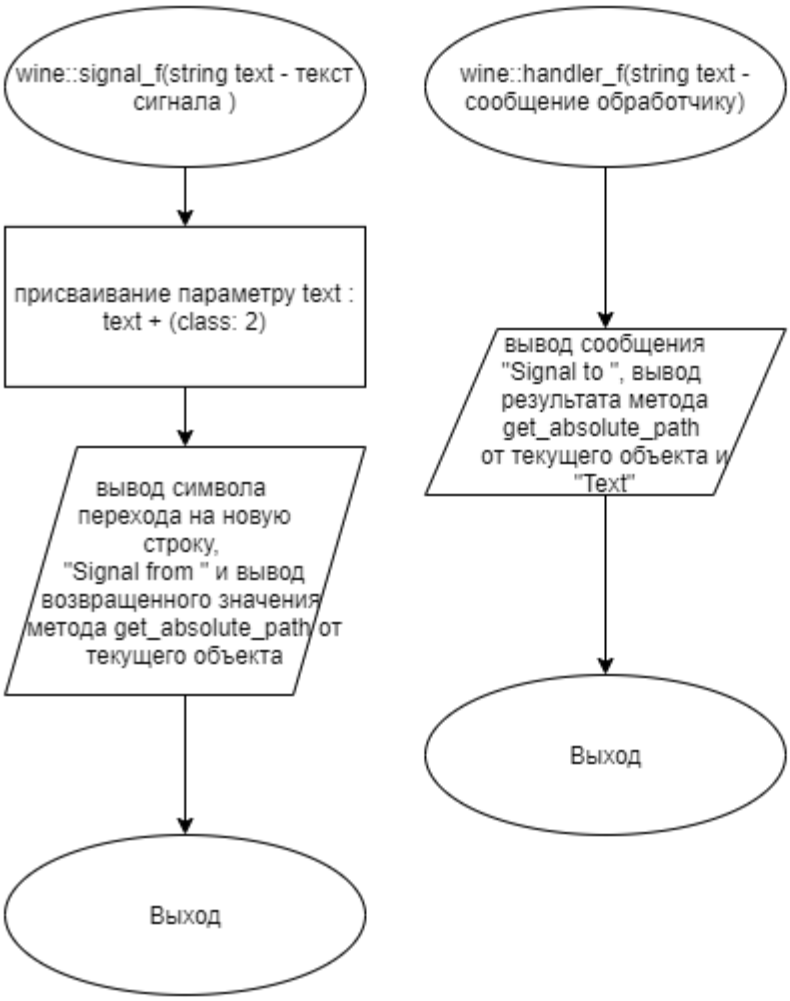

Рисунок 20 – Блок-схема алгоритма

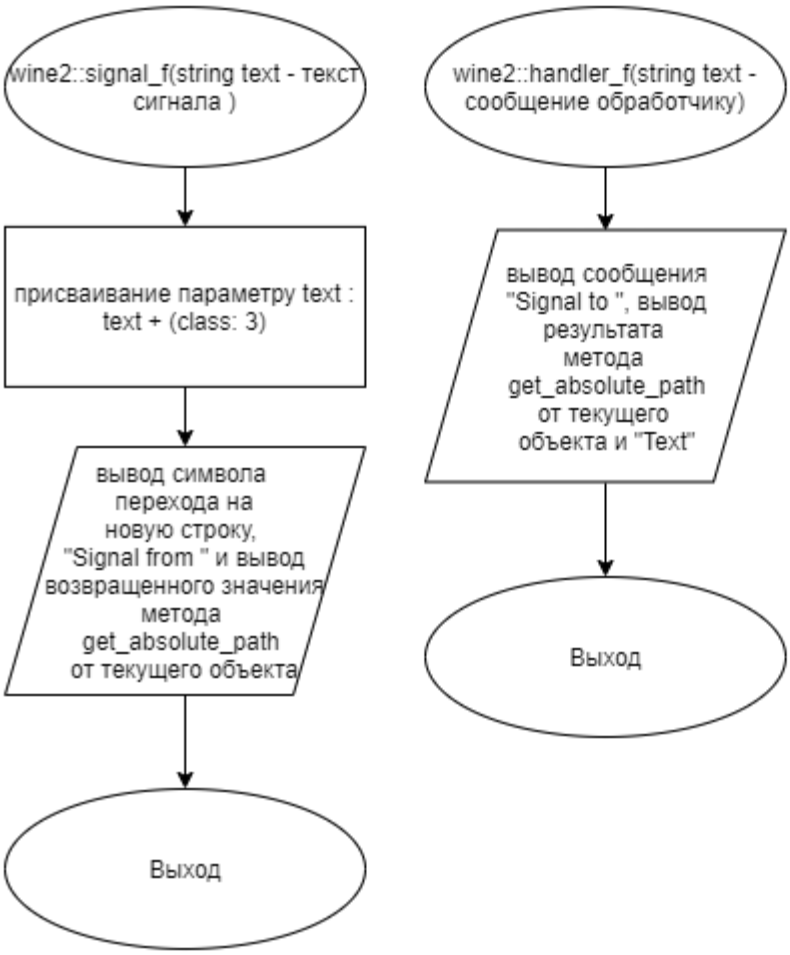

Рисунок 21 – Блок-схема алгоритма

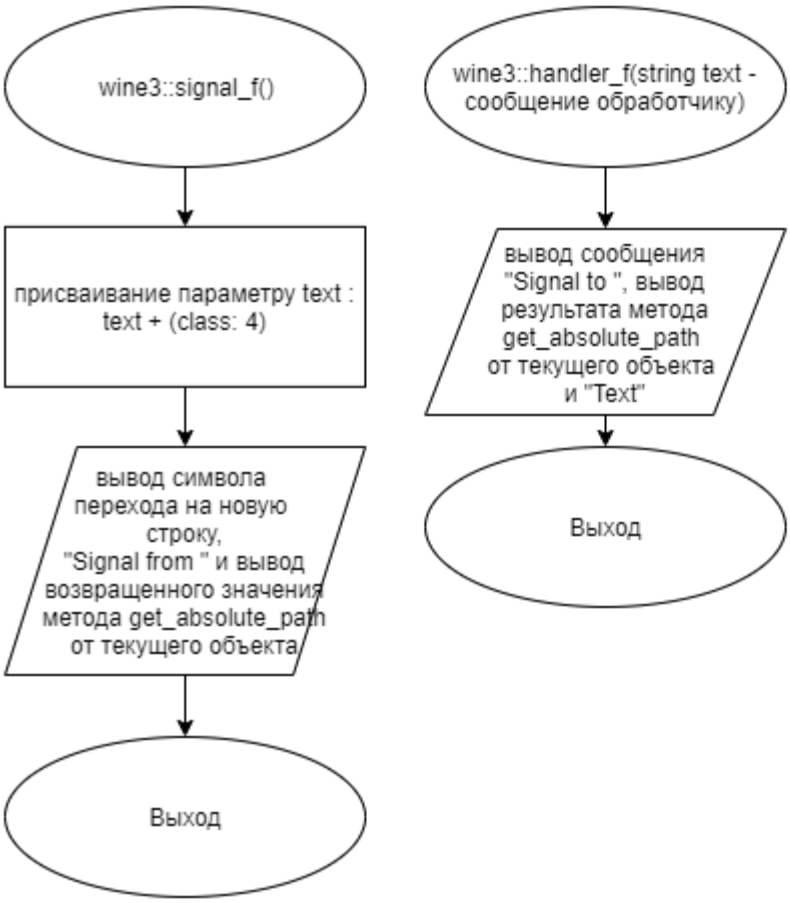

Рисунок 22 – Блок-схема алгоритма

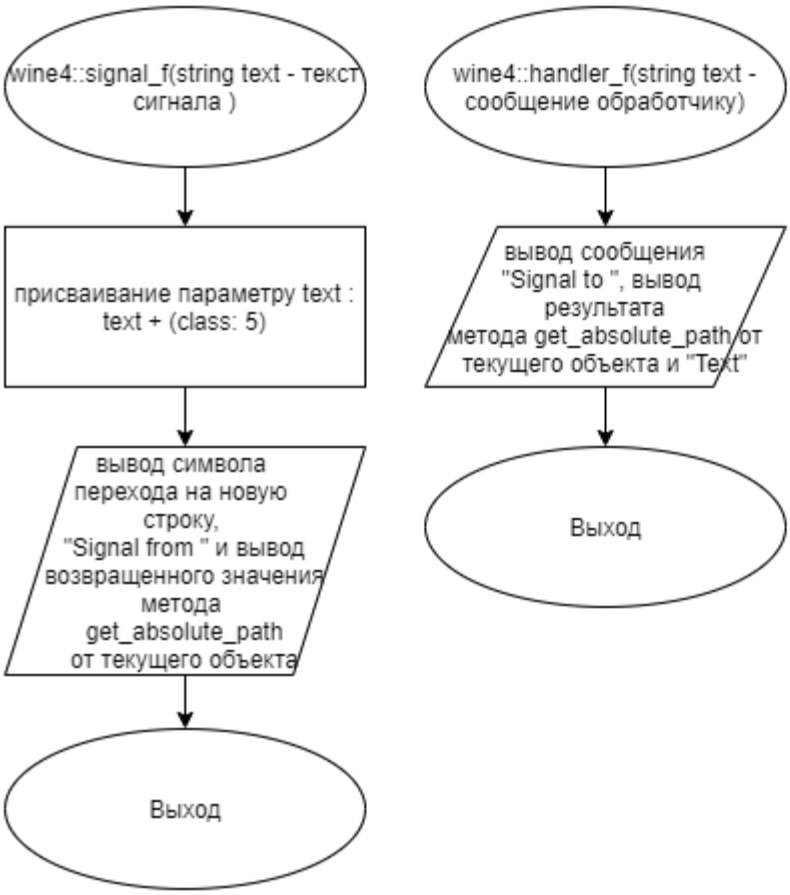

Рисунок 23 – Блок-схема алгоритма

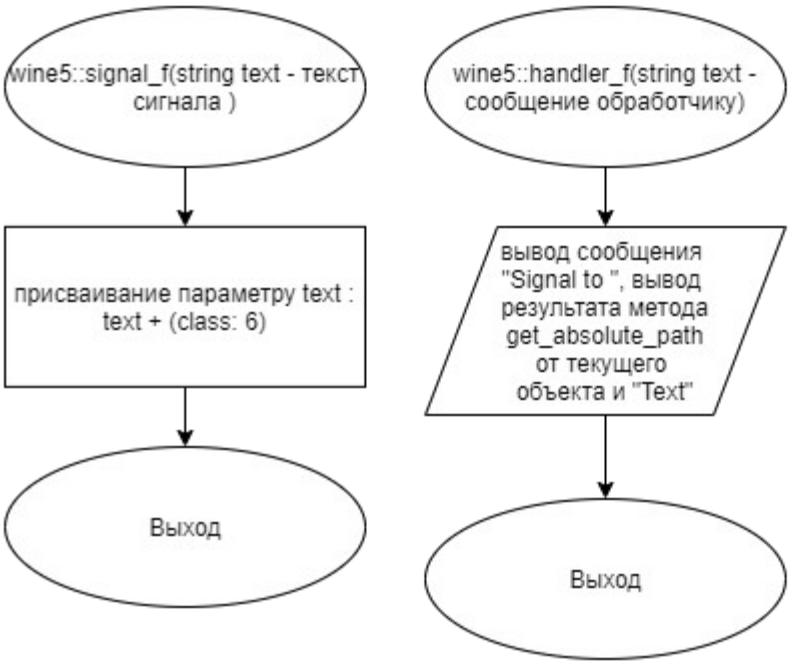

Рисунок 24 – Блок-схема алгоритма

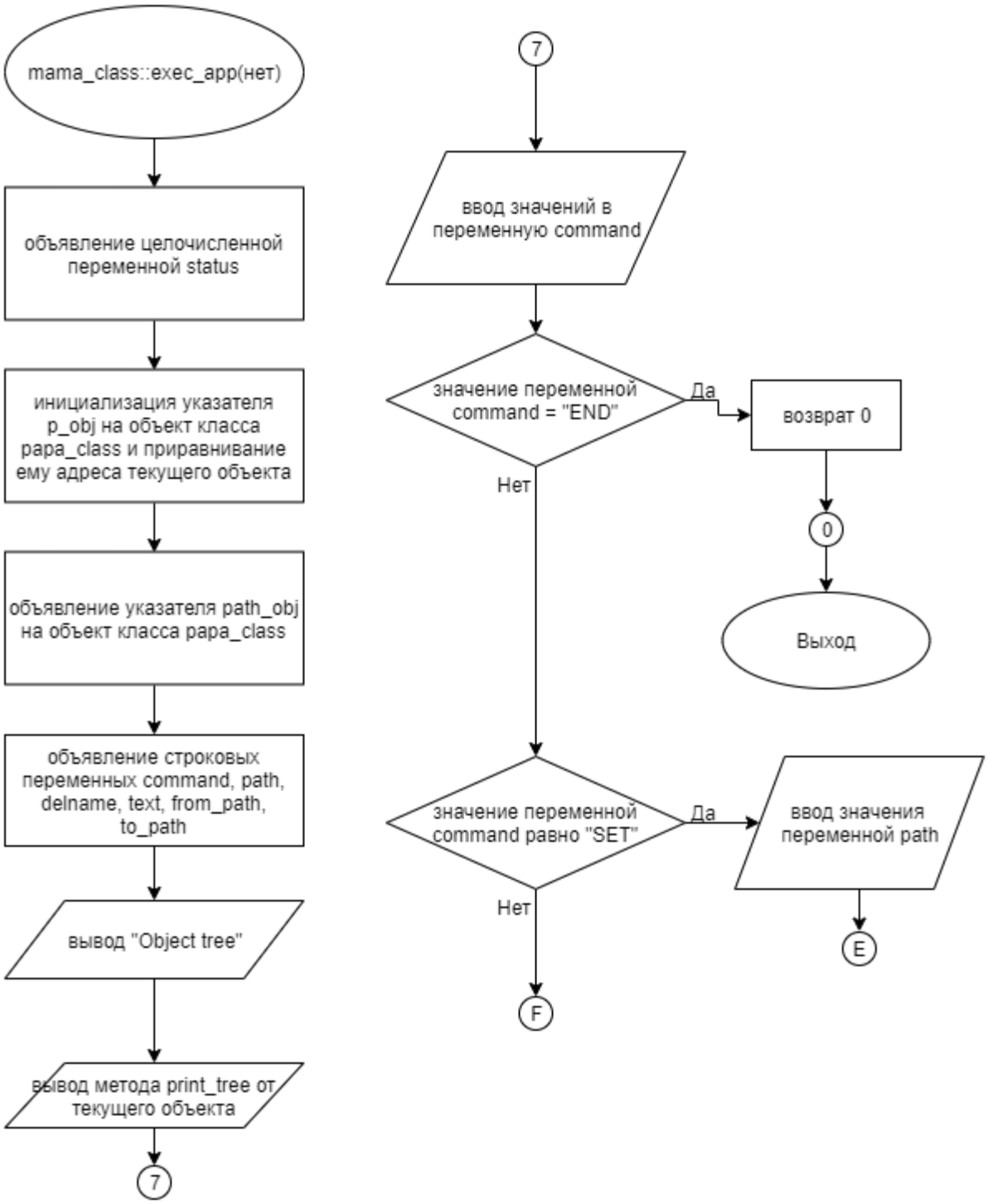

Рисунок 25 – Блок-схема алгоритма

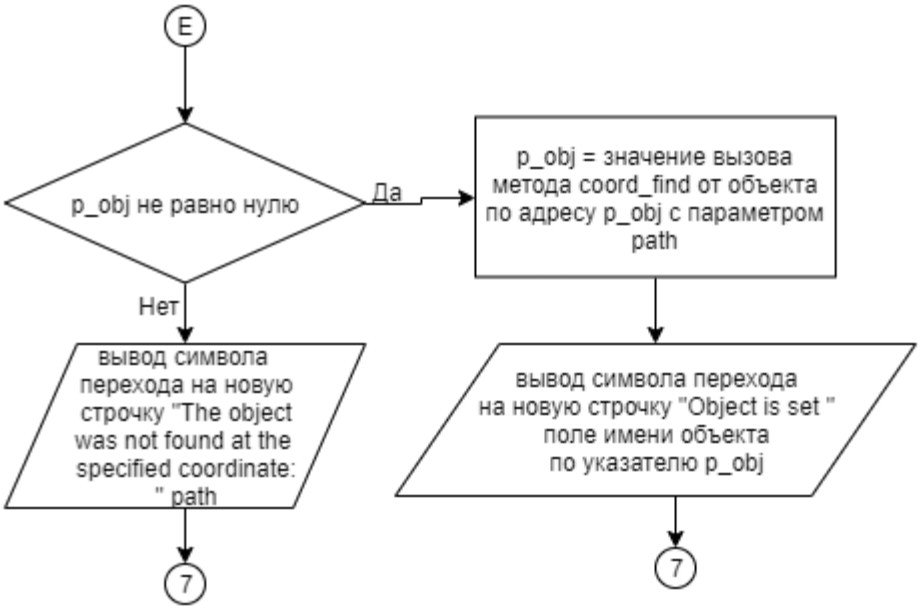

Рисунок 26 – Блок-схема алгоритма

5 КОД ПРОГРАММЫ

Программная реализация алгоритмов для решения задачи представлена ниже.

5.1 Файл main.cpp

Листинг 1 – main.cpp

```
#include <stdlib.h>
#include <stdio.h>
#include "mama_class.h"


int main (){
   mama_class  mommy( nullptr ); // создание корневого объекта
   mommy.build_tree(); // конструирование системы, построение дерева объектов
   return mommy.exec_app(); // запуск системы
}
```

5.2 Файл mama_class.cpp

Листинг 2 – mama_class.cpp

```
#include <iostream>
#include <vector>
#include <string>
#include <algorithm>
#include "mama_class.h"

mama_class::mama_class(papa_class* papa) : papa_class(papa){}

void mama_class::build_tree(){
   int i_class;
   papa_class* ob;
   papa_class* p_heag = this;
   std::string path, object_name, from_path, to_path;

   std::cin >> object_name;
   this->set_name(object_name);

   while(true){
      std::cin >> path;
```

```
        if (path == "endtree"){
           break;
        }
        std::cin >> object_name >> i_class;

        if      ((object_name.find('/')      !=      object_name.npos)      ||
    (object_name.find('.') != object_name.npos)){
           continue;
        }
        p_heag = p_heag->coord_find(path);
        if (p_heag != nullptr){
           if (p_heag->get_child_obj(object_name)){
              continue;
           }
           switch(i_class){
              case 2:
                 ob = new wine(p_heag, object_name);
                 ob->set_class_id(2);
                 break;
              case 3:
                 ob = new wine2(p_heag, object_name);
                 ob->set_class_id(3);
                 break;
              case 4:
                 ob = new wine3(p_heag, object_name);
                 ob->set_class_id(4);
                 break;
              case 5:
                 ob = new wine4(p_heag, object_name);
                 ob->set_class_id(5);
                 break;
              case 6:
                 ob = new wine5(p_heag, object_name);
                 ob->set_class_id(6);
                 break;
              default:
                 break;
           }
        }
        else{
           std::cout << "Object tree";
           this->build_tree_nst();
           std::cout << "\n" << "The head object " << path << " is not found";
           exit(1);
        }
     }
     while (true){
        std::cin >> from_path;

        if (from_path == "end_of_connections"){
           break;
        }
        std::cin >> to_path;
        p_heag = p_heag->coord_find(from_path);
```

```
      if (p_heag != nullptr){
         p_heag->set_connection(get_method(p_heag-
>get_class_id()),coord_find(to_path),get_handler(coord_find(to_path)-
>get_class_id()));
      }
   }
}

int mama_class::exec_app(){
   int status = 0;
   std::string s_command, s_coordinate, delname, text, from_path, to_path;
   papa_class* p_current = this;
   papa_class* p_obj = nullptr;

   std::cout << "Object tree";
   build_tree_nst();
   turn_on_tree();
   std::cout << "\n";
   while (true){
      std::cin >> s_command;
      if (s_command == "END"){
         break;
      }
      p_obj = p_current->coord_find(s_coordinate);
      if (s_command == "SET"){
         std::cin >> s_coordinate;
         if (p_obj != nullptr){
            p_current = p_obj;
            std::cout << "Object is set: " << p_current->get_name() << "\n";
         }
         else{
            std::cout  <<  "The  object  was  not  found  at  the  specified
coordinate: " << s_coordinate <<"\n";
         }
      }
      if (s_command == "FIND"){
         std::cin >> s_coordinate;
         if (p_obj != nullptr){
            std::cout  <<  s_coordinate  <<  "       Object  name:  "  <<  p_obj-
>get_name()<<"\n";
         }
         else{
            std::cout << s_coordinate << "     Object is not found" << "\n";
         }
      }
      if (s_command == "MOVE"){
         std::cin >> s_coordinate;
         if (p_current->new_papa(p_obj)){
            std::cout << "New head object: " << p_obj->get_name()<<"\n";
         }
         else if (p_obj == nullptr){
            std::cout << s_coordinate << "    Head object is not found" << "\
n";
         }
         else if (p_obj->get_child_obj(p_current->get_name())!=nullptr){
            std::cout  <<  s_coordinate  <<  "         Dubbing  the  names  of
```

```
subordinate objects" <<"\n";
        }
        else{
           std::cout << s_coordinate << "      Redefining the head object
failed"<<"\n";
        }
     }
     if (s_command == "DELETE"){
        std::cin >> s_coordinate;
        if (p_obj != nullptr){
           std::string s_abs_path = "";
           papa_class* p_object = p_obj;
           while (p_object->get_papa_obj() != nullptr){
              s_abs_path = "/" + p_object->get_name() + s_abs_path;
              p_object = p_object->get_papa_obj();
           }
           std::cout  <<  "The  object  "  <<  s_abs_path  <<  "  has  been
deleted"<<"\n";
           p_current->delete_subobj(p_obj->get_name());
        }
     }
     if (s_command == "EMIT"){
        std::cin >> s_coordinate;
        getline(std::cin, text);
        p_obj = coord_find(s_coordinate);
        if (p_obj != nullptr){
           p_obj->emit_signal(get_method(p_obj->get_class_id()), text);
        }
        else{
           std::cout << "Object " << s_coordinate << " not found \n";
        }
     }
     if (s_command == "SET_CONNECT"){
        std::cin >> from_path;
        std::cin >> to_path;
        p_obj = p_current->coord_find(from_path);
        if (p_obj == nullptr){
           std::cout << "Object " << from_path << " not found" << "\n";
        }
        else if (p_obj->coord_find(to_path) == nullptr){
           std::cout << "Handler object " << to_path << " not found" << "\
n";
        }
        else{
           p_obj->set_connection(get_method(p_obj->get_class_id()),
coord_find(to_path), get_handler(coord_find(to_path)->get_class_id()));
        }
     }
     if (s_command == "DELETE_CONNECT"){
        std::cin >> from_path;
        std::cin >> to_path;
        p_obj = coord_find(from_path);
        if (p_obj == nullptr){
           std::cout << "Object " << s_coordinate << " not found" << "\n";
        }
```

```
            else if(coord_find(to_path)==nullptr){
               std::cout << "Handler object " << from_path << " not found" << "\
n";
            }
            else{
               p_obj->delete_connection(get_method(p_obj->get_class_id()),
coord_find(to_path), get_handler(coord_find(to_path)->get_class_id()));
            }
         }
         if (s_command == "SET_CONDITION"){
            std::cin >> s_coordinate;
            std::cin >> status;
            if (coord_find(s_coordinate)==nullptr){
               std::cout << "Object " << s_coordinate << " not found" << "\n";
               continue;
            }
            else{
               coord_find(s_coordinate)->set_status(status);
            }
         }
      }
      return 0;
   }
   TYPE_HANDLER mama_class::get_handler(int id){
      switch(id){
         case 2:
            return HANDLER_D(wine::handler_f);
            break;
         case 3:
            return HANDLER_D(wine2::handler_f);
            break;
         case 4:
            return HANDLER_D(wine3::handler_f);
            break;
         case 5:
            return HANDLER_D(wine4::handler_f);
            break;
         case 6:
            return HANDLER_D(wine5::handler_f);
            break;
         default:
            return HANDLER_D(mama_class::handler_f);
            break;
      }
   }

   TYPE_SIGNAL mama_class::get_method(int id){
      switch(id){
         case 2:
            return SIGNAL_D(wine::signal_f);
            break;
         case 3:
            return SIGNAL_D(wine2::signal_f);
            break;
         case 4:
```

```
            return SIGNAL_D(wine3::signal_f);
            break;
         case 5:
            return SIGNAL_D(wine4::signal_f);
            break;
         case 6:
            return SIGNAL_D(wine5::signal_f);
            break;
         default:
            return SIGNAL_D(mama_class::signal_f);
            break;
      }
   }
   void mama_class::signal_f(std::string& text){
      text += " (class: 1)";
      std::cout << "Signal from " << get_absolute_path()<<"\n";
   }

   void mama_class::handler_f(std::string text){
      std::cout << "Signal to " << get_absolute_path() << " Text: " << text <<
   "\n";
   }
```

5.3 Файл mama_class.h

Листинг 3 – mama_class.h

```
   #ifndef __MAMA_CLASS__H
   #define __MAMA_CLASS__H

   #include "papa_class.h"
   #include "wine.h"
   #include "wine2.h"
   #include "wine3.h"
   #include "wine4.h"
   #include "wine5.h"

   void ADP(std::string text);
   void ADP(int text);
   void ADI(std::string text, std::string separator);
   void ADI(int text, std::string separator);
   void error_check_data();

   class mama_class : public papa_class{
   public:
      mama_class(papa_class* papa);
      void build_tree();
      int exec_app();
      //KB 4
      void signal_f(std::string &text);
```

```
   void handler_f(std::string text);
private:
   TYPE_SIGNAL get_method(int id);
   TYPE_HANDLER get_handler(int id);
};


#endif
```

5.4 Файл papa_class.cpp

Листинг 4 – papa_class.cpp

```
#include "papa_class.h"
#include <iostream>
#include <queue>
#include <string>
#include <algorithm>


papa_class::papa_class(papa_class*         papa,std::string         s_name){
//Параметризированный конструктор
   this -> s_name = s_name;
   this -> papa = papa;
   if (papa != nullptr){
      papa -> child_vector.push_back(this);
   }
}
bool papa_class::set_name(std::string s_new_name){
   if (this -> papa){
      for (int i = 0; i < papa -> child_vector.size(); i++){
         if (papa -> child_vector[i] -> get_name() == s_new_name){
            return false;
         }
      }
   }
   this -> s_name = s_new_name;
   return true;
}
papa_class::~papa_class(){ //Деструктор
   papa_class* root_ptr = this;
   while (root_ptr->get_papa_obj() != nullptr){
      root_ptr = root_ptr->get_papa_obj();
   }
   std::queue<papa_class*>q;
   q.push(root_ptr);
   int i = 0;
   while (!q.empty()){
      i = 0;
      if (q.front()==this){
```

```
            q.pop();
            continue;
         }
         for (i = 0; i < connects.size(); ++i){
            q.front()->get_name();
            if (q.front()->connects[i]->p_target == this){
               delete q.front()->connects[i];
               q.front()->connects.erase(q.front()->connects.begin()+i);
               i--;
            }
         }
      }
   }
   papa_class* papa_class::get_child_obj(std::string s_name){
      for (int i = 0; i < child_vector.size(); i++){
         if(child_vector[i] -> get_name() == s_name){
            return child_vector[i];
         }
      }
      return nullptr;
   }
   void papa_class::build_tree_nst(int lev){
      int i;
      std::cout << std::endl;
      for (i = 0; i < lev; i++){
         std::cout << "    ";
      }
      std::cout << this->get_name();
      for (i = 0; i < this->child_vector.size(); i++){
         this->child_vector[i]->build_tree_nst(lev+1);
      }
   }
   void papa_class::build_tree_st(int lev){
      int i;
      std::cout << std::endl;
      for (i = 0; i < lev; i++){
         std::cout << "    ";
      }
      std::cout << get_name();
      if (this->status != 0){
         std::cout << " is ready";
      }
      else{
         std::cout << " is not ready";
      }
      for (i = 0; i < this->child_vector.size(); i++){
         child_vector[i]->build_tree_st(lev+1);
      }
   }
   std::string papa_class::get_name(){
      return s_name;
   }
   papa_class* papa_class::get_papa_obj(){
      return papa;
   }
```

```
papa_class* papa_class::search_obj_fc(std::string s_name){
   papa_class* p_founded = nullptr;
   std::queue<papa_class*> q;
   q.push(this);
   while (q.empty() != true){
      papa_class* p_front = q.front();
      q.pop();
      if (p_front->get_name()==s_name){
         if (p_founded == nullptr){
            p_founded = p_front;
         }
         else{
            return nullptr;
         }
      }
      for (auto child:p_front->child_vector){
         q.push(child);
      }
   }
   return p_founded;
}
papa_class* papa_class::search_obj_al(std::string s_name){
   papa_class* root = this;
   while (root->get_papa_obj() != nullptr){
      root = root->get_papa_obj();
   }
   return root->search_obj_fc(s_name);
}
void papa_class::set_status(int status){
   if ((papa == nullptr) || (papa->status!=0)){
         this->status = status;
      }
      if (status == 0){
         this->status = status;
         for (int i = 0; i<this->child_vector.size();i++){
            child_vector[i]->set_status(status);
         }
      }
}
bool papa_class::new_papa(papa_class* p_new_papa){
   papa_class* p_temp = p_new_papa;
   if (this->get_papa_obj() == nullptr){
      return false;
   }
   if (p_new_papa == nullptr){
      return false;
   }
   if (p_new_papa->get_child_obj(this->get_name()) != nullptr){
      return false;
   }
   while (p_temp != nullptr){
      if (p_temp == this){
         return false;
      }
      else{
```

```
            p_temp = p_temp->get_papa_obj();
         }
      }
      papa->child_vector.erase(find(papa->child_vector.begin(),            papa-
   >child_vector.end(), this));
      this->papa = p_new_papa;
      papa->child_vector.push_back(this);

      return true;
   }
   void papa_class::delete_subobj(std::string s_name){
      papa_class* temp = this->get_child_obj(s_name);
      if (temp){
         for (int i=0; i < child_vector.size(); i++){
            if (child_vector[i] == temp){
               delete temp;
               child_vector.erase(child_vector.begin()+i);
               break;
            }
         }
      }
   }
   papa_class* papa_class::coord_find(std::string s_coord){
      papa_class* p_root = this;
      int i_slash;
      std::string s_name;
      if (s_coord == "/"){
         while (p_root->get_papa_obj() != nullptr){
            p_root = p_root->get_papa_obj();
         }
         return p_root;
      }
      else if (s_coord == "."){
         return this;
      }
      else if (s_coord[0] == '/' && s_coord[1] == '/'){
         return this->search_obj_al(s_coord.substr(2));
      }
      else if (s_coord[0] == '.'){
         return this->search_obj_fc(s_coord.substr(1));
      }
      else{
         if (s_coord[0] == '/'){
            s_coord = s_coord.substr(1);
            while (p_root->get_papa_obj() != nullptr){
               p_root = p_root->get_papa_obj();
            }
         }
         i_slash = s_coord.find("/");
         if (i_slash != -1){
            s_name = s_coord.substr(0, i_slash);
            p_root = p_root->get_child_obj(s_name);
            if (p_root != nullptr){
               return p_root->coord_find(s_coord.substr(i_slash+1));
            }
```

```
            else{
               return nullptr;
            }
         }
         else{
            return p_root->get_child_obj(s_coord);
         }
      }
      return nullptr;
   }
   void papa_class::set_connection(TYPE_SIGNAL p_signal, papa_class* p_target,
   TYPE_HANDLER p_handler){
      o_sh* p_value;
      for (int i = 0; i < connects.size(); ++i){
         if  (connects[i]->signal  ==  p_signal  &&  connects[i]->p_target  ==
   p_target && connects[i]->p_handler == p_handler){
            return ;
         }
      }
      p_value = new o_sh();
      p_value->signal = p_signal;
      p_value->p_target = p_target;
      p_value->p_handler = p_handler;

      connects.push_back(p_value);
   }
   void    papa_class::delete_connection(TYPE_SIGNAL    p_signal,    papa_class*
   p_target, TYPE_HANDLER p_handler){
      for (int i = 0; i < connects.size(); ++i){
         if  (connects[i]->signal  ==  p_signal  &&  connects[i]->p_target  ==
   p_target && connects[i]->p_handler == p_handler){
            delete connects[i];
            connects.erase(connects.begin()+i);
            break;
         }
      }
   }
   void papa_class::emit_signal(TYPE_SIGNAL p_signal, std::string& message){
      if (!this->status){
         return;
      }
      TYPE_HANDLER handler;
      papa_class* p_object;
      (this->*p_signal)(message);
      for (int i = 0; i < connects.size(); ++i){
         if (connects[i]->signal == p_signal){
            handler = connects[i]->p_handler;
            p_object = connects[i]->p_target;
            if (connects[i]->p_target->status)(p_object->*handler)(message);
         }
      }
   }
   std::string papa_class::get_absolute_path(){
      std::string path;
      papa_class* p_obj = this;
```

```
   while (p_obj->get_papa_obj() != nullptr){
      path = "/" + p_obj->get_name() + path;
      p_obj = p_obj->get_papa_obj();
   }
   if (path == ""){
      path = "/";
   }
   return path;
}
int papa_class::get_class_id(){
   return class_id;
}
void papa_class::set_class_id(int class_id){
   this->class_id = class_id;
}
void papa_class::turn_on_tree(){
   status = 1;
   for (int i = 0; i < child_vector.size(); i++){
      child_vector[i]-> turn_on_tree();
   }
}
```

5.5 Файл papa_class.h

Листинг 5 – papa_class.h

```
#ifndef __PAPA_CLASS__H
#define __PAPA_CLASS__H
#define SIGNAL_D(signal_f) (TYPE_SIGNAL)(&signal_f);
#define HANDLER_D(handler_f) (TYPE_HANDLER)(&handler_f);
#include <string>
#include <vector>
#include <queue>

class papa_class;
typedef void(papa_class::*TYPE_SIGNAL)(std::string &message);
typedef void(papa_class::*TYPE_HANDLER)(std::string msg);

struct o_sh{
   TYPE_SIGNAL signal;
   papa_class* p_target;
   TYPE_HANDLER p_handler;
};

class papa_class{
private:
   std::string s_name;
   papa_class* papa;
   std::vector<papa_class*> child_vector;
   int status = 0;
```

```
        int class_id = 0;
        std::vector<o_sh*>connects;

    public:
        papa_class(papa_class* papa,std::string s_name = "papa obj");
        bool set_name(std::string s_new_name);
        bool new_papa(papa_class* s_name);
        std::string get_name();

        papa_class* get_papa_obj();
        papa_class* search_obj_fc(std::string s_name);
        papa_class* search_obj_al(std::string s_name);
        papa_class* get_child_obj(std::string s_name);
        papa_class* coord_find(std::string s_coord);

        void set_status(int status=0);
        void delete_subobj(std::string s_name);
        void build_tree_st(int lev=0);
        void build_tree_nst(int lev=0);
        ~papa_class();

        //KB4
        void    set_connection(TYPE_SIGNAL    p_signal,    papa_class*    p_target,
    TYPE_HANDLER p_handler);
        void   delete_connection(TYPE_SIGNAL   p_signal,   papa_class*   p_target,
    TYPE_HANDLER p_handler);
        void emit_signal(TYPE_SIGNAL p_signal, std::string &message);
        std::string get_absolute_path();
        int get_class_id();
        void set_class_id(int class_id);
        void turn_on_tree();
    };

    #endif
```

5.6 Файл wine2.cpp

Листинг 6 – wine2.cpp

```
    #include "wine2.h"
    #include <iostream>

    wine2::wine2(papa_class*   papa,   std::string   s_name)   :   papa_class(papa,
    s_name){}

    void wine2::signal_f(std::string& text){
        text += " (class: 3)";
        std::cout << "Signal from " << get_absolute_path() << "\n";
    }
    void wine2::handler_f(std::string text){
        std::cout << "Signal to " << get_absolute_path() << " Text: " << text <<
```

```
"\n";
}
```

5.7 Файл wine2.h

Листинг 7 – wine2.h

```
#ifndef __WINE2__H
#define __WINE2__H
#include "papa_class.h"
#include <string>

class wine2 : public papa_class{
public:
   wine2(papa_class* papa, std::string s_name);
   void signal_f(std::string &msg);
   void handler_f(std::string msg);
};

#endif
```

5.8 Файл wine3.cpp

Листинг 8 – wine3.cpp

```
#include "wine3.h"
#include <iostream>

wine3::wine3(papa_class*  papa,  std::string  s_name)  :  papa_class(papa,
s_name){}

void wine3::signal_f(std::string& text){
   text +=" (class: 4)";
   std::cout << "Signal from " << get_absolute_path() << "\n";
}
void wine3::handler_f(std::string text){
   std::cout << "Signal to " << get_absolute_path() << " Text: " << text <<
"\n";
}
```

5.9 Файл wine3.h

Листинг 9 – wine3.h

```
#ifndef __WINE3__H
#define __WINE3__H
#include "papa_class.h"
#include <string>

class wine3 : public papa_class{
public:
   wine3(papa_class* papa, std::string s_name);
   void signal_f(std::string &msg);
   void handler_f(std::string msg);
};

#endif
```

5.10 Файл wine4.cpp

Листинг 10 – wine4.cpp

```
#include "wine4.h"
#include <iostream>

wine4::wine4(papa_class*  papa,  std::string  s_name)  :  papa_class(papa,
s_name){}

void wine4::signal_f(std::string& text){
   text += " (class: 5)";
   std::cout << "Signal from " << get_absolute_path() << "\n";
}
void wine4::handler_f(std::string text){
   std::cout << "Signal to " << get_absolute_path() << " Text: " << text <<
"\n";
}
```

5.11 Файл wine4.h

Листинг 11 – wine4.h

```
#ifndef __WINE4__H
#define __WINE4__H
```

```
#include "papa_class.h"
#include <string>

class wine4 : public papa_class{
public:
   wine4(papa_class* papa, std::string s_name);
   void signal_f(std::string &msg);
   void handler_f(std::string msg);
};


#endif
```

5.12 Файл wine5.cpp

Листинг 12 – wine5.cpp

```
#include "wine5.h"
#include <iostream>

wine5::wine5(papa_class*  papa,  std::string  s_name)  :  papa_class(papa,
s_name){}

void wine5::signal_f(std::string& text){
   text += " (class: 6)";
   std::cout << "Signal from " << get_absolute_path() << "\n";
}
void wine5::handler_f(std::string text){
   std::cout << "Signal to " << get_absolute_path() << " Text: " << text <<
"\n";
}
```

5.13 Файл wine5.h

Листинг 13 – wine5.h

```
#ifndef __WINE5__H
#define __WINE5__H
#include "papa_class.h"
#include <string>

class wine5 : public papa_class{
public:
   wine5(papa_class* papa, std::string s_name);
   void signal_f(std::string &msg);
```

```
   void handler_f(std::string msg);
};


#endif
```

5.14 Файл wine.cpp

Листинг 14 – wine.cpp

```
#include "wine.h"
#include <iostream>
#include <string>

wine::wine(papa_class* papa, std::string s_name) : papa_class(papa, s_name)
{}

void wine::signal_f(std::string& text){
   text += " (class: 2)";
   std::cout << "Signal from " << get_absolute_path() << "\n";
}
void wine::handler_f(std::string text){
   std::cout << "Signal to " << get_absolute_path() << " Text: " << text <<
"\n";
}
```

5.15 Файл wine.h

Листинг 15 – wine.h

```
#ifndef __WINE__H
#define __WINE__H
#include "papa_class.h"
#include <string>

class wine : public papa_class{
public:
   wine(papa_class* papa, std::string s_name);
   void signal_f(std::string &msg);
   void handler_f(std::string msg);
};

#endif
```

6 ТЕСТИРОВАНИЕ

Результат тестирования программы представлен в таблице 23.

Таблица 23 – Результат тестирования программы

Входные данные	Ожидаемые выходные данные	Фактические выходные данные
appls_root / object_s1 3 / object_s2 2 /object_s2 object_s4 4 / object_s13 5 /object_s2 object_s6 6 /object_s1 object_s7 2 endtree /object_s2/object_s4 /object_s2/object_s6 /object_s2 /object_s1/object_s7 / /object_s2/object_s4 /object_s2/object_s4 / end_of_connections EMIT /object_s2/object_s4 Send message 1 EMIT /object_s2/object_s4 Send message 2 EMIT /object_s2/object_s4 Send message 3 EMIT /object_s1 Send message 4 END	Object tree appls_root object_s1 object_s7 object_s2 object_s4 object_s6 object_s13 Signal from /object_s2/object_s4 Signal to /object_s2/object_s6 Text: Send message 1 (class: 4) Signal to / Text: Send message 1 (class: 4) Signal from /object_s2/object_s4 Signal to /object_s2/object_s6 Text: Send message 2 (class: 4) Signal to / Text: Send message 2 (class: 4) Signal from /object_s2/object_s4 Signal to /object_s2/object_s6 Text: Send message 3 (class: 4) Signal to / Text: Send message 3 (class: 4) Signal from /object_s1	Object tree appls_root object_s1 object_s7 object_s2 object_s4 object_s6 object_s13 Signal from /object_s2/object_s4 Signal to /object_s2/object_s6 Text: Send message 1 (class: 4) Signal to / Text: Send message 1 (class: 4) Signal from /object_s2/object_s4 Signal to /object_s2/object_s6 Text: Send message 2 (class: 4) Signal to / Text: Send message 2 (class: 4) Signal from /object_s2/object_s4 Signal to /object_s2/object_s6 Text: Send message 3 (class: 4) Signal to / Text: Send message 3 (class: 4) Signal from /object_s1

ЗАКЛЮЧЕНИЕ

В процессе выполнения курсовой работы, были освоены принципы ООП,были получены теоретические знания и пркатичесские навыки по разработке классов, а также их методов. В ходе выполнения курсовой была освоена механика обработки и передачи специализированных сигналов. Так же стоит заметить, что курсовая работа была выполнена в цифровой среде Aurora, которая была разработана на базе РТУ МИРЭА. Она имеет как множество достоинств, так и несколько недочётов. Хочу отметить не точную работу отладчика, однако хочу заметить, что частые обновления программы, а так же отзывчивость разработчиков и удобство в получении задач невилирует все её недостатки.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ГОСТ 19 Единая система программной документации.

2. Методическое пособие студента для выполнения практических заданий, контрольных и курсовых работ по дисциплине «Объектно-ориентированное программирование» [Электронный ресурс] – URL: https://mirea.aco-avrora.ru/student/files/methodichescoe_posobie_dlya_laboratornyh_rabot_3.pdf (дата обращения 05.05.2021).

3. 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).

4. Шилдт Г. C++: базовый курс. 3-е изд. Пер. с англ.. — М.: Вильямс, 2019. — 624 с.

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).