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 Моделирование работы кофе-машины
(наименование темы)

Студент группы ИКБО-13-21 Черномуров Семён Андреевич
(учебная группа) *(Фамилия Имя Отчество)* *(подпись студента)*

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

Консультант ст.пр.каф.ВТ Асадова Ю.С.
(Должность, звание, ученая степень) *(подпись консультанта)*

Работа представлена к защите « 16 » мая 2022 г.

Допущен к защите « 16 » мая 2022 г.

Москва 2022 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

«_14_» марта 2022г.

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

Студент Черномуров Семён Андреевич Группа ИКБО-13-21

Тема Моделирование работы кофе-машины

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 «_16_» мая 2022 г.

Задание на курсовую работу выдал _____ (Асадова Ю.С.)
Подпись *ФИО консультанта*
«_28_» февраля 2022 г.

Задание на курсовую работу получил _____ (Черномуров С.А.)
Подпись *ФИО исполнителя*
«_28_» февраля 2022 г.

Москва 2022г.

ОТЗЫВ

на курсовую работу

по дисциплине «Объектно-ориентированное программирование»

Студент **Черномуров Семён Андреевич** группа ИКБО-13-21

(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: нет

Рекомендуемая оценка: отлично

_____ доцент каф.ВТ Ингтем Ж.Г.

(Подпись руководителя) *(ФИО руководителя)*

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

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

В настоящей работе разрабатывается и реализуется решение задачи по моделированию работы кофемашины. **Актуальность** данного решения заключена в том, что оно может быть применено в одной из самых распространенных сфер жизни - торговой сфере, а именно для автоматизированного изготовления тех или иных напитков (в данном случае – кофе) на продажу.

Целями курсовой работы являются: развитие навыков проектирования и реализации задач в области объектно-ориентированного программирования (далее – ООП), знакомство с основными принципами ООП, такими как наследование, инкапсуляция и полиморфизм, ознакомление с понятием параметризированного макроопределения препроцессора, а также получение навыков по работе с сигналами и обработчиками.

Задачей настоящей работы является разработка и построение системы в стиле ООП, и написание на ее основе программы, реализующей алгоритм моделирования работы кофемашины с соответствующей реализацией сигналов и обработчиков и связей между ними.

1 ПОСТАНОВКА ЗАДАЧИ

Надо моделировать работу кофемашины следующей конструкции. Кофемашина состоит из следующих элементов:

- пульт управления;
- устройство приема денег;
- устройство выдачи кофе;
- устройство возврата сдачи;
- экран отображения состояния и информации.

Пульт управления содержит кнопки:

- выбора кофе (множество кнопок);
- возврата денег.

Правила работы с кофемашиной.

Кофемашина готовится к работе следующим образом:

1. Задается количество сортов кофе (количество кнопок для выбора кофе) их названия и их стоимость, кратная 5 рублям. Загружается кофе. Подразумевается, что объем достаточен для работы.
2. Загружается заданное количество монет для выдачи сдачи с достоинством пять и десять рублей.
3. После этого выводится сообщение о готовности кофемашины к работе.

После готовности кофемашины выполняются действия:

- ввод денег достоинством 5, 10, 50 или 100 рублей. При вводе денег осуществляется суммирование;
- выбор кофе, если денег достаточно, то выдается кофе и сдача, при наличии. Выводится сообщение о готовности кофемашины к работе;
- выбор кофе, если денег недостаточно, сообщает о недостаточности средств;

- возврат денег, возвращаются все внесенные средства и выводится сообщение о готовности кофемашины к работе.

Устройство возврата сдачи может вернуть только монеты достоинством 5 и 10 рублей.

После ввода купюр достоинством 50 или 100 рублей проверяется возможность возврата внесенной суммы. Если монет с достоинством 5 и 10 рублей недостаточно, то купюры 50 или 100 не принимаются.

Возврат денег или сдача выдается максимальным количеством монет достоинством 10 рублей.

Нажатие на кнопки пульта управления и подача денег моделируется посредством клавиатурного ввода. Ввод делится на команды:

- «натуральное число кратное 5» – ввод денег;
- Coffee «наименование кофе» – нажатие кнопки сорта кофе (выбора кофе);
- Refund money – нажатие кнопки «вернуть деньги»;
- Cancel – завершение работы системы.

Отображение текста состояния кофемашины и результата операции моделируется посредством вывода на консоли.

Построить систему, которая использует объекты:

1. Объект «система».
2. Объект для чтения команд и данных. Считывает данные для подготовки и настройки кофемашины. После чтения очередной порции данных для настройки или данных команды, объект выдает сигнал с текстом полученных данных. Все данные настройки и данные команды синтаксически корректны.
3. Объект пульта управления, для отработки нажатия кнопок выбора кофе. Объект после нажатия кнопки анализирует достаточность средств и

выдает соответствующий сигнал.

4. Объект, моделирующий устройство приема денег. После принятия очередной купюры производит суммирование и выдает сигнал, содержащий сумму введенных денег для отображения на экран.
5. Объект, моделирующий устройство возврата денег. Выдает сигнал, содержащий количество возвращаемой суммы. После выводится сообщение о готовности кофемашины к работе.
6. Объект, моделирующий устройство выдачи кофе. Выдает сигнал, содержащий текст. После выдачи кофе выдает сигнал о готовности кофемашины к работе.
7. Объект для вывода состояния или результата операции кофемашины на консоль.

Написать программу, реализующую следующий алгоритм:

1. Вызов метода объекта-приложения BuildTree().

1.1. Построение дерева иерархии объектов.

1.2. Установка связей сигналов и обработчиков между объектами.

2. Вызов метода объекта-приложения StartApp().

2.1. Приведение всех объектов в состояние готовности.

2.2. Цикл для обработки вводимых данных для настройки и команд.

2.2.1. Выдача сигнала объекту для ввода команды.

2.2.2. Отработка команды.

2.3. После ввода команды «Cancel» завершить работу.

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 программы.

1.1 Описание входных данных

Первая строка.

«натуральное число» «название кофе 1» … «название кофе n»

Задает количество сортов кофе и их наименования в количестве не более 5. Выполняется операция загрузки кофе.

Вторая строка содержит целые числа кратные 5 в количестве сортов кофе. Каждое значение соответствует цене сорта кофе согласно индексу (порядку ввода). Выполняется операция настройки цен.

«натуральное число» «натуральное число» ….

Третья строка содержит исходное количество монет для выдачи сдачи. Выполняется операция первоначальной загрузки монет для сдачи.

«натуральное число» «натуральное число»

Первое число – количество пятирублевых монет, второе число – количество десятирублевых монет.

Последующие строки содержат команды (нажатия на кнопки или подача денег).

Подача денег, число 5, 10, 50 или 100

«натуральное число»

Возврат денег

Refund money

Выбор кофе

Coffee «наименование кофе»

Последняя команда присутствует всегда

Cancel

Пример ввода:

3 Espresso Americano Cappuchino

25 50 50

3 5

50

Coffee Cappuchino

10

10

10

Coffee Espresso

5

5

Refund money

Cancel

1.2 Описание выходных данных

Шаблоны текстов, которые отображаются на консоли:

Готов к работе, отображение в начале работы системы, после завершения загрузки кофемашины. Также отображается после завершения очередной операции и готовности кофемашины для обслуживания нового клиента.

Ready to work

Сумма после ввода очередной монеты или купюры.

The amount: «сумма денег»

Сообщение о готовности кофе

Take the coffee «наименование кофе»

Сообщение для получения сдачи:

Take the change: 10 * «количество десятирублевых монет» rub., 5 * «количество пятирублевых монет» rub.

Сообщение о недостаточности средств

There is not enough money

Сообщение для получения введенных денег обратно:

Take the money: 10 * «количество десятирублевых монет» rub., 5 * «количество пятирублевых монет» rub.

Сообщение о возврате 50 или 100 рублевой купюры:

Take the money back, no change

Сообщение о завершении работы кофемашины:

Turned off

Пример вывода:

Ready to work

The amount: 50

Take the coffee Cappuchino

Ready to work

The amount: 10

The amount: 20

The amount: 30

Take the coffee Espresso

Take the change: 10 * 0 rub., 5 * 1 rub.

Ready to work

The amount: 5

The amount: 10

Take the money: 10 * 1 rub., 5 * 0 rub.

Ready to work

Turned off

2 МЕТОД РЕШЕНИЯ

Для решения задачи используются:

1. Объекты стандартных потоков ввода и вывода cin и cout соответственно. Используются для ввода с клавиатуры и вывода на экран.
2. Условный оператор if .. else. Используется для ветвления алгоритма.
3. Оператор цикла со счетчиком for. Используется для циклического вызова набора действий.
4. Оператор цикла с предусловием while. Используется для циклического вызова набора действий.
5. Методы find и rfind класса string. Используются для поиска разделителей строки.
6. Метод substr класса string. Используется для выделения подстроки из строки.
7. Метод erase класса string. Используется для удаления подстрок в строке.
8. Параметризированное макроопределение препроцессора. Используется для получения указателей на методы сигнала и обработчика объекта.
9. Функция atoi библиотеки stdlib.h. Используется для преобразования строкового формата в целочисленный.
10. Функция to_string библиотеки string. Используется для преобразования целочисленного формата в строковый.
11. Объекты sys, settings, checker, taker, giver, coffeegiver, объект класса Application. Используются для построения системы.

Класс Application:

- поля / свойства:
    - свойство вектора всех объектов:
        - наименование – ukaz;
        - тип – вектор указателей на объекты класса Base;
        - модификатор доступа – public.
- функционал:
    - метод BuildTree() – метод постройки дерева иерархии объектов и связей сигнал-обработчик между ними;
    - метод StartApp() – метод запуска приложения.

Класс Base:

- поля / свойства:
    - поле имени объекта:
        - наименование – name;
        - тип – строковый;
        - модификатор доступа – protected.
    - свойство готовности объекта:
        - наименование – status;
        - тип – целочисленный;
        - модификатор доступа – protected.
    - свойство вектора детей объекта:
        - наименование – children;
        - тип – вектор указателей на объекты класса Base;
        - модификатор доступа – protected.

- свойство родителя объекта:
    - наименование – parent;
    - тип – указатель на объект класса Base;
    - модификатор доступа – private.
- поле номера класса объекта:
    - наименование – class_number;
    - тип – целочисленный;
    - модификатор доступа – private.
- свойство связи сигнал-обработчик:
    - наименование – o_sh;
    - тип – структура{сигнал signal, указатель на целевой объект класса Base connected_obj, обработчик handler};
    - модификатор доступа – public.
- свойство вектора связей:
    - наименование – connects;
    - тип – вектор указателей на объекты структуры o_sh;
    - модификатор доступа – public.

- функционал:
    - конструктор Base(указатель на объект класса Base parent, строка name, целочисленная переменная class_number) – параметризированный конструктор, ставящий объект на свое место в дереве иерархии;
    - метод SetName(строка name) – параметризированный метод для определения имени объекта;
    - метод PrintTree() – метод вывода дерева иерархии объектов на экран;

- метод SetParent(указатель на объект класса Base parent) – параметризированный метод для переопределения головного объекта для текущего в дереве иерархии;
- метод GetHead() – метод для получения указателя на головной объект текущего объекта;
- метод ObjectByName(строка name, ссылка на вектор указателей объектов класса Base vec, указатель на объект класса Base current) – параметризированный метод, возвращающий указатель на объект по его координате;
- метод GetName() – метод для получения имени объекта;
- метод PrintTreeAndStatus() – метод вывода дерева иерархии объектов и отметок их готовности;
- метод SetStatus(целочисленная переменная status) – параметризированный метод, устанавливающий готовность объекта;
- метод SetConnect(сигнал p_signal, указатель на объект класса Base p_object, обработчик p_ob_hendler) – параметризированный метод установки связи между сигналом текущего объекта и обработчиком целевого объекта;
- метод DeleteConnect(сигнал p_signal, указатель на объект класса Base p_object, обработчик p_ob_hendler) – параметризированный метод удаления (разрыва) связи между сигналом текущего объекта и обработчиком целевого объекта;
- метод EmitSignal(сигнал p_signal, ссылка на строку s_command) – параметризированный метод выдачи сигнала от текущего объекта с передачей строковой переменной;

- метод GetCoordinates() – метод получения абсолютного пути текущего объекта;
- метод GetClassNumber() – метод получения номера класса объекта.

Класс ChangeChecker:

- поля / свойства:
  - свойство суммы денег:
    - наименование – money_sum;
    - тип – целочисленный;
    - модификатор доступа – private.
  - свойство количества пятирублевых монет:
    - наименование – change5;
    - тип – целочисленный;
    - модификатор доступа – private.
  - свойство количества десятирублевых монет:
    - наименование – change10;
    - тип – целочисленный;
    - модификатор доступа – private.
- функционал:
  - метод RefundMoney() – метод возврата денег;
  - метод Signal1(ссылка на строку s) – параметризированный метод сигнала выдачи денег;
  - метод Handler1(строка s) – параметризированный метод-обработчик количества монет, доступных для выдачи;
  - метод Handler2(строка s) – параметризированный метод-обработчик пересчета количества монет, доступных для выдачи;

- метод Handler3(строка msg) – параметризированный метод-обработчик пересчета суммы введенных денег.

Класс CoffeeGiver:

- поля / свойства:
  - свойство количества дачи:
    - наименование – change;
    - тип – целочисленный;
    - модификатор доступа – private.
  - свойство сорта кофе:
    - наименование – sort;
    - тип – строковый;
    - модификатор доступа – private.
- функционал:
  - метод GiveCoffee() – метод выдачи кофе;
  - метод GiveChange() – метод выдачи сдачи;
  - метод Signal1(ссылка на строку s) – параметризированный метод сигнала выдачи кофе;
  - метод Signal2(ссылка на строку s) – параметризированный метод сигнала выдачи сдачи;
  - метод Handler(строка s) – параметризированный метод-обработчик выдачи сдачи и определенного сорта кофе.

Класс MoneyChecker:

- поля / свойства:
    - свойство суммы денег:
        - наименование – money_sum;
        - тип – целочисленный;
        - модификатор доступа – private.
- функционал:
    - метод CheckMoney(строка sort, целочисленная переменная cost) – параметризированный метод анализа достаточности средств;
    - метод Signal(ссылка на строку s) – параметризированный метод сигнала достаточности средств;
    - метод Handler(строка s) – параметризированный метод-обработчик введенных денег.

Класс MoneyTaker:

- поля / свойства:
    - свойство количества введенных денег:
        - наименование – money_sum;
        - тип – целочисленный;
        - модификатор доступа – private.
- функционал:
    - метод TakeMoney(целочисленные переменные money, change5, change10) – параметризированный метод приема денег;
    - метод Signal(ссылка на строку s) – параметризированный метод сигнала количества введенных денег;
    - метод Handler(строка s) – параметризированный метод-обработчик обнуления суммы введенных денег.

Класс System:

- поля / свойства:
    - поле количества сортов кофе:
        - наименование – number_of_sorts;
        - тип – целочисленный;
        - модификатор доступа – private.
    - поле массива названий сортов кофе:
        - наименование – sorts;
        - тип – указатель на массив строк;
        - модификатор доступа – private.
    - поле массива цен на кофе:
        - наименование – costs;
        - тип – указатель на целочисленный массив;
        - модификатор доступа – private.
- функционал:
    - метод SetSettings(ссылка на вектор указателей объектов класса Base allobj) – параметризированный метод чтения команд и данных;
    - метод Signal1(ссылка на строку msg) – параметризированный метод сигнала монет, доступных для сдачи;
    - метод SignalReady(ссылка на строку msg) – параметризированный метод сигнала готовности кофемашины.

Иерархия наследования классов:

Таблица 1 – Иерархия наследования классов

№	Имя класса	Классы–наследники	Модификатор доступа при наследовании	Описание	Номер	Комментарий
1	Base			Базовый класс в иерархии классов. Содержит основные поля и методы.		
		Application	public		2	
		ChangeChecker	public		4	
		CoffeeGiver	public		5	
		MoneyChecker	public		6	
		MoneyTaker	public		7	
		System	public		3	
2	Application			Класс корневого объекта (приложения).		
3	System			Класс объекта-системы и объекта чтения команд и данных.		
4	ChangeChecker			Класс объекта возврата денег, подчиненного корневому объекту.		
5	CoffeeGiver			Класс объекта выдачи кофе, подчиненного корневому объекту.		
6	MoneyChecker			Класс объекта, анализирующего достаточность средств, подчиненного корневому объекту.		
7	MoneyTaker			Класс объекта приема денег, подчиненного корневому объекту.		

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 функции main

Функционал: основной алгоритм программы.

Параметры: отсутствуют.

Возвращаемое значение: целочисленный тип данных – код возврата.

Алгоритм функции представлен в таблице 2.

Таблица 2 – Алгоритм функции main

№	Предикат	Действия	№ перехода
1		Создание объекта app класса Application с параметрами nullptr, "", 1	2
2		Вызов метода BuildTree объекта app	3
3		Возврат функцией значения, возвращенного методом StartApp объекта app	∅

3.2 Алгоритм метода BuildTree класса Application

Функционал: метод постройки дерева иерархии объектов и связей сигнал-обработчик между ними.

Параметры: отсутствуют.

Возвращаемое значение: отсутствует.

Алгоритм метода представлен в таблице 3.

Таблица 3 – Алгоритм метода BuildTree класса Application

№	Предикат	Действия	№ перехода
1		Создание диннамических объектов sys, settings, checker, taker, giver, coffegiver классов System, MoneyChecker, MoneyTaker, ChangeChecker, CoffeeGiver и постройка дерева иерархии объектов	2
2		Установка связи между объектами settings и giver, и их методами Signal1 и Handler1	3
3		Установка связи между объектами taker и checker, и их методами Signal и Handler	4
4		Установка связи между объектами checker и coffeegiver, и их методами Signal и Handler	5
5		Установка связи между объектами coffeegiver и giver, и их методами Signal2 и Handler2	6
6		Установка связи между объектами coffeegiver и taker, и их методами Signal2 и Handler	7
7		Установка связи между объектами giver и taker, и их методами Signal1 и Handler	8
8		Установка связи между объектами taker и giver, и их методами Signal и Handler3	∅

3.3 Алгоритм метода StartApp класса Application

Функционал: метод запуска приложения.

Параметры: отсутствуют.

Возвращаемое значение: целочисленный тип данных – код возврата.

Алгоритм метода представлен в таблице 4.

Таблица 4 – Алгоритм метода StartApp класса Application

№	Предикат	Действия	№ перехода
1		Приведение всех объектов в состояние готовности	2

№	Предикат	Действия	№ перехода
2		Объявление указателя на объект класса Base с инициализацией tmp=значение, возвращенное методом ObjectByName с параметрами "//settings_and_commands", ukaz, указатель на текущий объект	3
3		Объявление указателя на объект класса System с инициализацией settings=приведенный к типу: указатель на объект класса System; объект tmp	4
4		Вызов метода SetSettings объекта settings с параметром ukaz	∅

3.4 Алгоритм метода RefundMoney класса ChangeChecker

Функционал: метод возврата денег.

Параметры: отсутствуют.

Возвращаемое значение: отсутствует.

Алгоритм метода представлен в таблице 5.

Таблица 5 – Алгоритм метода RefundMoney класса ChangeChecker

№	Предикат	Действия	№ перехода
1		Объявление строковой переменной с инициализацией msg=""	2
2		Вызов метода EmitSignal текущего объекта с параметрами: метод Signal1 класса ChangeChecker, msg	∅

3.5 Алгоритм метода Handler1 класса ChangeChecker

Функционал: параметризированный метод-обработчик количества монет, доступных для выдачи.

Параметры: строковый параметр change_sum – сумма сдачи.

Возвращаемое значение: отсутствует.

Алгоритм метода представлен в таблице 6.

Таблица 6 – Алгоритм метода Handler1 класса ChangeChecker

№	Предикат	Действия	№ перехода
1		Присвоение полям change5, change10 класса ChangeChecker количеств пяти- и десятирублевых монет, доступных для выдачи	∅

3.6 Алгоритм метода Handler2 класса ChangeChecker

Функционал: параметризированный метод-обработчик пересчета количества монет, доступных для выдачи.

Параметры: строковый параметр decr_change – число выданных монет.

Возвращаемое значение: отсутствует.

Алгоритм метода представлен в таблице 7.

Таблица 7 – Алгоритм метода Handler2 класса ChangeChecker

№	Предикат	Действия	№ перехода
1		Пересчет количества пяти- и десятирублевых монет, доступных для выдачи	∅

3.7 Алгоритм метода Handler3 класса ChangeChecker

Функционал: параметризированный метод-обработчик пересчета суммы введенных денег.

Параметры: строковый параметр msg – пересчитанное значение суммы введенных денег.

Возвращаемое значение: отсутствует.

Алгоритм метода представлен в таблице 8.

Таблица 8 – Алгоритм метода Handler3 класса ChangeChecker

№	Предикат	Действия	№ перехода
1		Присвоение свойству money_sum текущего объекта пересчитанного значения суммы введенных денег	∅

3.8 Алгоритм метода Signal1 класса ChangeChecker

Функционал: параметризированный метод сигнала выдачи денег.

Параметры: ссылка на строковый параметр msg – пустая строка.

Возвращаемое значение: отсутствует.

Алгоритм метода представлен в таблице 9.

Таблица 9 – Алгоритм метода Signal1 класса ChangeChecker

№	Предикат	Действия	№ перехода
1		Расчет количества пяти- и десятирублевых монет, необходимых для выдачи	2
2	Сумма пяти- и десятирублевых монет, необходимых для выдачи, больше нуля	Вывод на экран "Take the money: 10 * ", количество десятирублевых монет для выдачи," rub., 5 * ", количество пятирублевых монет для выдачи, " rub.\n"	3
			4
3		Вывод на экран "Ready to work\n"	4
4		Присвоение msg значения количеств пяти- и десятирублевых монет	∅

3.9 Алгоритм метода GiveCoffee класса CoffeeGiver

Функционал: метод выдачи кофе.

Параметры: отсутствуют.

Возвращаемое значение: отсутствует.

Алгоритм метода представлен в таблице 10.

Таблица 10 – Алгоритм метода GiveCoffee класса CoffeeGiver

№	Предикат	Действия	№ перехода
1		Вызов метода EmitSignal текущего объекта с параметрами: метод Signal1 объекта CoffeeGiver, значение свойства sort	2
2		Вызов метода GiveChange текущего объекта	3
3	Значение свойства change равно нулю	Вывод на экран "Ready to work\n"	∅
			∅

3.10 Алгоритм метода GiveChange класса CoffeeGiver

Функционал: метод выдачи сдачи.

Параметры: отсутствуют.

Возвращаемое значение: отсутствует.

Алгоритм метода представлен в таблице 11.

Таблица 11 – Алгоритм метода GiveChange класса CoffeeGiver

№	Предикат	Действия	№ перехода
1		Объявление строковой переменной с инициализацией msg=приведенное к строковому типу значение свойства change	2
2		Вызов метода EmitSignal текущего объекта с параметрами: метод Signal2 класса CoffeeGiver, msg	∅

3.11 Алгоритм метода Handler класса CoffeeGiver

Функционал: параметризированный метод-обработчик выдачи сдачи и определенного сорта кофе.

Параметры: строковый параметр msg – сдача и сорт выдаваемого кофе.

Возвращаемое значение: отсутствует.

Алгоритм метода представлен в таблице 12.

Таблица 12 – Алгоритм метода Handler класса CoffeeGiver

№	Предикат	Действия	№ перехода
1		Привоение свойствам change и sort значений количества выдаваемой сдачи и сорта выдаваемого кофе	2
2		Вызов метода GiveCoffee текущего объекта	∅

3.12 Алгоритм метода Signal1 класса CoffeeGiver

Функционал: параметризированный метод сигнала выдачи кофе.

Параметры: ссылка на строковый параметр msg – сорт выдаваемого кофе.

Возвращаемое значение: отсутствует.

Алгоритм метода представлен в таблице 13.

Таблица 13 – Алгоритм метода Signal1 класса CoffeeGiver

№	Предикат	Действия	№ перехода
1		Вывод на экран "Take the coffee ", msg, "\n"	∅

3.13 Алгоритм метода Signal2 класса CoffeeGiver

Функционал: параметризированный метод сигнала выдачи сдачи.

Параметры: ссылка на строковый параметр msg – общая сумма сдачи.

Возвращаемое значение: отсутствует.

Алгоритм метода представлен в таблице 14.

Таблица 14 – Алгоритм метода Signal2 класса CoffeeGiver

№	Предикат	Действия	№ перехода
1		Расчет количества пяти- и десятирублевых монет, необходимых для сдачи	2
2	Сумма пяти- и десятирублевых монет, необходимых для сдачи, больше нуля	Вывод на экран "Take the money: 10 * ", количество десятирублевых монет для сдачи," rub., 5 * ", количество пятирублевых монет для сдачи, " rub.\n"	3
			4
3		Вывод на экран "Ready to work\n"	4
4		Присвоение msg значения количеств пяти- и десятирублевых монет	∅

3.14 Алгоритм метода CheckMoney класса MoneyChecker

Функционал: параметризированный метод анализа достаточности средств.

Параметры: строковый параметр sort – сорт кофе, целочисленный параметр cost – цена кофе.

Возвращаемое значение: отсутствует.

Алгоритм метода представлен в таблице 15.

Таблица 15 – Алгоритм метода CheckMoney класса MoneyChecker

№	Предикат	Действия	№ перехода
1	Значение cost меньше или равно значению свойства money_sum		2
		Вывод на экран "There is not enough money\n"	∅
2		Объявление строковой переменной с инициализацией msg=приведенное к строковому типу значение разности money_sum и cost+"	3

№	Предикат	Действия	№ перехода
		"+значение свойства sort	
3		Вызов метода EmitSignal текущего объекта с параметрами: метод Signal класса MoneyChecker, msg	∅

3.15 Алгоритм метода Handler класса MoneyChecker

Функционал: параметризированный метод-обработчик введенных денег.

Параметры: строковый параметр money_sum – число введенных денег.

Возвращаемое значение: отсутствует.

Алгоритм метода представлен в таблице 16.

Таблица 16 – Алгоритм метода Handler класса MoneyChecker

№	Предикат	Действия	№ перехода
1		Присвоение свойству money_sum текущего объекта значения money_sum приведенного к целочисленному типу данных	∅

3.16 Алгоритм метода Signal класса MoneyChecker

Функционал: параметризированный метод сигнала достаточности средств.

Параметры: ссылка на строковый параметр msg – сдача и сорт выдаваемого кофе.

Возвращаемое значение: отсутствует.

Алгоритм метода представлен в таблице 17.

Таблица 17 – Алгоритм метода Signal класса MoneyChecker

№	Предикат	Действия	№ перехода
1		Передача связанному обработчику значения msg	∅

3.17 Алгоритм метода TakeMoney класса MoneyTaker

Функционал: параметризированный метод приема денег.

Параметры: целочисленные параметры money – количество вводимых денег, change5 – количество доступных пятирублевых монет, change10 – количество доступных десятирублевых монет.

Возвращаемое значение: отсутствует.

Алгоритм метода представлен в таблице 18.

Таблица 18 – Алгоритм метода TakeMoney класса MoneyTaker

№	Предикат	Действия	№ перехода
1	Количество денег, доступных для сдачи больше или равно сумме money_sum и money	Присвоение свойству money_sum текущего объекта значения money_sum+money	2
		Вывод на экран "Take the money back, no change\n"	∅
2		Объявление строковой переменной с инициализацией msg=значение money_sum приведенное к строковому типу	3
3		Вызов метода EmitSignal текущего объекта с параметрами: метод Signal класса MoneyTaker, msg	∅

3.18 Алгоритм метода Signal класса MoneyTaker

Функционал: параметризированный метод сигнала количества введенных денег.

Параметры: ссылка на строковый параметр s – пересчитанное значение суммы введенных денег.

Возвращаемое значение: отсутствует.

Алгоритм метода представлен в таблице 19.

Таблица 19 – Алгоритм метода Signal класса MoneyTaker

№	Предикат	Действия	№ перехода
1		Вывод на экран "The amount: ", значение свойства money_sum,"\n"	∅

3.19 Алгоритм метода Handler класса MoneyTaker

Функционал: параметризированный метод-обработчик обнуления суммы введенных денег.

Параметры: строковый параметр msg – общая сумма сдачи.

Возвращаемое значение: отсутствует.

Алгоритм метода представлен в таблице 20.

Таблица 20 – Алгоритм метода Handler класса MoneyTaker

№	Предикат	Действия	№ перехода
1		Присвоение свойству money_sum нуля	∅

3.20 Алгоритм метода Signal1 класса System

Функционал: параметризированный метод сигнала монет, доступных для сдачи.

Параметры: ссылка на строковый параметр msg – количество пяти- и десятирублевых монет, доступных для сдачи.

Возвращаемое значение: отсутствует.

Алгоритм метода представлен в таблице 21.

Таблица 21 – Алгоритм метода Signal1 класса System

№	Предикат	Действия	№ перехода
1		Передача связанному обработчику значения msg	∅

3.21 Алгоритм метода SignalReady класса System

Функционал: параметризированный метод сигнала готовности кофемашины.

Параметры: ссылка на строковый параметр msg – сообщение о готовности кофемашины.

Возвращаемое значение: отсутствует.

Алгоритм метода представлен в таблице 22.

Таблица 22 – Алгоритм метода SignalReady класса System

№	Предикат	Действия	№ перехода
1		Вывод на экран значения msg	∅

3.22 Алгоритм метода SetSettings класса System

Функционал: параметризированный метод чтения команд и данных.

Параметры: ссылка на контейнер класса vector с значениями указателей на объекты класса Base – вектор со всеми объектами, созданными в программе allobj.

Возвращаемое значение: отсутствует.

Алгоритм метода представлен в таблице 23.

Таблица 23 – Алгоритм метода SetSettings класса System

№	Предикат	Действия	№ перехода
1		Считывание с клавиатуры значения свойства number_of_sorts	2
2		Создание динамических массивов sorts, costs длиной number_of_costs	3
3		Считывание с клавиатуры значений элементов массива sorts	4
4		Считывание с клавиатуры значений элементов массива costs	5

№	Предикат	Действия	№ перехода
5		Объявление целочисленных переменных change5, change10	6
6		Считывание с клавиатуры значений переменных change5, change10	7
7		Объявление строковой переменной с инициализацией msg=значения change5, change10, приведенные к строковому типу данных, разделенные пробелом	8
8		Вызов метода EmitSignal текущего объекта с параметрами: метод Signal1 класса System, msg	9
9		Объявление строковой переменной с инициализацией msg2="Ready to work"	10
10		Вызов метода EmitSignal текущего объекта с параметрами: метод SignalReady класса System, msg2	11
11		Создание контейнера класса vector с указателями на объекты класса Base; с инициализацией ukaz=allobj	12
12		Объявление строковой переменной vvod	13
13		Объявление указателя на объект класса Base с инициализацией ob1=значение, возвращенное методом ObjectByName с параметрами "//money_taker",ukaz,укзатель на текущий объект	14
14		Объявление указателя на объект класса MoneyTaker с инициализацией obj1=приведенный к типу: указатель на объект класса MoneyTaker; объект ob1	15
15		Объявление указателя на объект класса Base с инициализацией ob2=значение, возвращенное	16

№	Предикат	Действия	№ перехода
		методом ObjectByName с параметрами "//buttons",ukaz,укзатель на текущий объект	
16		Объявление указателя на объект класса MoneyChecker с инициализацией obj2=приведенный к типу: указатель на объект класса MoneyChecker; объект ob2	17
17		Объявление указателя на объект класса Base с инициализацией ob3=значение, возвращенное методом ObjectByName с параметрами "//money_checker",ukaz,укзатель на текущий объект	18
18		Объявление указателя на объект класса ChangeChecker с инициализацией obj3=приведенный к типу: указатель на объект класса ChangeChecker; объект ob3	19
19	Значение vvod не равно "Cancel"		20
		Вывод на экран "Turned off"	∅
20		Считывание с клавиатуры значения vvod	21
21	Значение, возвращенное методом find объекта vvod с параметром "5" не равно -1 или значение, возвращенное методом find объекта vvod с параметром "1" не равно -1	Вызов метода TakeMoney объекта obj1 с параметрами: значение vvod, приведенное к целочисленному типу, change5, change10	19
			22
22	Значение, возвращенное методом find объекта vvod с параметром "Coffee" не равно -1	Вызов метода CheckMoney Объекта obj2 с параметрами: соответствующий сорт кофе и его цена	19

№	Предикат	Действия	№ перехода
			23
23	Значение vvod равно "Refund money"	Вызов метода RefundMoney объекта obj3	19
			24
24	Значение vvod равно "SHOWTREE"	Вывод на экран дерева иерархии объектов и их готовности	∅
			19

3.23 Алгоритм конструктора класса Base

Функционал: параметризированный конструктор, ставящий объект на свое место в дереве иерархии.

Параметры: указатель на объект класса Base – объект-родитель parent, строковый параметр name – имя объекта, целочисленный параметр class_number – номер класса объекта.

Алгоритм конструктора представлен в таблице 24.

Таблица 24 – Алгоритм конструктора класса Base

№	Предикат	Действия	№ перехода
1		Вызов метода SetName текущего объекта с параметром name	2
2		Вызов метода SetParent текущего объекта с параметром parent	3
3	Значение parent не равно nullptr	Добавление текущего объекта в вектор children объекта parent	4
			4
4		Присвоение полю class_number текущего объекта значения class_number	∅

3.24 Алгоритм метода SetName класса Base

Функционал: параметризированный метод для определения имени объекта.

Параметры: строковый параметр name – имя объекта.

Возвращаемое значение: отсутствует.

Алгоритм метода представлен в таблице 25.

Таблица 25 – Алгоритм метода SetName класса Base

№	Предикат	Действия	№ перехода
1		Присвоение значения name полю name текущего объекта	∅

3.25 Алгоритм метода PrintTree класса Base

Функционал: метод вывода дерева иерархии объектов на экран.

Параметры: отсутствуют.

Возвращаемое значение: отсутствует.

Алгоритм метода представлен в таблице 26.

Таблица 26 – Алгоритм метода PrintTree класса Base

№	Предикат	Действия	№ перехода
1		Объявление целочисленной переменной с инициализацией i=0	2
2	Значение i меньше значения, возвращенного методом size объекта children	Выбор отступа в зависимости от глубины нахождения объекта children[i] в дереве иерархии объектов	3
			∅
3		Вывод на экран переноса на новую строку и выбранного отступа	4
4		Вывод на экран значения, возвращенного методом GetName объекта children[i]	5

№	Предикат	Действия	№ перехода
5		Вызов метода PrintTree объекта children[i]	6
6		Инкрементирование i	2

3.26 Алгоритм метода SetParent класса Base

Функционал: параметризированный метод для переопределения головного объекта для текущего в дереве иерархии.

Параметры: указатель на объект класса Base – объект-родитель parent.

Возвращаемое значение: отсутствует.

Алгоритм метода представлен в таблице 27.

Таблица 27 – Алгоритм метода SetParent класса Base

№	Предикат	Действия	№ перехода
1		Присвоение parent свойству parent текущего объекта	∅

3.27 Алгоритм метода GetHead класса Base

Функционал: метод для получения указателя на головной объект текущего объекта.

Параметры: отсутствуют.

Возвращаемое значение: указатель на объект класса Base – объект-родитель parent.

Алгоритм метода представлен в таблице 28.

Таблица 28 – Алгоритм метода GetHead класса Base

№	Предикат	Действия	№ перехода
1		Возврат методом значения свойства parent	∅

3.28 Алгоритм метода ObjectByName класса Base

Функционал: параметризированный метод, возвращающий указатель на объект по его координате.

Параметры: строковый параметр name – координата объекта, ссылка на контейнер класса vector – вектор всех объектов vec с указателями на объекты класса Base, указатель на объект класса Base – текущий объект current.

Возвращаемое значение: указатель на объект класса Base – объект, найденный по координате.

Алгоритм метода представлен в таблице 29.

Таблица 29 – Алгоритм метода ObjectByName класса Base

№	Предикат	Действия	№ перехода
1	Значение name равно "/"	Возврат методом указателя на корневой объект	∅
			2
2	Значение name равно "."	Возврат методом значения current	∅
			3
3	Значение, возвращенное методом size объекта name больше 1 и name[0] равно '/' и name[1] не равно '/'		4
			7
4		Объявление целочисленной переменной с инициализацией i=0	5
5	Значение i меньше значения, возвращенного методом size объекта vec		6
		Возврат методом nullptr	∅
6	Абсолютная координата объекта vec[i] соответствует	Возврат методом значения vec[i]	∅

№	Предикат	Действия	№ перехода
	координате name		
		Инкрементирование i	5
7	Значение name[0] равно '/' и значение name[1] равно '/'	Вызов метода erase объекта name с параметрами 0,2	8
			11
8		Объявление целочисленной переменной с инициализацией i=0	9
9	Значение i меньше значения, возвращенного методом size объекта vec		10
		Возврат методом nullptr	∅
10	Значение, возвращенное методом GetName объекта vec[i] равно name	Возврат методом значения vec[i]	∅
		Инкрементирование i	9
11	Значение, возвращенное методом size объекта name равно нулю или значение name[0] не равно '/' и значение, возвращенное методом find объекта name с параметром "/" больше нуля		12
		Возврат методом nullptr	∅
12	Значение, возвращенное методом find объекта name с параметром "/" не равно нулю	Присвоение name значения "/"+name	13
			13
13		Объявление целочисленной переменной с	14

№	Предикат	Действия	№ перехода
		инициализацией i=0	
14	Значение i меньше значения, возвращенного методом size объекта vec		15
		Возврат методом nullptr	∅
15	Координата объекта vec[i] относительно объекта current соответствует координате name	Возврат методом значения vec[i]	∅
		Инкрементирование i	14

3.29 Алгоритм метода GetName класса Base

Функционал: метод для получения имени объекта.

Параметры: отсутствуют.

Возвращаемое значение: строковый тип данных – значение поля name объекта.

Алгоритм метода представлен в таблице 30.

Таблица 30 – Алгоритм метода GetName класса Base

№	Предикат	Действия	№ перехода
1		Возврат методом значения поля name	∅

3.30 Алгоритм метода PrintTreeAndStatus класса Base

Функционал: метод вывода дерева иерархии объектов и отметок их готовности.

Параметры: отсутствуют.

Возвращаемое значение: отсутствует.

Алгоритм метода представлен в таблице 31.

Таблица 31 – Алгоритм метода PrintTreeAndStatus класса Base

№	Предикат	Действия	№ перехода
1		Объявление целочисленной переменной с инициализацией i=0	2
2	Значение i меньше значения, возвращенного методом size объекта children	Выбор отступа в зависимости от глубины нахождения объекта children[i] в дереве иерархии объектов	3
			∅
3		Вывод на экран переноса на новую строку и выбранного отступа	4
4		Вывод на экран значения, возвращенного методом GetName объекта children[i], " is "	5
5	Значение свойства status объекта children[i] не равно нулю	Вывод на экран "ready"	6
		Вывод на экран "not ready"	6
6		Вызов метода PrintTree объекта children[i]	7
7		Инкрементирование i	2

3.31 Алгоритм метода SetStatus класса Base

Функционал: параметризированный метод, устанавливающий готовность объекта.

Параметры: целочисленный параметр status – состояние объекта.

Возвращаемое значение: отсутствует.

Алгоритм метода представлен в таблице 32.

Таблица 32 – Алгоритм метода SetStatus класса Base

№	Предикат	Действия	№ перехода
1	Значение status равно нулю	Присвоение свойству status текущего объекта значения status	2
			5
2		Объявление целочисленной переменной с инициализацией i=0	3
3	Значение i меньше значения, возвращенного методом size объекта children	Вызов метода SetStatus объекта children[i] с параметром 0	4
			∅
4		Инкрементирование i	3
5	Выше по дереву иерархии есть объект, значение свойства status которого равно нулю		∅
		Присвоение свойству status текущего объекта значения status	∅

3.32 Алгоритм метода SetConnect класса Base

Функционал: параметризированный метод установки связи между сигналом текущего объекта и обработчиком целевого объекта.

Параметры: указатель на метод сигнала signal – сигнал, указатель connected_obj на объект класса Base – целевой объект, указатель на метод обработчика handler – обработчик.

Возвращаемое значение: отсутствует.

Алгоритм метода представлен в таблице 33.

Таблица 33 – Алгоритм метода SetConnect класса Base

№	Предикат	Действия	№ перехода
1		Объявление указателя на объект структуры o_sh – p_value	2
2		Объявление целочисленной переменной с инициализацией i=0	3
3	Значение i меньше значения, возвращенного методом size объекта connects		4
			6
4	Поля объекта connects[i] равны signal, connected_obj и handler		∅
			5
5		Инкрементирование i	3
6		Создание динамической структуры p_value типа o_sh	7
7		Присвоение полям p_value значений signal, connected_obj, handler	8
8		Вызов метода push_back объекта connects с параметром p_value	∅

3.33 Алгоритм метода DeleteConnect класса Base

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

Параметры: указатель на метод сигнала signal – сигнал, указатель connected_obj на объект класса Base – целевой объект, указатель на метод обработчика handler – обработчик.

Возвращаемое значение: отсутствует.

Алгоритм метода представлен в таблице 34.

Таблица 34 – Алгоритм метода DeleteConnect класса Base

№	Предикат	Действия	№ перехода
1		Объявление целочисленной переменной с инициализацией i=0	2
2	Значение i меньше значения, возвращенного методом size объекта connects		3
			∅
3	Поля объекта connects[i] равны signal, connected_obj и handler	Вызов метода erase объекта connects с параметром connects.begin()+i	∅
			4
4		Инкрементирование i	2

3.34 Алгоритм метода EmitSignal класса Base

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

Параметры: указатель на метод сигнала signal – сигнал, ссылка на строку message – сообщение, передаваемое сигналом.

Возвращаемое значение: отсутствует.

Алгоритм метода представлен в таблице 35.

Таблица 35 – Алгоритм метода EmitSignal класса Base

№	Предикат	Действия	№ перехода
1	Значение status не равно нулю	Вызов метода signal текущего объекта по указателю с параметром message	2
			∅

№	Предикат	Действия	№ перехода
2		Объявление целочисленной переменной с инициализацией i=0	3
3	Значение i меньше значения возвращенного методом size объекта connects		4
			∅
4	Значение поля signal объекта connects[i] равно signal и значение поля status поля connected_obj объекта connects[i] не равно нулю	Вызов метода handler поля connects[i] объекта connected_obj по указателю с параметром message	5
			5
5		Инкрементирование i	3

3.35 Алгоритм метода GetCoordinates класса Base

Функционал: метод получения абсолютного пути текущего объекта.

Параметры: отсутствуют.

Возвращаемое значение: строковый тип данных – абсолютный путь до объекта path.

Алгоритм метода представлен в таблице 36.

Таблица 36 – Алгоритм метода GetCoordinates класса Base

№	Предикат	Действия	№ перехода
1		Объявление указателя на объект класса Base с инициализацией obj=указатель на текущий объект	2
2		Объявление строковой переменной с инициализацией path=""	3
3	Значение возвращенное	Присвоение path значения "/"+значение,	4

№	Предикат	Действия	№ перехода
	методом GetHead объекта obj не равно nullptr	возвращенное методом GetName объекта obj+path	
			5
4		Присвоение obj значения возвращенного методом GetHead объекта obj	3
5	Значение, возвращенное методом size объекта path равно нулю	Возврат методом "/"	∅
		Возврат методом path	∅

3.36 Алгоритм метода GetClassNumber класса Base

Функционал: метод получения номера класса объекта.

Параметры: отсутствуют.

Возвращаемое значение: целочисленный тип данных – значение поля class_number объекта.

Алгоритм метода представлен в таблице 37.

Таблица 37 – Алгоритм метода GetClassNumber класса Base

№	Предикат	Действия	№ перехода
1		Возврат методом значения поля class_number	∅

4 БЛОК-СХЕМЫ АЛГОРИТМОВ

Представим описание алгоритмов в графическом виде на рисунках 1-28.

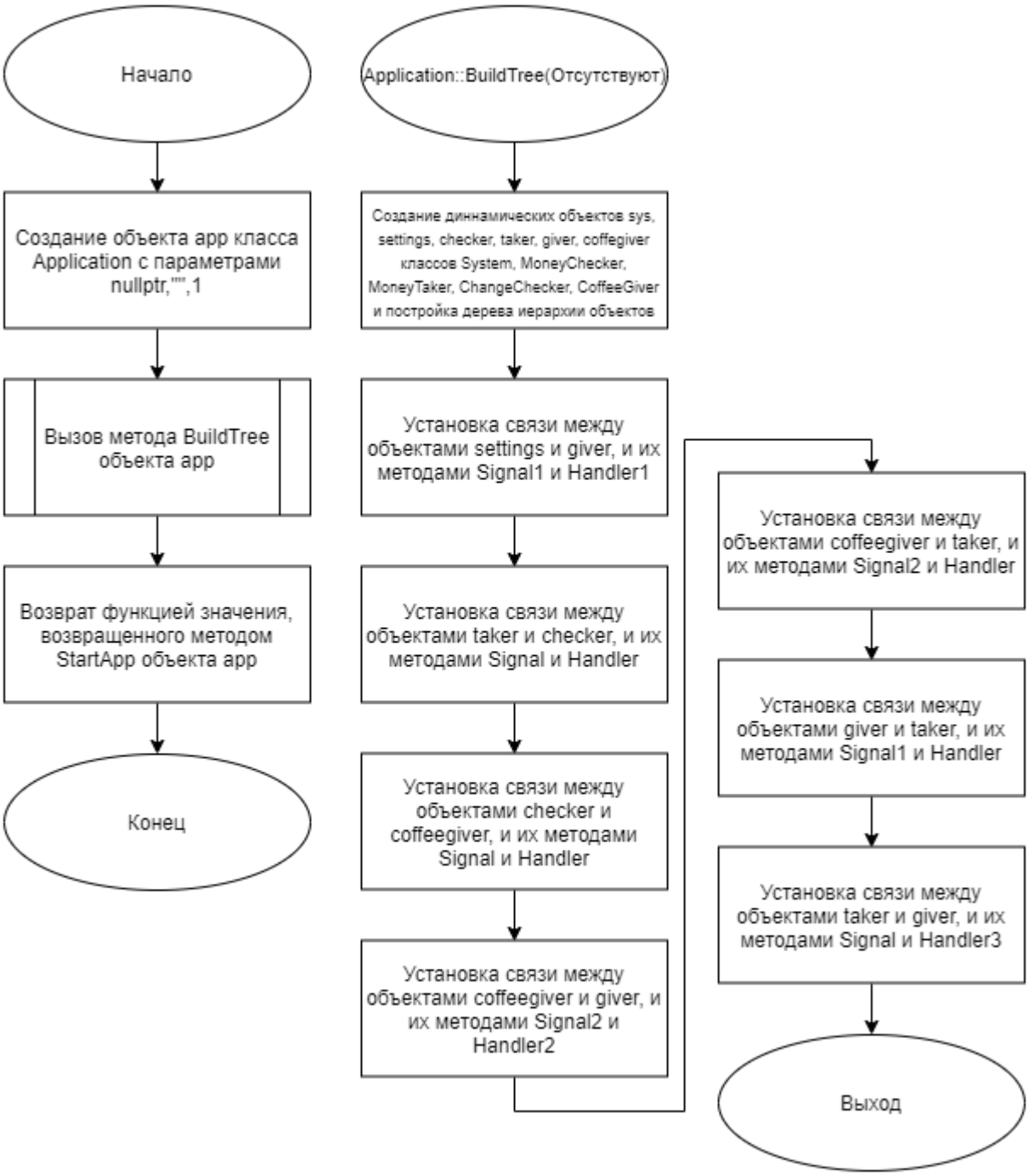

Рисунок 1 – Блок-схема алгоритма

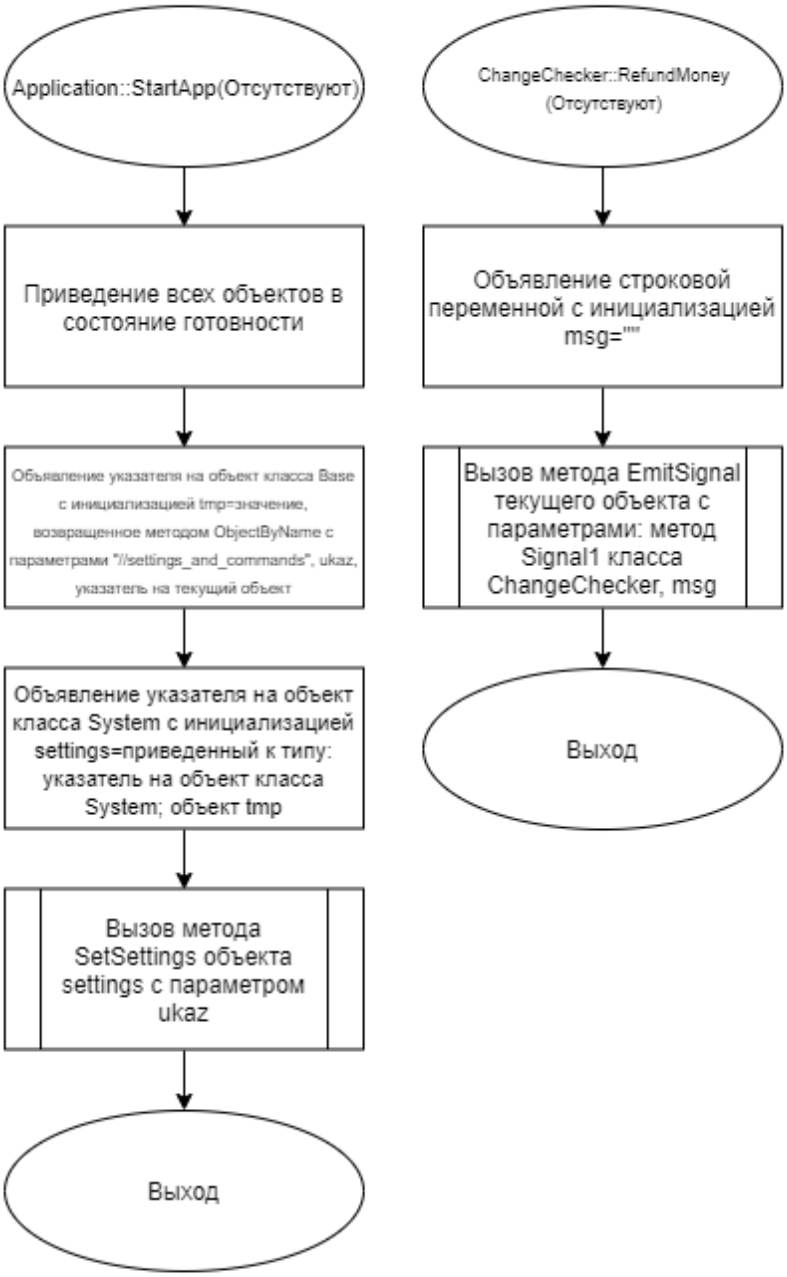

Рисунок 2 – Блок-схема алгоритма

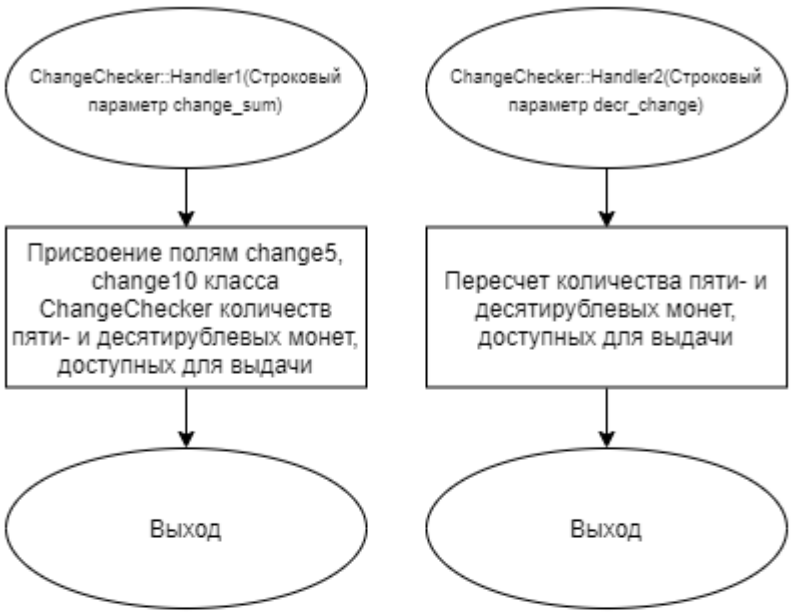

Рисунок 3 – Блок-схема алгоритма

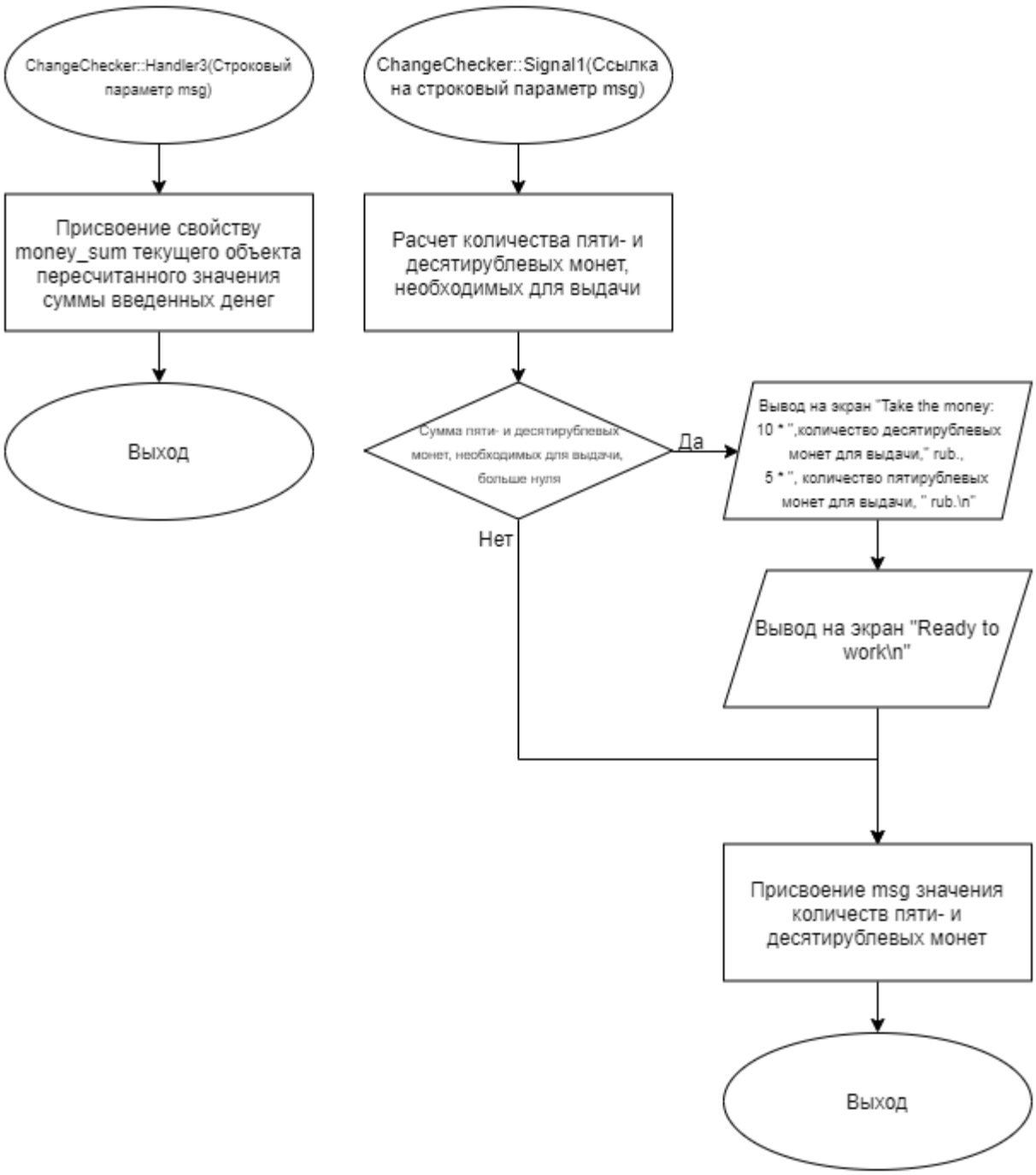

Рисунок 4 – Блок-схема алгоритма

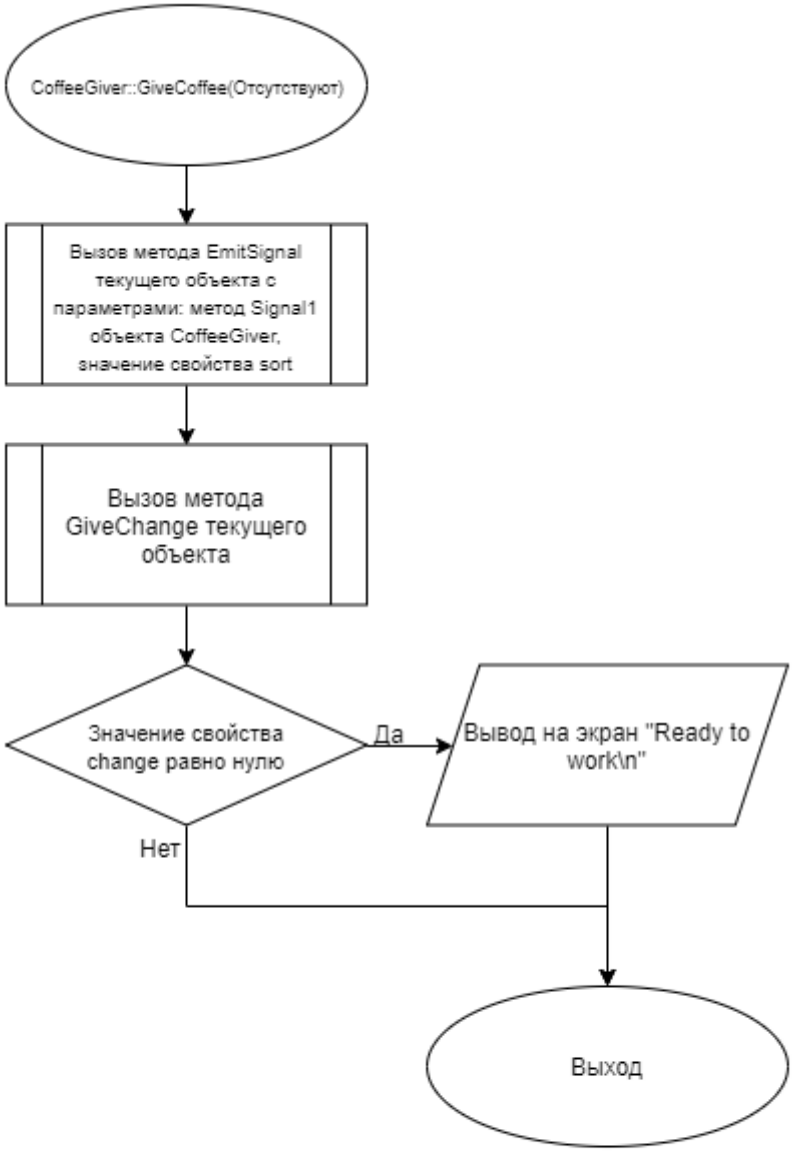

Рисунок 5 – Блок-схема алгоритма

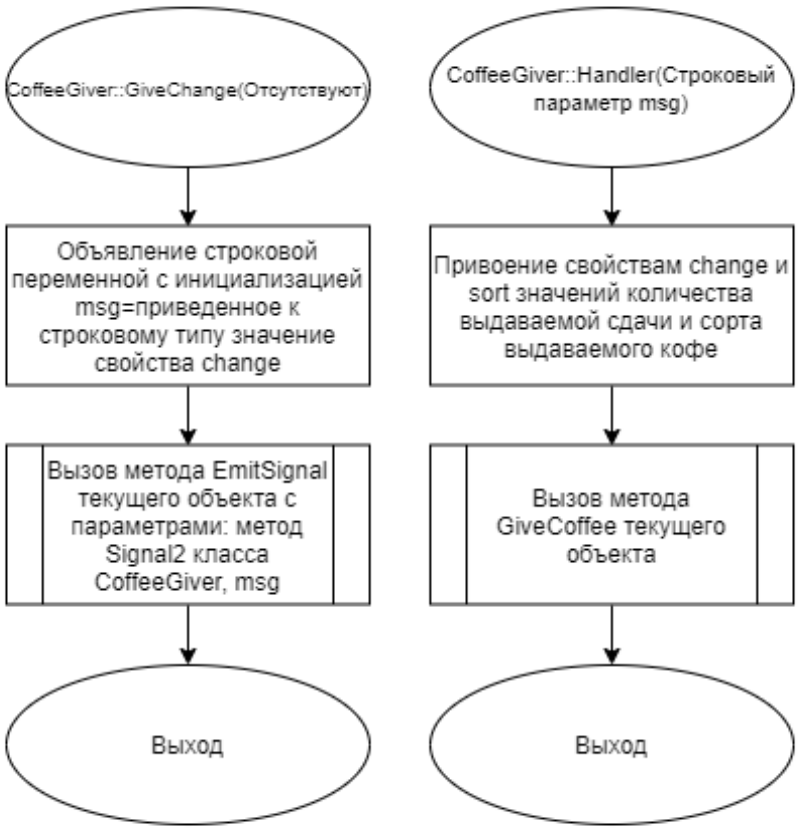

Рисунок 6 – Блок-схема алгоритма

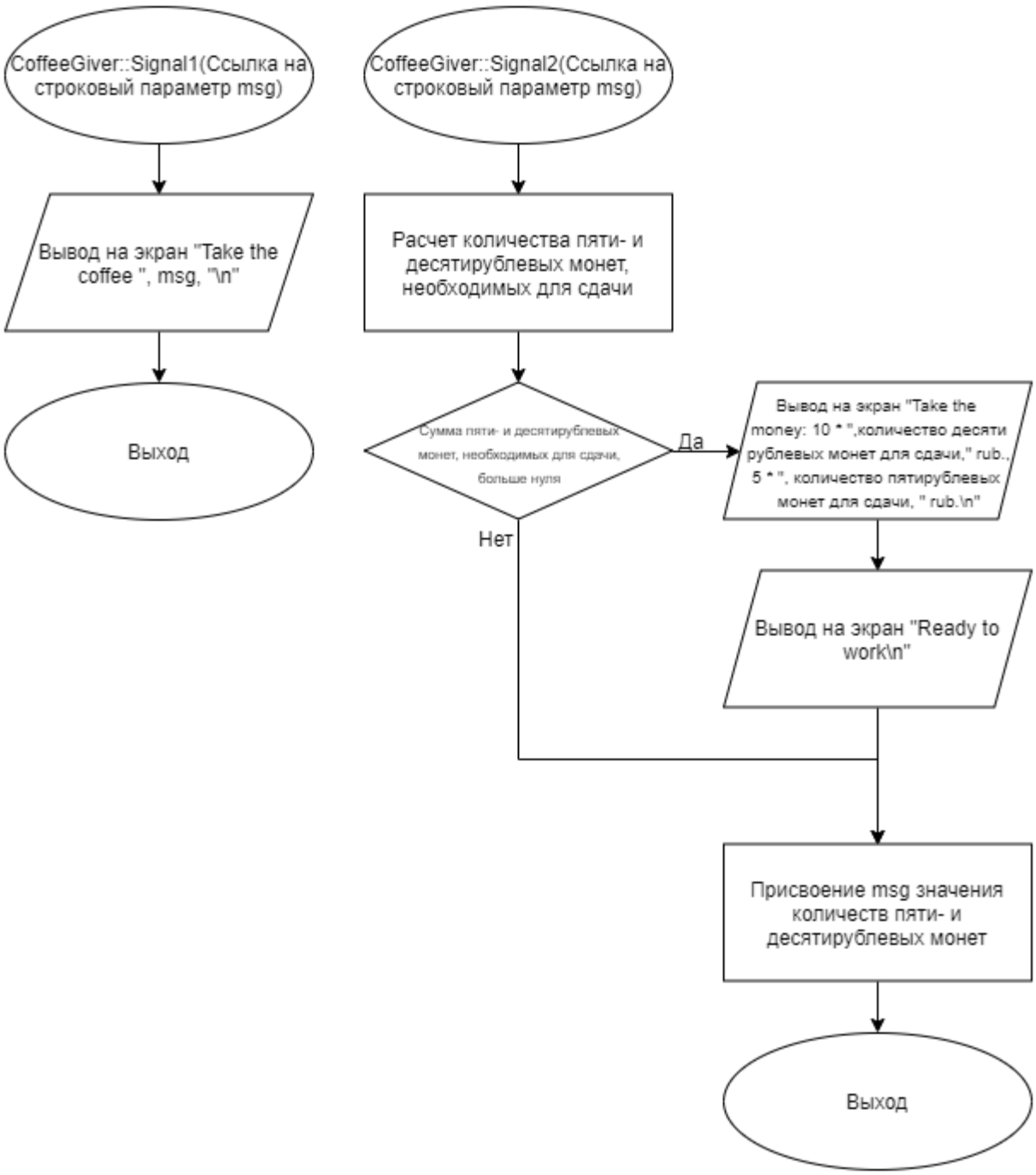

Рисунок 7 – Блок-схема алгоритма

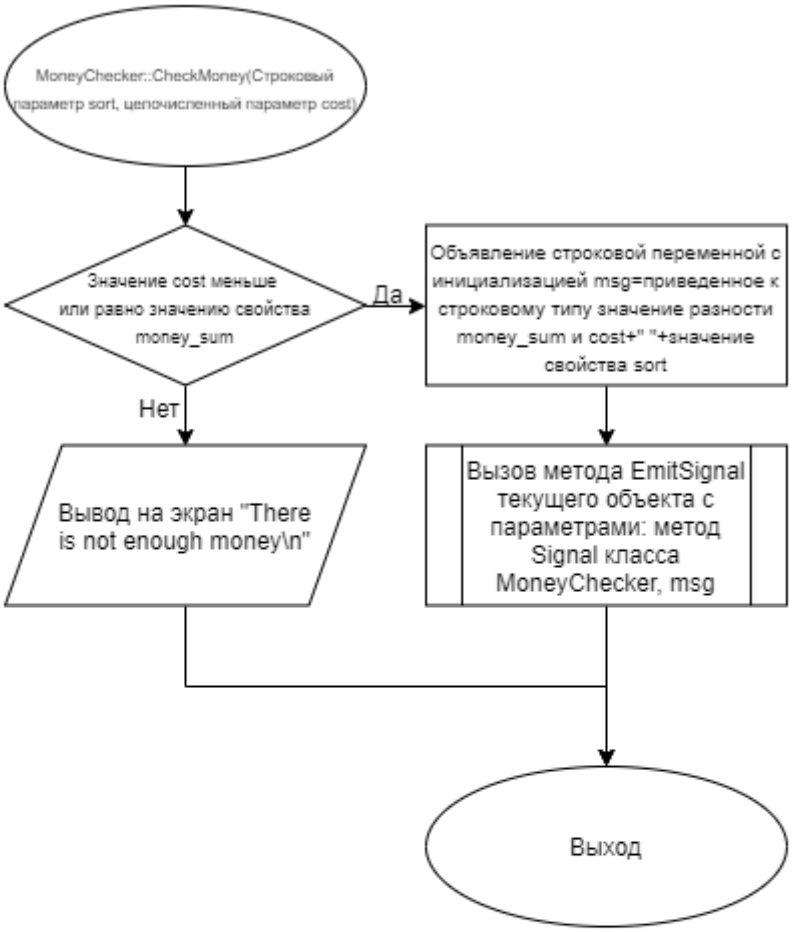

Рисунок 8 – Блок-схема алгоритма

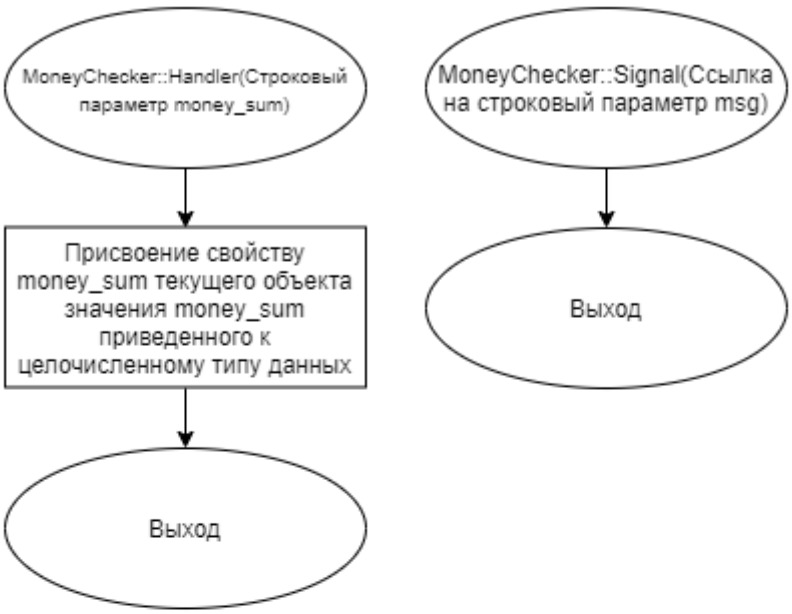

Рисунок 9 – Блок-схема алгоритма

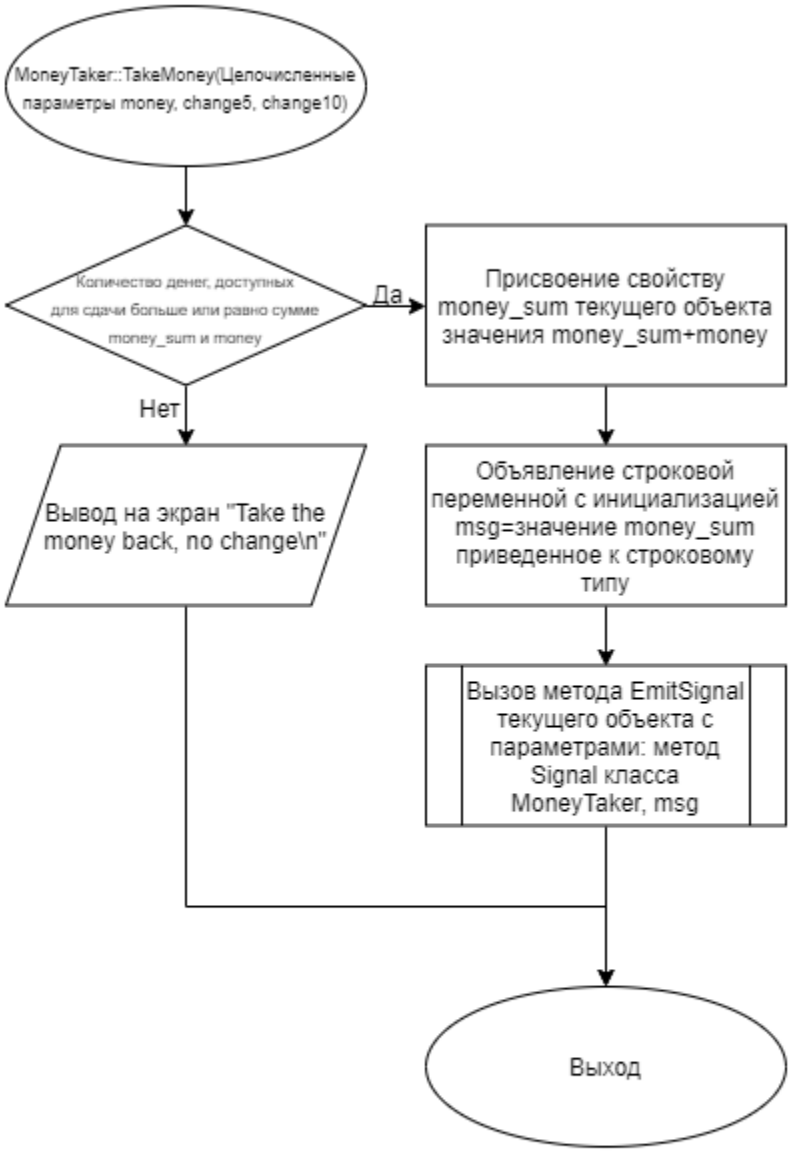

Рисунок 10 – Блок-схема алгоритма

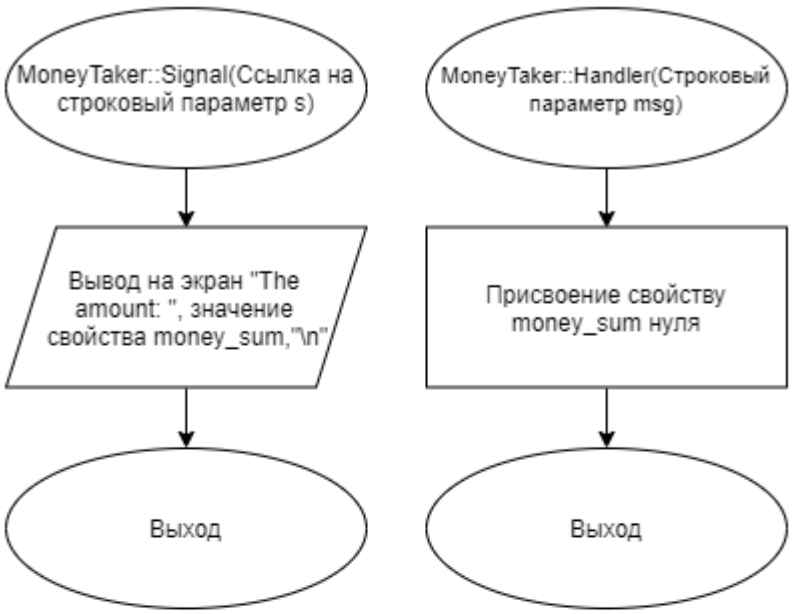

Рисунок 11 – Блок-схема алгоритма

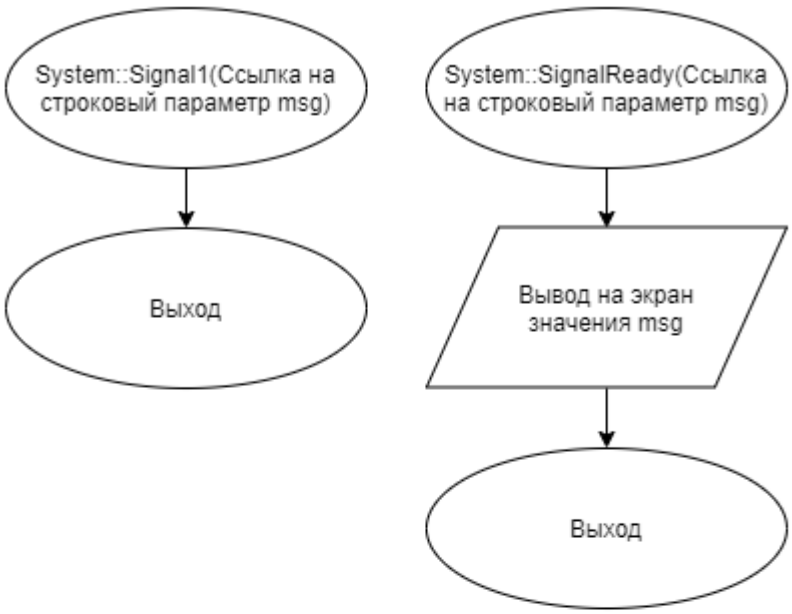

Рисунок 12 – Блок-схема алгоритма

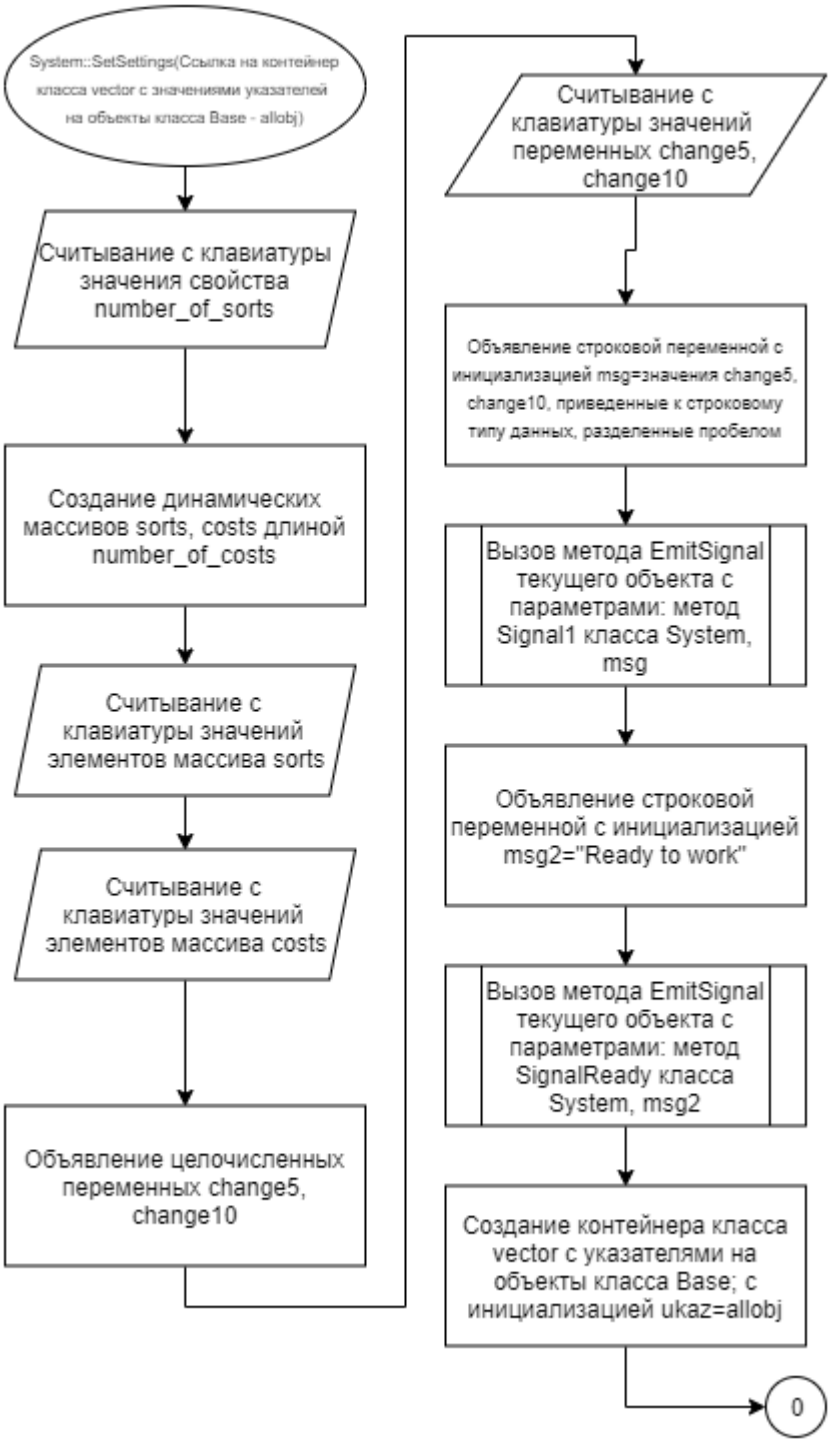

Рисунок 13 – Блок-схема алгоритма

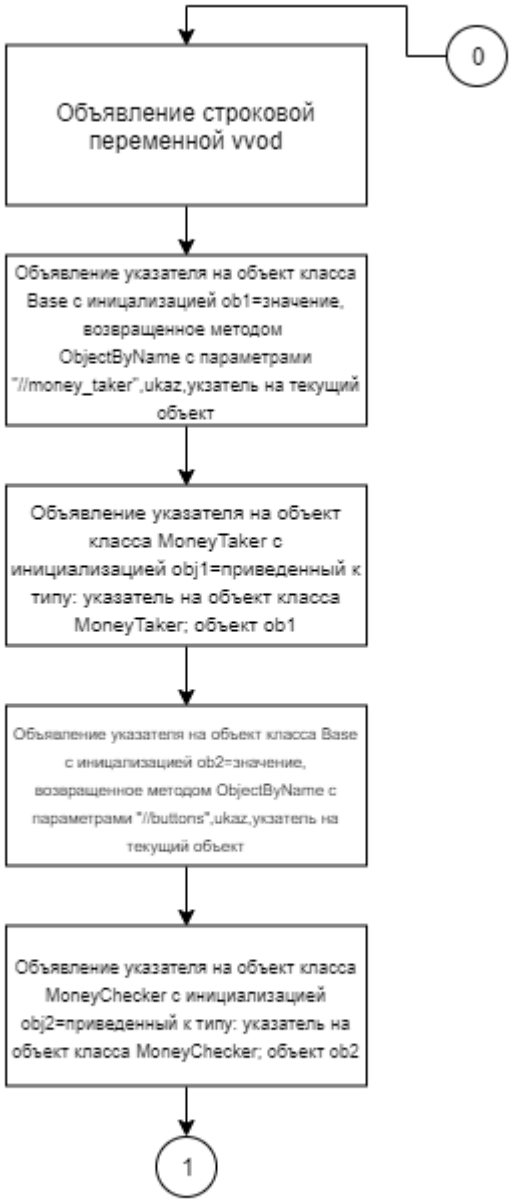

Рисунок 14 – Блок-схема алгоритма

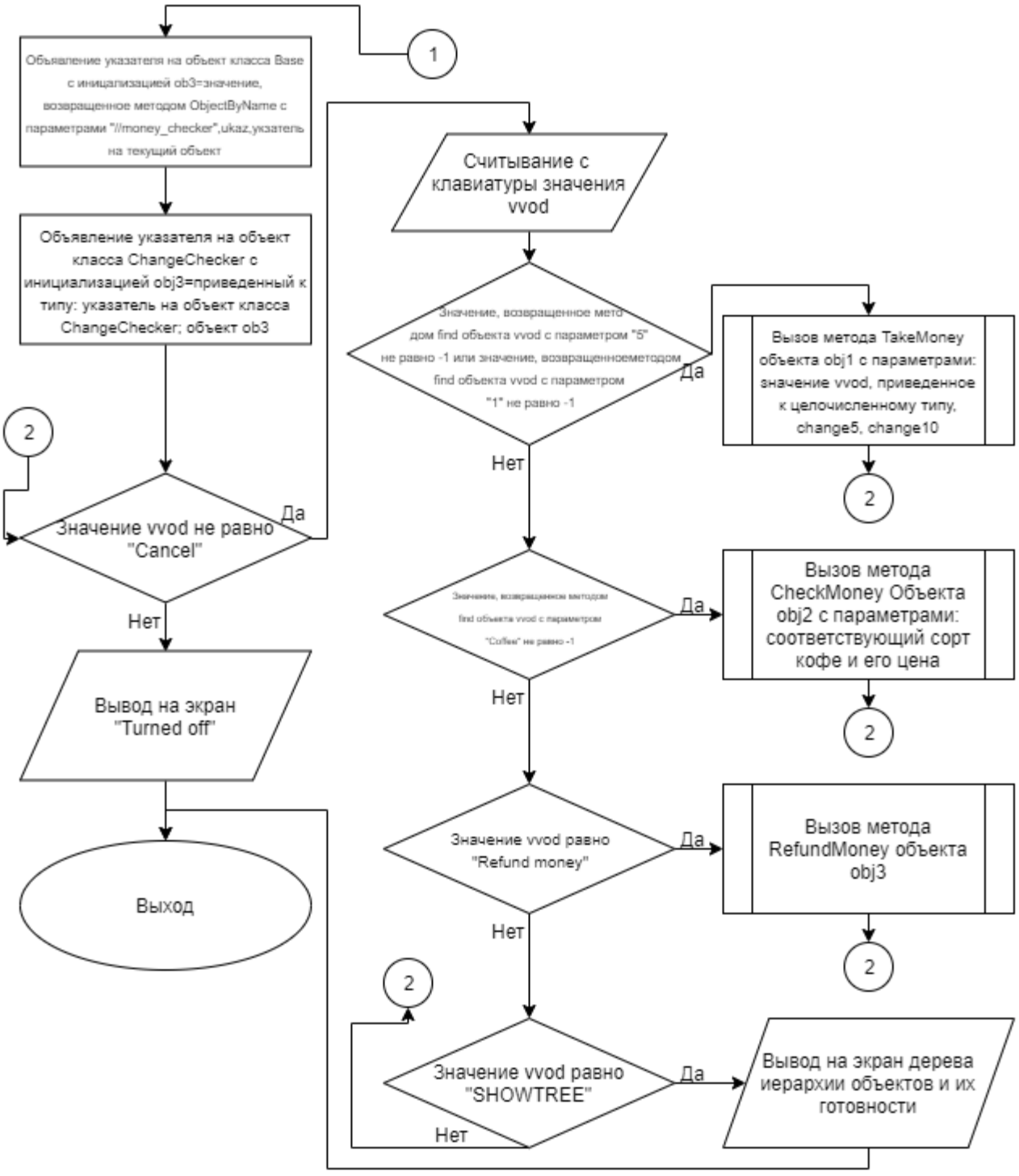

Рисунок 15 – Блок-схема алгоритма

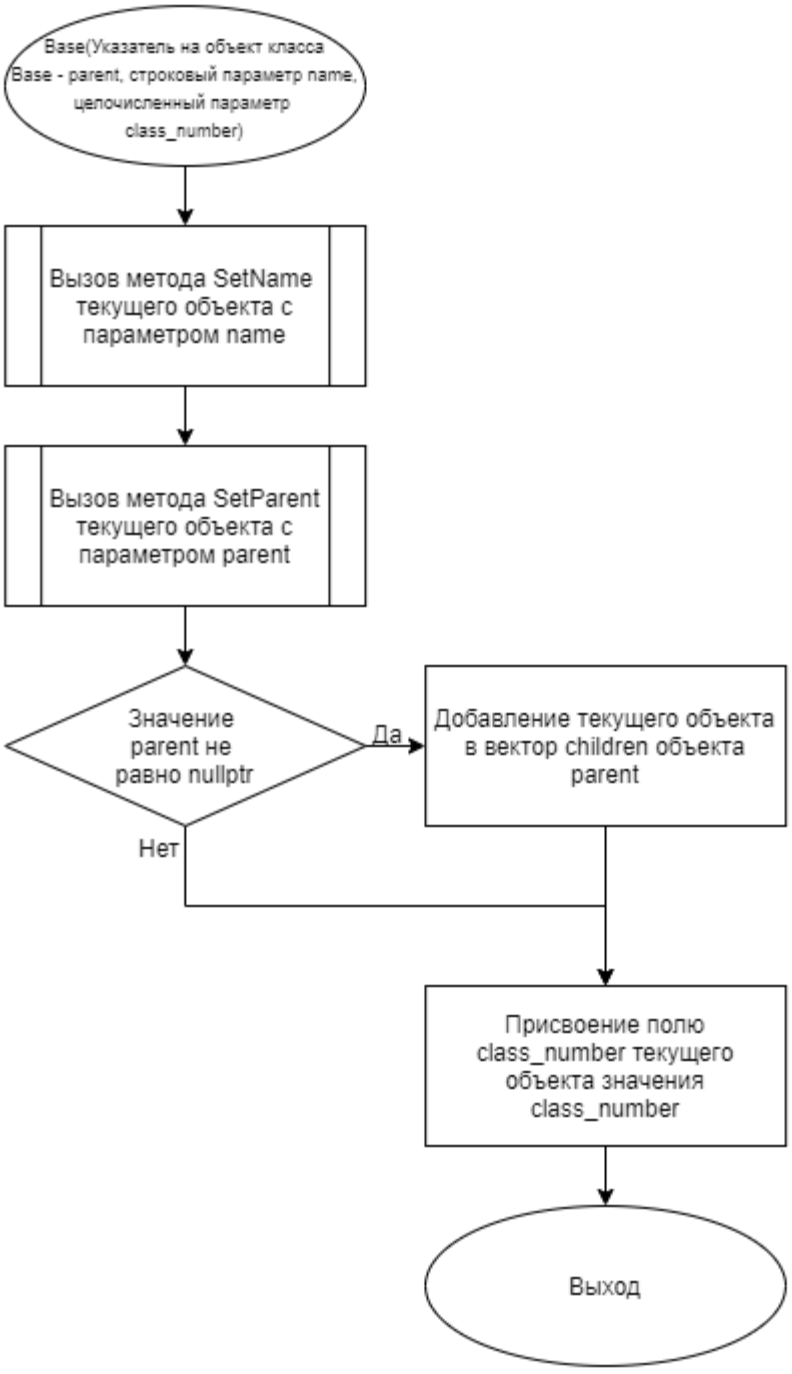

Рисунок 16 – Блок-схема алгоритма

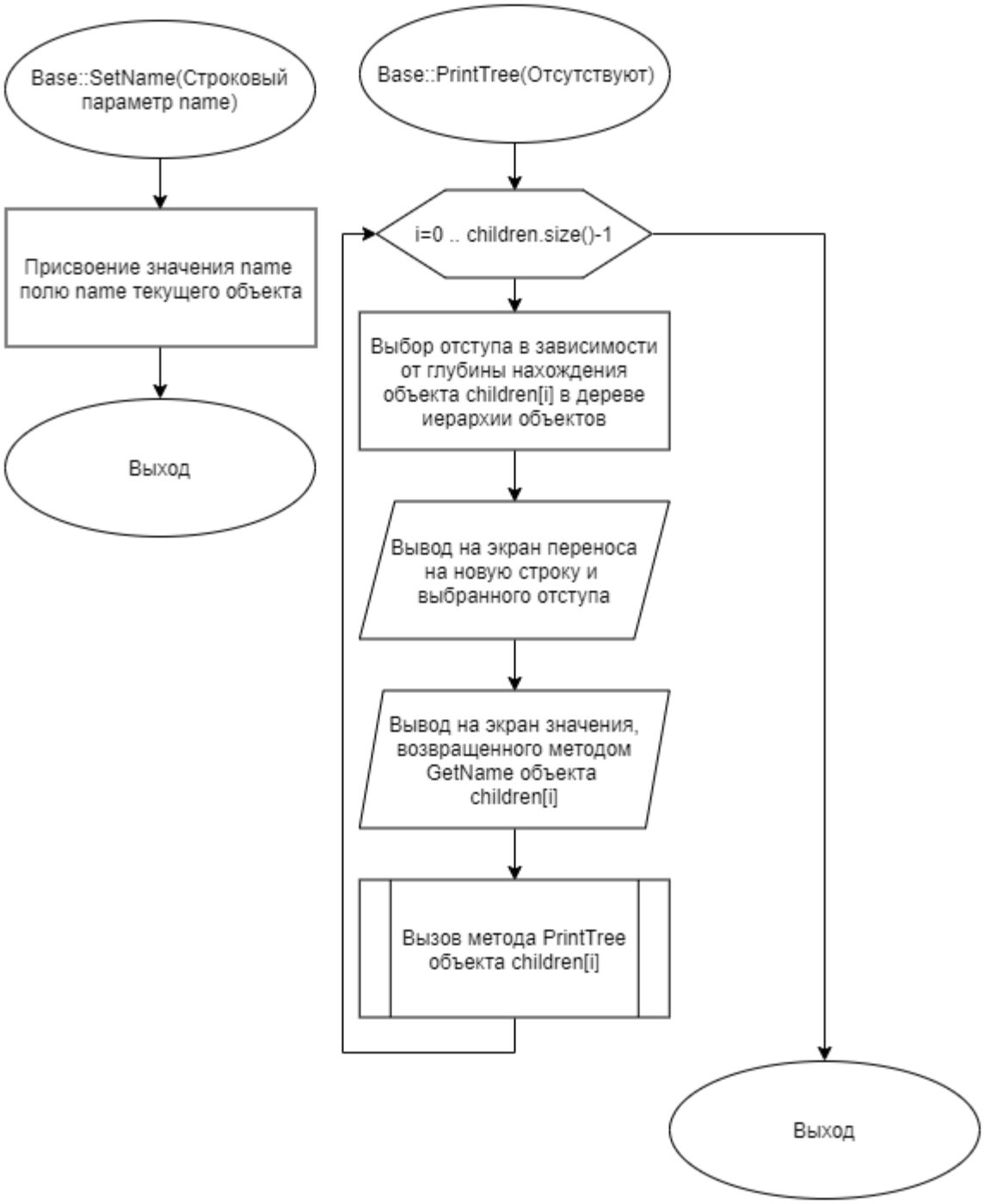

Рисунок 17 – Блок-схема алгоритма

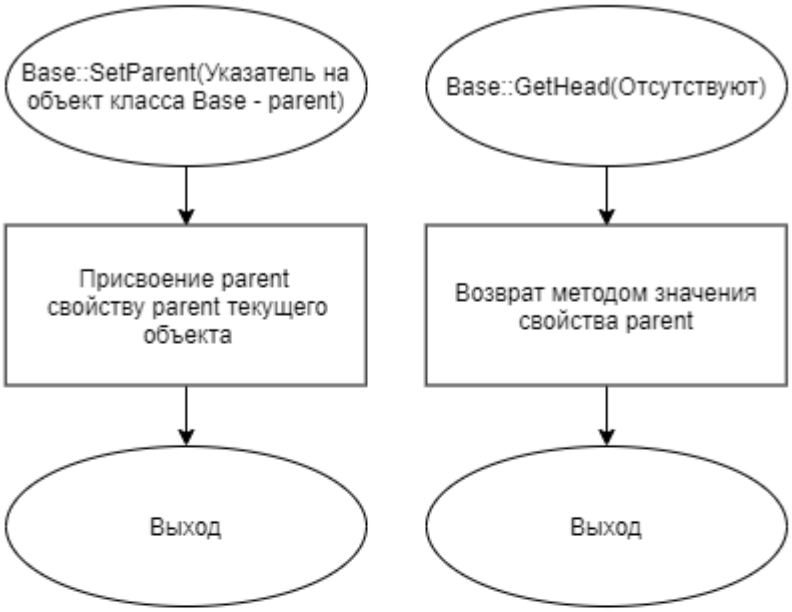

Рисунок 18 – Блок-схема алгоритма

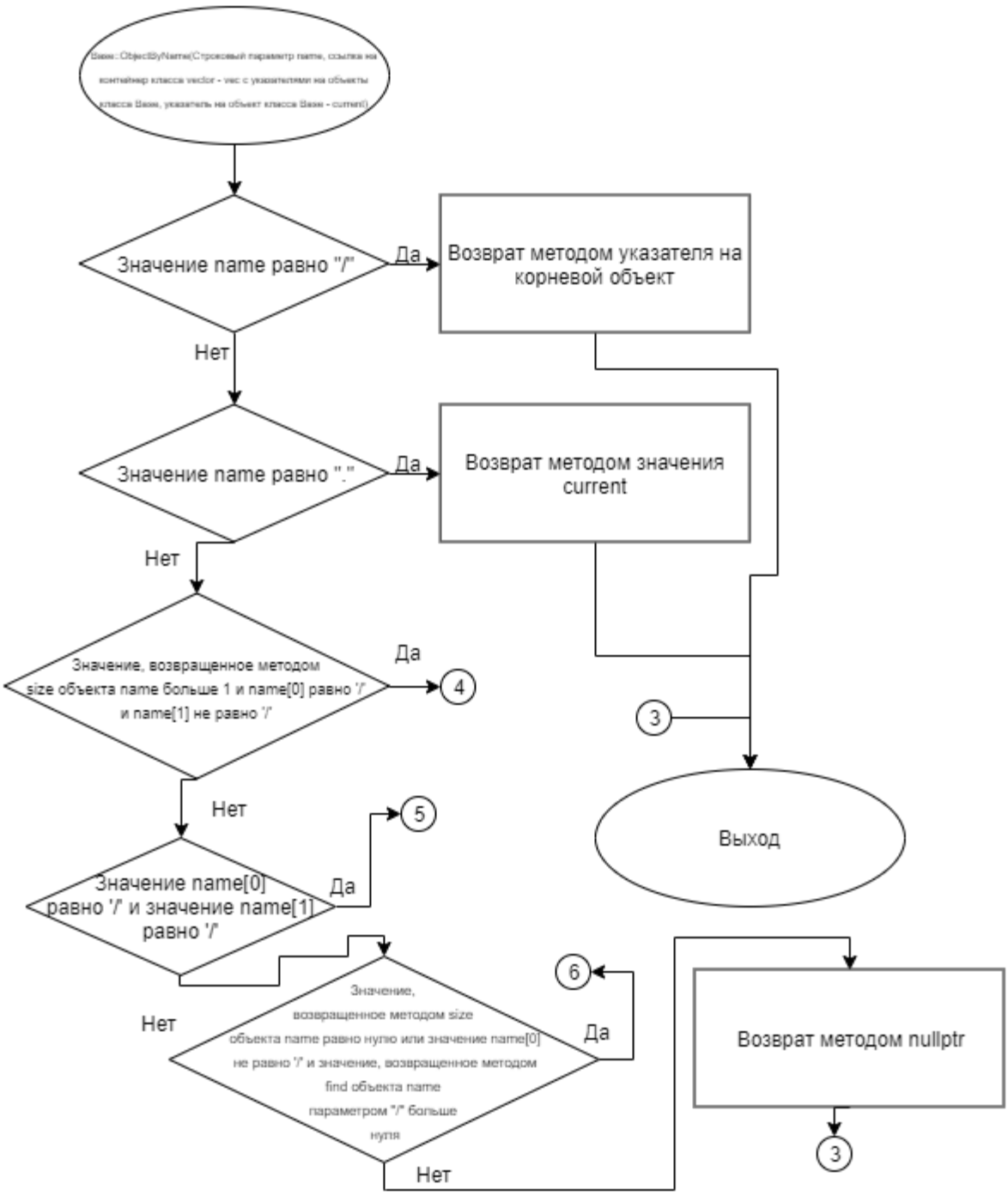

Рисунок 19 – Блок-схема алгоритма

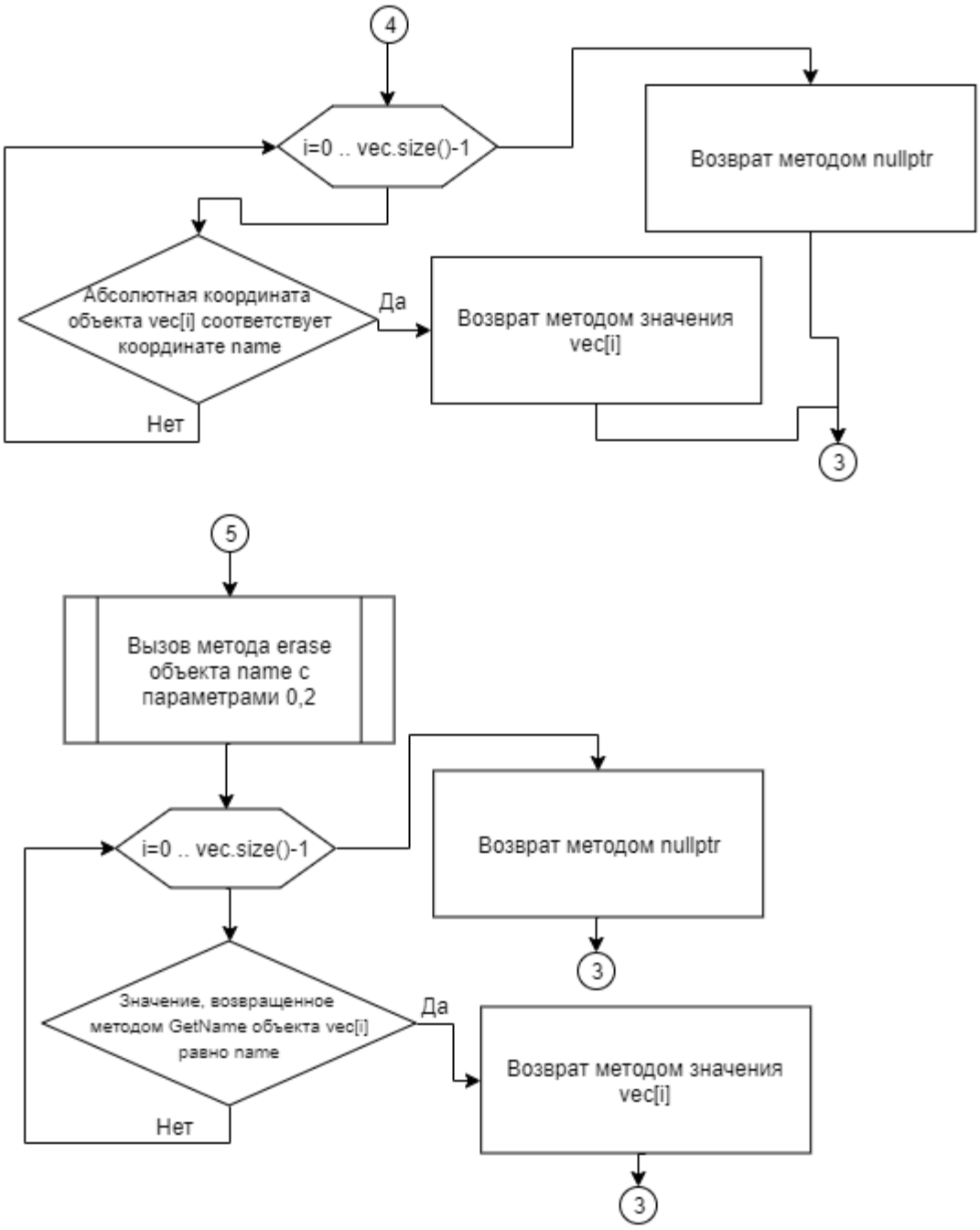

Рисунок 20 – Блок-схема алгоритма

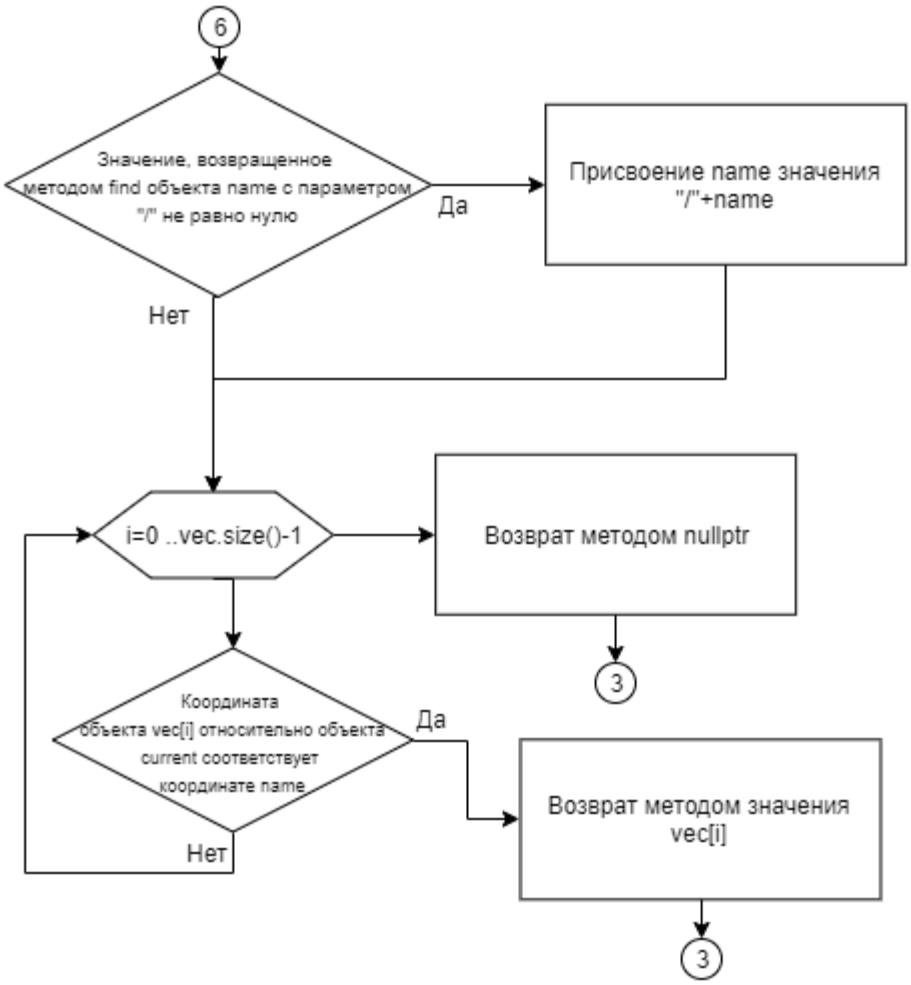

Рисунок 21 – Блок-схема алгоритма

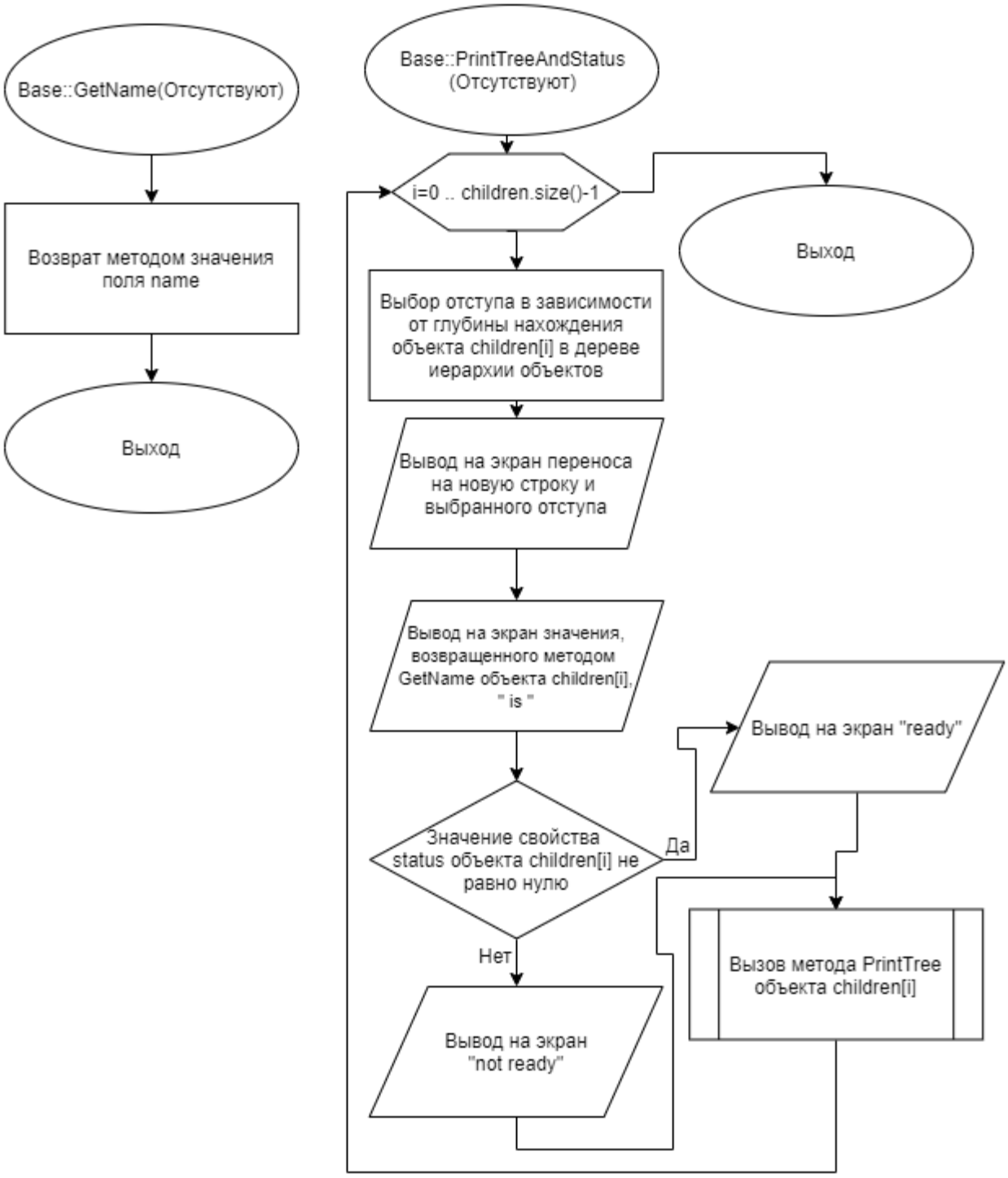

Рисунок 22 – Блок-схема алгоритма

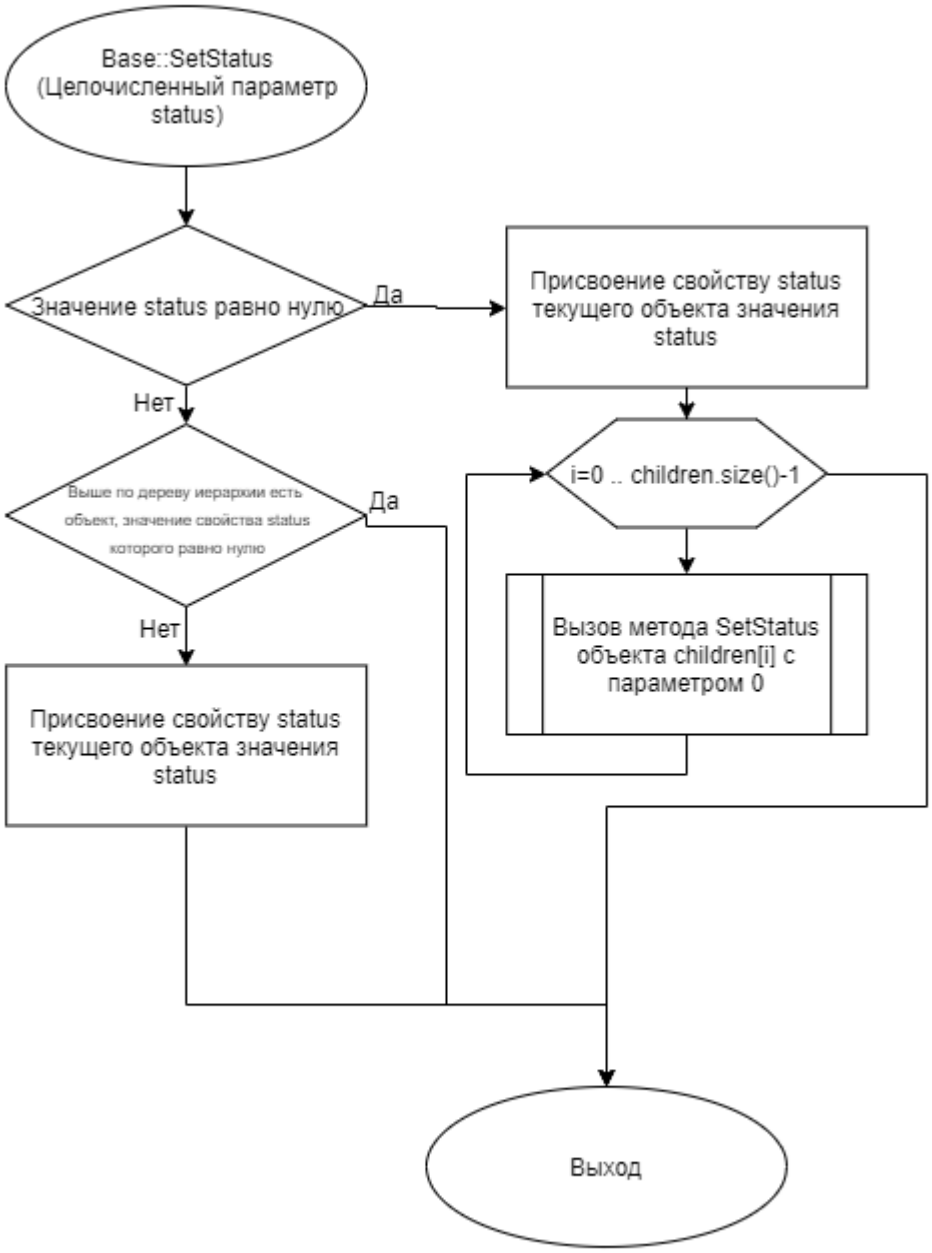

Рисунок 23 – Блок-схема алгоритма

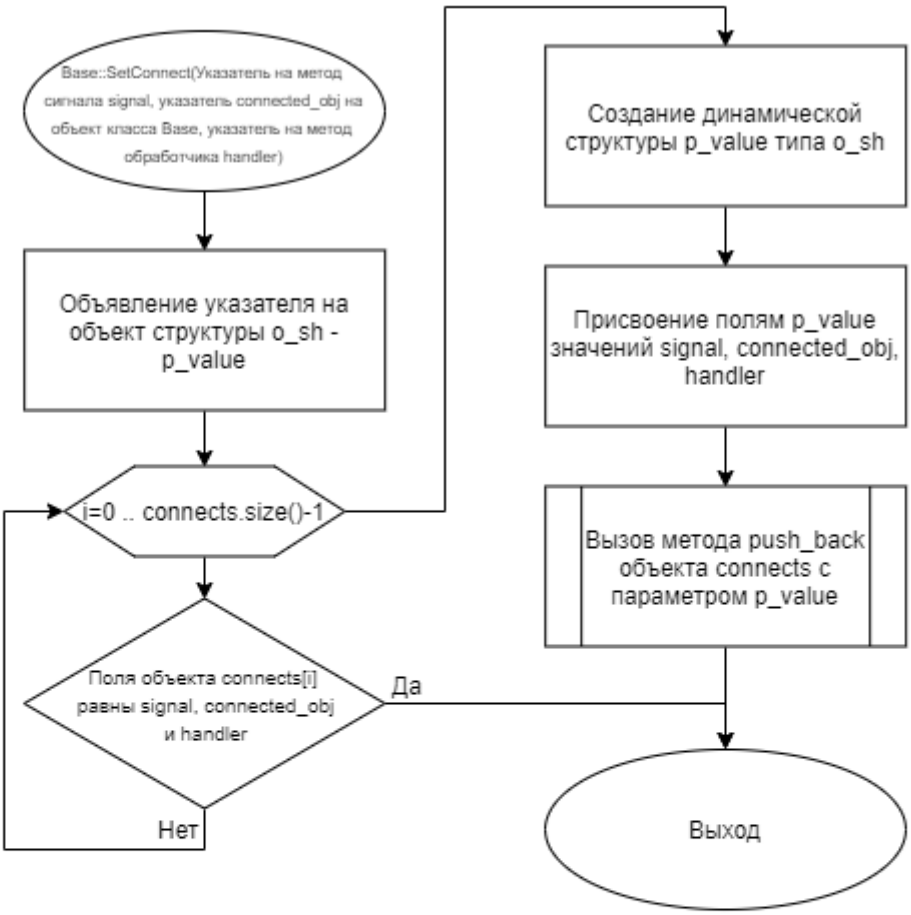

Рисунок 24 – Блок-схема алгоритма

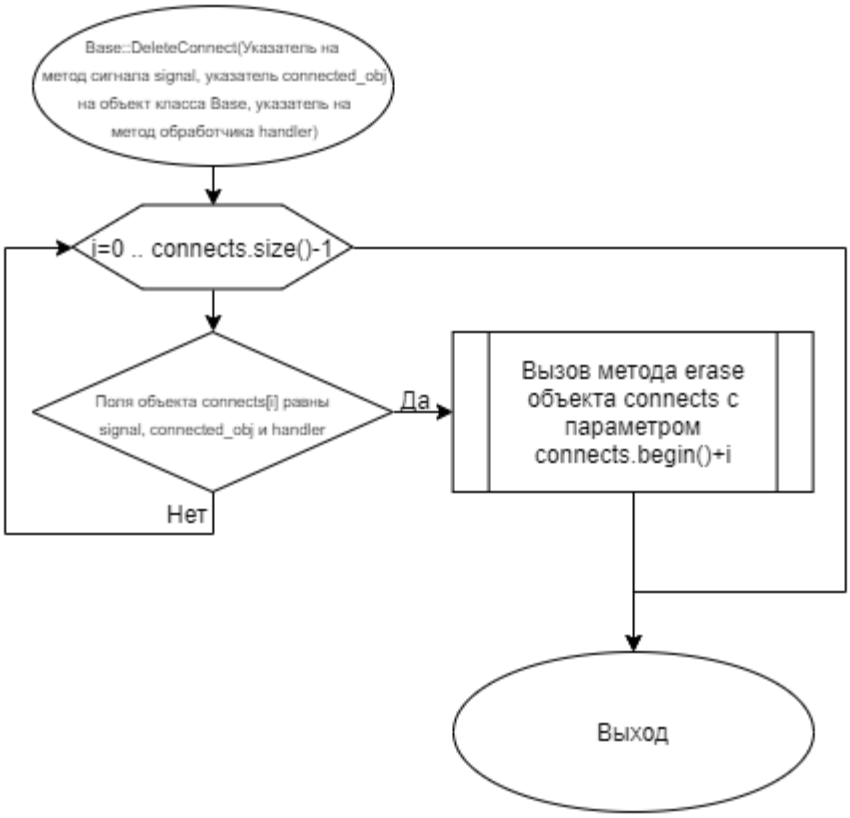

Рисунок 25 – Блок-схема алгоритма

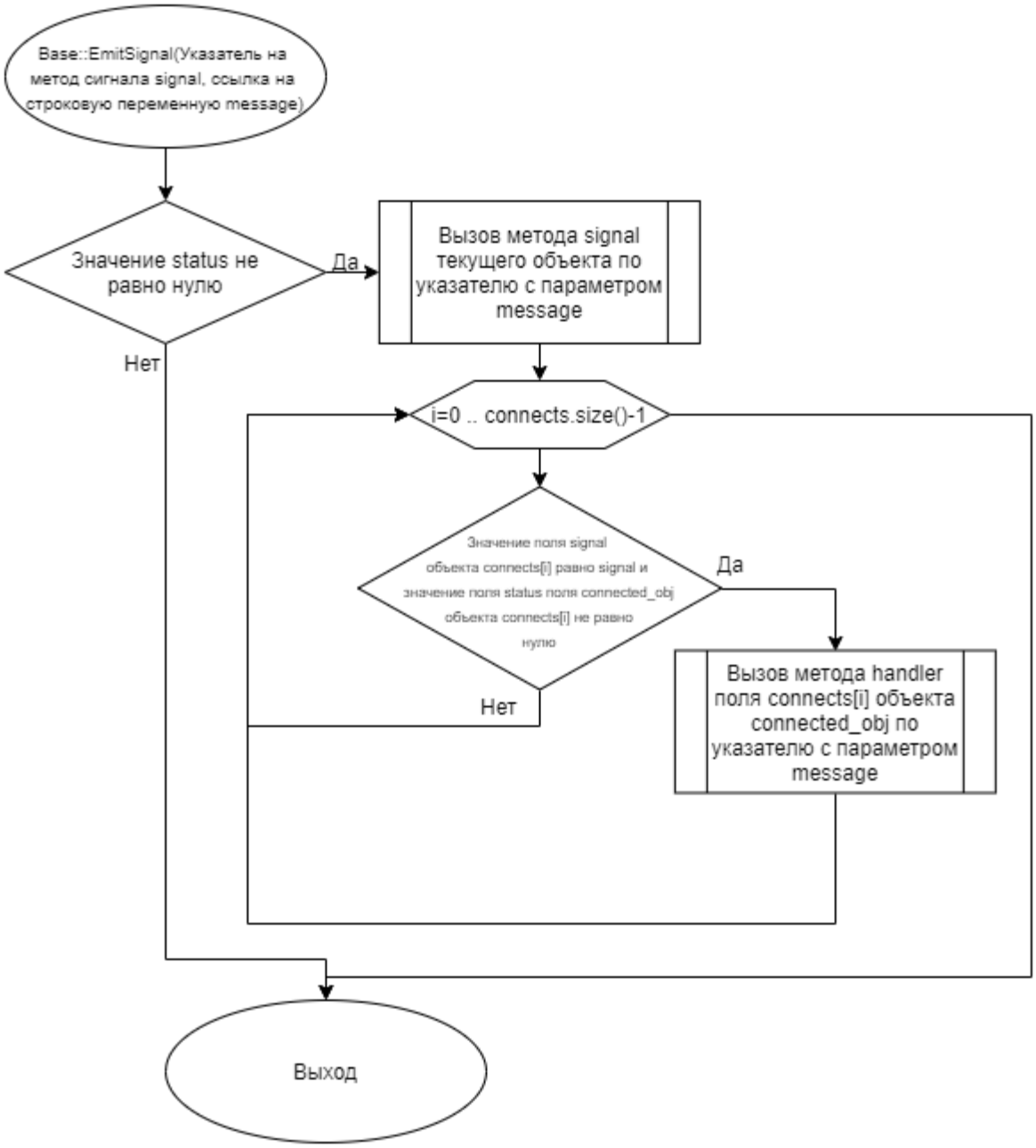

Рисунок 26 – Блок-схема алгоритма

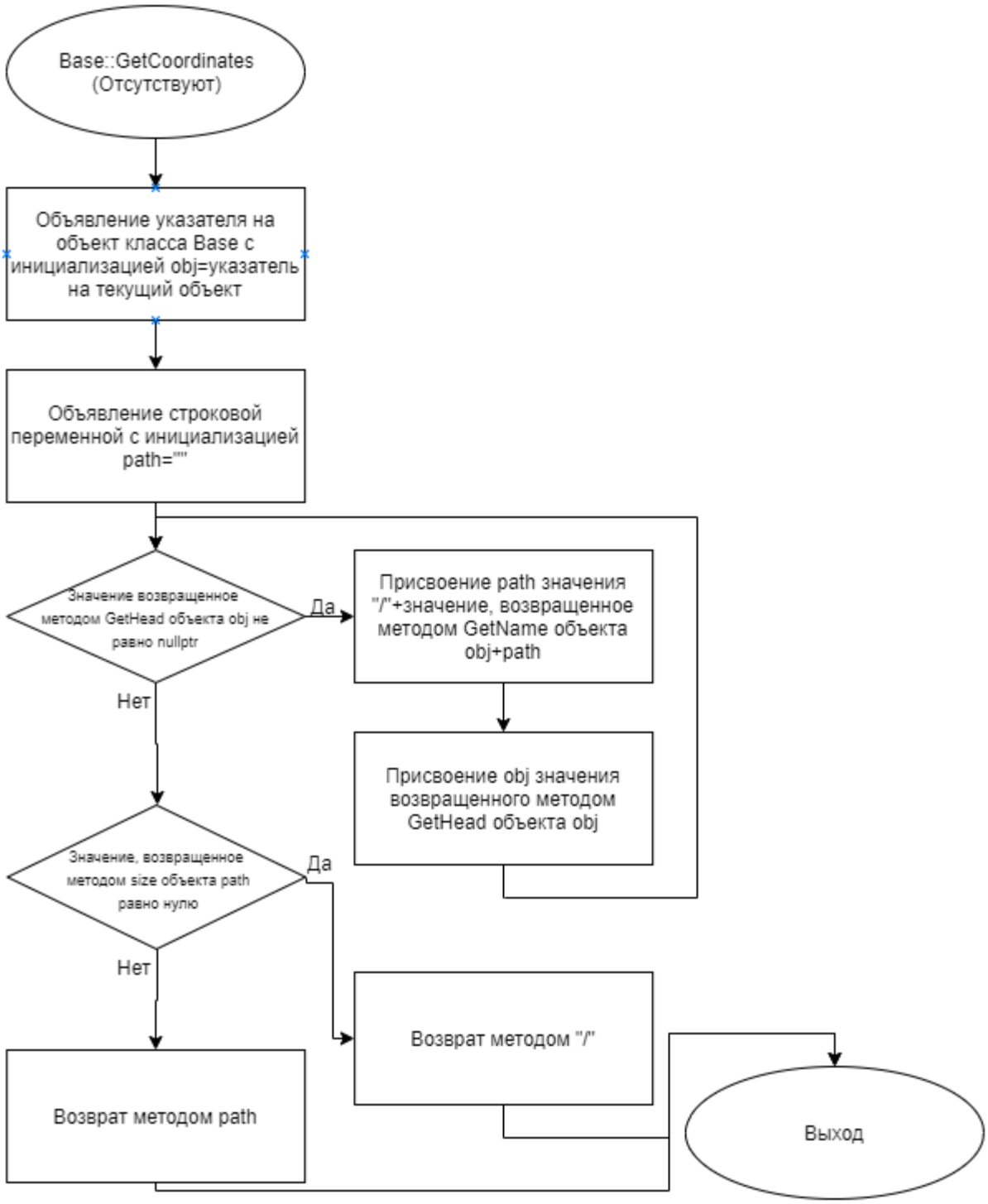

Рисунок 27 – Блок-схема алгоритма

Рисунок 28 – Блок-схема алгоритма

5 КОД ПРОГРАММЫ

Программная реализация алгоритмов для решения задачи представлена ниже.

5.1 Файл Application.cpp

Листинг 1 – Application.cpp

```
#include "Application.h"

#include "MoneyChecker.h"
#include "CoffeeGiver.h"
#include "ChangeChecker.h"
#include "MoneyTaker.h"
#include "System.h"
#include <iostream>
#include <string>
using namespace std;

#define SIGNAL_D(signal_f) (TYPE_SIGNAL) (&signal_f)
#define HENDLER_D(hendler_f) (TYPE_HANDLER) (&hendler_f)
typedef void (Base :: * TYPE_SIGNAL) (string&);
typedef void (Base :: * TYPE_HANDLER) (string);


void Application::BuildTree() {

           this->SetName("app"); ukaz.push_back(this);
           System* sys=new System(this,"system",2); ukaz.push_back(sys);
           System*       settings=new       System(sys,"settings_and_commands",2);
ukaz.push_back(settings);
           MoneyChecker*         checker=new         MoneyChecker(sys,"buttons",3);
ukaz.push_back(checker);
           MoneyTaker*          taker=new          MoneyTaker(sys,"money_taker",4);
ukaz.push_back(taker);
           ChangeChecker*      giver=new      ChangeChecker(sys,"money_checker",5);
ukaz.push_back(giver);
           CoffeeGiver* coffeegiver=new CoffeeGiver(sys,"coffee_giver",6);

           settings-
>SetConnect(SIGNAL_D(System::Signal1),giver,HENDLER_D(ChangeChecker::Handler1));
           taker->SetConnect(SIGNAL_D(MoneyTaker::Signal),                checker,
HENDLER_D(MoneyChecker::Handler));
           checker-
>SetConnect(SIGNAL_D(MoneyChecker::Signal),coffeegiver,HENDLER_D(CoffeeGiver::Hand
ler));
           coffeegiver-
>SetConnect(SIGNAL_D(CoffeeGiver::Signal2),giver,HENDLER_D(ChangeChecker::Handler2
));
```

```
            coffeegiver-
>SetConnect(SIGNAL_D(CoffeeGiver::Signal2),taker,HENDLER_D(MoneyTaker::Handler));
            giver-
>SetConnect(SIGNAL_D(ChangeChecker::Signal1),taker,HENDLER_D(MoneyTaker::Handler))
;
            taker-
>SetConnect(SIGNAL_D(MoneyTaker::Signal),giver,HENDLER_D(ChangeChecker::Handler3))
;

}

int Application::StartApp() {

            for (int i=0;i<ukaz.size();i++)
                        ukaz[i]->SetStatus(1);

            Base* tmp=ObjectByName("//settings_and_commands",ukaz,this);
            System* settings=(System*)tmp;
            settings->SetSettings(ukaz);
            return 0;
}
```

5.2 Файл Application.h

Листинг 2 – Application.h

```
#ifndef APP_H
#define APP_H

#define SIGNAL_D(signal_f) (TYPE_SIGNAL) (&signal_f)
#define HENDLER_D(hendler_f) (TYPE_HANDLER) (&hendler_f)

#include "Base.h"

class Application : public Base {
            using Base::Base;




public:
            vector <Base*> ukaz;

            void BuildTree(); // доработан
            int StartApp(); // доработан

};

#endif
```

5.3 Файл Base.cpp

Листинг 3 – Base.cpp

```
#include "Base.h"
#include <iostream>
using namespace std;

#define SIGNAL_D(signal_f) (TYPE_SIGNAL) (&signal_f)
#define HENDLER_D(hendler_f) (TYPE_HANDLER) (&hendler_f)

typedef void (Base :: * TYPE_SIGNAL) (string&);
typedef void (Base :: * TYPE_HANDLER) (string);



Base::Base(Base* parent, string name, int class_number) {
	SetName(name);
	SetParent(parent);
	if (parent) { // !=nullptr
		parent->children.push_back(this);
	}
	this->class_number=class_number;
}

void Base::SetName(string name) {
	this->name = name; // наименование объекта
}

string Base::GetName() {
	return name; // возврат имени объекта
}

void Base::PrintTree() {


	for (int i = 0; i < children.size(); i++) {
		int flag=0;
		Base* ob=this;
		string s="    ";

		while (ob->GetHead()!=nullptr){
			flag++;
			ob=ob->GetHead();
		}

		for (int j=0;j<flag;j++)
			s+="    ";

		cout<<"\n"<<s;

		cout<<children[i]->GetName();

		children[i]->PrintTree();
	}
```

```
	}


Base* Base::GetHead() {
	return	parent; // возврат указателя на головной объект
}

void Base::SetParent(Base* parent) {
	this->parent = parent;
}

Base*  Base::ObjectByName(string  name,  vector  <Base*>  &vec,  Base*  current){
//получение объекта по имении в дереве иерархи
	if (name=="/"){
		Base* ob=this;
		while (ob->GetHead()!=nullptr){
			ob=ob->GetHead();
		}
		return ob;
	}
	else
	if (name=="."){
		return current;
	}
	else
	if (name.size()>1 && name[0]=='/' && name[1]!='/'){
		for (int i=0;i<vec.size();i++){
			string copy=name;
			Base* object;
			int last_slash_pos=copy.rfind("/");

			string ob="";
			for (int k=last_slash_pos+1;k<copy.size();k++)
				ob+=copy[k];

			if (vec[i]->GetName()==ob){
				object=vec[i];

				while (object->GetHead() != nullptr){

	last_slash_pos=copy.rfind("/");
							string ob="", ob2="";
							if	(copy.rfind("/")==-1)
return nullptr;
							for			(int
f=last_slash_pos+1;f<copy.size();f++)
								ob+=copy[f];

							copy.erase(last_slash_pos,
ob.size()+1);


	last_slash_pos=copy.rfind("/");
							if (last_slash_pos!=-1)
							for			(int
p=last_slash_pos+1;p<copy.size();p++)
								ob2+=copy[p];
```

```
                                                  if (ob2!=""){
                                                            for             (int
j=0;j<vec.size();j++){
                                                                                  if
(vec[j]->GetName()==ob2) break;
                                                                                  if
(j==vec.size()-1 && vec[j]->GetName()!=ob2) return nullptr;
                                                            }
                                                  }
                                                  if     (object->GetName()==ob)
object=object->GetHead();
                                                  else break;
                                        }
                                        if  (object->GetHead()==nullptr)  return
vec[i];
                              }
                    }
                    return nullptr;
          }
          if (name[0]=='/' && name[1]=='/'){
                    name.erase(0,2);
                    for (int i=0;i<vec.size();i++)
                              if (vec[i]->GetName()==name) return vec[i];
                    return nullptr;
          }
          else
          if (name.size()==0 || (name[0]!='/' && name.find("/")>0)){
                    if (name.find("/")!=0) name="/"+name;

                    for (int i=0;i<vec.size();i++){
                              string copy=name;
                              Base* object;
                              int last_slash_pos=copy.rfind("/");

                              string ob="";
                              for (int k=last_slash_pos+1;k<copy.size();k++)
                                        ob+=copy[k];

                              if (vec[i]->GetName()==ob){
                                        object=vec[i];

                                        while (object->GetHead() != current){

          last_slash_pos=copy.rfind("/");
                                                  string ob="", ob2="";
                                                  if      (copy.rfind("/")==-1)
return nullptr;
                                                  for                       (int
f=last_slash_pos+1;f<copy.size();f++)
                                                            ob+=copy[f];

                                                  copy.erase(last_slash_pos,
ob.size()+1);


          last_slash_pos=copy.rfind("/");
```

```
                                                            if (last_slash_pos!=-1)
                                                            for                         (int
p=last_slash_pos+1;p<copy.size();p++)
                                                                        ob2+=copy[p];

                                                            if (ob2!=""){
                                                                        for             (int
j=0;j<vec.size();j++){
                                                                                    if
(vec[j]->GetName()==ob2) break;
                                                                                    if
(j==vec.size()-1 && vec[j]->GetName()!=ob2) return nullptr;
                                                                        }
                                                            }
                                                            if     (object->GetName()==ob)
object=object->GetHead();
                                                            else break;
                                                }
                                                if  (object->GetHead()==current)  return
vec[i];
                                    }
                        }
                        return nullptr;
            }

            else return nullptr;
}
void Base::SetStatus(int status){
            if (status==0){
                        this->status=status;
                        for (int i = 0; i < children.size(); i++)
                                    children[i]->SetStatus(0);
            }
            else{
                        Base* ob=this;
                        while (ob->GetHead()!=nullptr){
                                    ob=ob->GetHead();
                                    if (ob->status==0) return;
                        }
                        this->status=status;
            }
}

void Base::PrintTreeAndStatus(){
            for (int i = 0; i < children.size(); i++) {
                        int flag=0;
                        Base* ob=this;
                        string s="    ";

                        while (ob->GetHead()!=nullptr){
                                    flag++;
                                    ob=ob->GetHead();
                        }

                        for (int j=0;j<flag;j++)
                                    s+="    ";
```

```
                    cout<<"\n"<<s;

                    cout<<children[i]->GetName()<<" is ";
                    if (children[i]->status!=0) cout<<"ready";
                    else cout<<"not ready";

                    children[i]->PrintTreeAndStatus();
          }
}

void Base :: SetConnect(TYPE_SIGNAL signal, Base* connected_obj, TYPE_HANDLER
handler){
          o_sh* p_value;

          for (int i=0;i<connects.size();i++){
                    if   (connects[i]->signal   ==   signal   &&   connects[i]-
>connected_obj == connected_obj &&
                              connects[i]->handler == handler) return;
          }

          p_value=new o_sh();

          p_value->signal=signal;
          p_value->connected_obj=connected_obj;
          p_value->handler=handler;

          connects.push_back(p_value);
}

void Base :: DeleteConnect(TYPE_SIGNAL signal, Base* connected_obj, TYPE_HANDLER
handler){
          for (int i=0;i<connects.size();i++)
                    if   (connects[i]->signal   ==   signal   &&   connects[i]-
>connected_obj == connected_obj &&
                              connects[i]->handler          ==          handler)
{connects.erase(connects.begin()+i); return;}
}

void Base :: EmitSignal(TYPE_SIGNAL signal, string& message){
          if (status!=0){
                    (this->*(signal))(message);
                    for (int i=0;i<connects.size();i++){
                              if  (connects[i]->signal==signal  &&  connects[i]-
>connected_obj->status!=0)
                                        (connects[i]->connected_obj-
>*(connects[i]->handler))(message);
                    }
          }
}

string Base::GetCoordinates(){
          Base* obj=this;
          string path="";

          while (obj->GetHead()!=nullptr){
                    path="/"+obj->GetName()+path;
                    obj=obj->GetHead();
```

```
            }
            if (path.size()==0) return "/";
            else return path;
}

int Base::GetClassNumber(){
            return class_number;
}
```

5.4 Файл Base.h

Листинг 4 – Base.h

```
#ifndef BASE_H
#define BASE_H

#define SIGNAL_D(signal_f) (TYPE_SIGNAL) (&signal_f)
#define HENDLER_D(hendler_f) (TYPE_HANDLER) (&hendler_f)
#include <iostream>
#include <string>
#include <vector>
using namespace std;

class Base {
            protected:
                        string name;
                        int status=1;
                        vector <Base*> children;
            private:
                        Base* parent;
                        int class_number; //новое

            public:

                        typedef void (Base :: * TYPE_SIGNAL) (string&);
                        typedef void (Base :: * TYPE_HANDLER) (string);

                        Base(Base* parent, string name, int class_number);

                        void SetName(string name);
                        void PrintTree();
                        void SetParent(Base* parent);

                        Base* GetHead();
                        Base*  ObjectByName(string  name,  vector  <Base*>  &vec,  Base*
current);

                        string GetName();

                        void PrintTreeAndStatus();
                        void SetStatus(int status);

//////////////////////////////////////////////
```

```
                    struct o_sh{ //новое
                              TYPE_SIGNAL signal;
                              Base* connected_obj;
                              TYPE_HANDLER handler;
                    };

                    vector <o_sh*> connects; //новое
                    void   SetConnect(TYPE_SIGNAL   p_signal,   Base*   p_object,
TYPE_HANDLER p_ob_hendler); //новое
                    void  DeleteConnect(TYPE_SIGNAL  p_signal,  Base*  p_object,
TYPE_HANDLER p_ob_hendler); //новое
                    void  EmitSignal(TYPE_SIGNAL  p_signal,  string&  s_command);
//новое

                    string GetCoordinates(); //новое

                    int GetClassNumber(); //новое

};

#endif
```

5.5 Файл ChangeChecker.cpp

Листинг 5 – ChangeChecker.cpp

```
#include "ChangeChecker.h"

void ChangeChecker::RefundMoney(){
          string msg="";

          this->EmitSignal(SIGNAL_D(ChangeChecker::Signal1),msg);
}

void ChangeChecker::Handler1(string change_sum){
          int space_pos=change_sum.find(" ");
          change5=atoi(change_sum.substr(0,space_pos).c_str());
          change10=atoi(change_sum.substr(space_pos+1).c_str());
}

void ChangeChecker::Handler2(string decr_change){
          int space_pos=decr_change.find(" ");

          change5-=atoi(decr_change.substr(0,space_pos).c_str());
          change10-=atoi(decr_change.substr(space_pos+1).c_str());
}

void ChangeChecker::Handler3(string msg){
          this->money_sum=atoi(msg.c_str());
}

void ChangeChecker::Signal1(string& msg){
```

```
            int num5=0,num10=0;
            while (money_sum>=10){
                        num10++;
                        money_sum-=10;
            }
            while (money_sum>0){
                        num5++;
                        money_sum-=5;
            }
            if (num5+num10>0) {cout<<"Take the money: 10 * "<<num10<<" rub.,   5 *
"<<num5<<" rub.\n";
                                                                        cout<<"Ready     to
work\n";}

            msg=to_string(num5)+" "+to_string(num10);
}
```

5.6 Файл ChangeChecker.h

Листинг 6 – ChangeChecker.h

```
#ifndef GIVEMONEY_H
#define GIVEMONEY_H

#include "Application.h"
#include "System.h"
class ChangeChecker : public Base {
            using Base::Base;
            private:
                        int money_sum=0;
                        int change5;
                        int change10;

            public:
                        void RefundMoney();
                        void Signal1(string& s);
                        void Handler1(string s); //новое
                        void Handler2(string s);
                        void Handler3(string msg);

};

#endif
```

5.7 Файл CoffeeGiver.cpp

Листинг 7 – CoffeeGiver.cpp

```
#include "CoffeeGiver.h"
```

```
void CoffeeGiver::GiveCoffee(){
          this->EmitSignal(SIGNAL_D(CoffeeGiver::Signal1),sort);
          this->GiveChange();
          if (change==0) cout<<"Ready to work\n";
}

void CoffeeGiver::GiveChange(){
          string msg=to_string(change);
          this->EmitSignal(SIGNAL_D(CoffeeGiver::Signal2),msg);
}


void CoffeeGiver::Handler(string msg){
          int space_pos=msg.find(" ");
          change=atoi(msg.substr(0,space_pos).c_str());
          sort=msg.substr(space_pos+1);
          this->GiveCoffee();

}

void CoffeeGiver::Signal1(string& msg){
          cout<<"Take the coffee "<<msg<<"\n";
}

void CoffeeGiver::Signal2(string& msg){
          int num5=0,num10=0;
          int chng=atoi(msg.c_str());
          while (chng>=10){
                    num10++;
                    chng-=10;
          }
          while (chng>0){
                    num5++;
                    chng-=5;
          }
          if (num5+num10>0) {cout<<"Take the change: 10 * "<<num10<<" rub.,  5 *
"<<num5<<" rub.\n";
                                                                    cout<<"Ready    to
work\n";}

          msg=to_string(num5)+" "+to_string(num10);
}
```

5.8 Файл CoffeeGiver.h

Листинг 8 – CoffeeGiver.h

```
#ifndef GIVECOFFEE_H
#define GIVECOFFEE_H

#include "Application.h"
#include "System.h"
```

```
class CoffeeGiver : public Base {
          using Base::Base;
          private:
                    int change=0;
                    string sort="";
          public:
                    void GiveCoffee();
                    void GiveChange();
                    void Signal1(string& s); //новое
                    void Signal2(string& s);
                    void Handler(string s); //новое

};

#endif
```

5.9 Файл main.cpp

Листинг 9 – main.cpp

```
#include "Base.h"
#include "Application.h"

#include <iostream>
#include <string>
using namespace std;

int main()
{
          Application app(nullptr, "", 1);
          app.BuildTree();
          return app.StartApp();
}
```

5.10 Файл MoneyChecker.cpp

Листинг 10 – MoneyChecker.cpp

```
#include "MoneyChecker.h"

void MoneyChecker::CheckMoney(string sort,int cost){
          if (cost<=money_sum) {string msg=to_string(money_sum-cost)+" "+sort;
                                                                        this-
>EmitSignal(SIGNAL_D(MoneyChecker::Signal),msg);}
          else cout<<"There is not enough money\n";
}

void MoneyChecker::Handler(string money_sum){
          this->money_sum=atoi(money_sum.c_str());
```

```
}

void MoneyChecker::Signal(string& msg){
}
```

5.11 Файл MoneyChecker.h

Листинг 11 – MoneyChecker.h

```
#ifndef CHECKMONEY_H
#define CHECKMONEY_H

#include "Application.h"
#include "System.h"

class MoneyChecker : public Base {
           using Base::Base;
           private:
                      int money_sum=0;

           public:
                      void CheckMoney(string sort,int cost);
                      void Signal(string& s); //новое
                      void Handler(string s); //новое


};

#endif
```

5.12 Файл MoneyTaker.cpp

Листинг 12 – MoneyTaker.cpp

```
#include "MoneyTaker.h"

void MoneyTaker::TakeMoney(int money,int change5, int change10){
           if ((5*change5+10*change10)>=money_sum+money) this->money_sum+=money;

           else {cout<<"Take the money back, no change\n";return;}
           string msg=to_string(money_sum);
           this->EmitSignal(SIGNAL_D(MoneyTaker::Signal), msg);
}

void MoneyTaker::Signal(string& s){
           cout<<"The amount: "<<money_sum<<"\n";
}
void MoneyTaker::Handler(string msg){
           money_sum=0; }
```

5.13 Файл MoneyTaker.h

Листинг 13 – MoneyTaker.h

```
#ifndef TAKEMONEY_H
#define TAKEMONEY_H

#include "Application.h"
#include "System.h"
#include "Base.h"
class MoneyTaker :  public Base{
           using Base::Base;
           private:
                      int money_sum=0;

           public:
                      void TakeMoney(int money,int,int);
                      void Signal(string& s); //новое
                      void Handler(string s); //новое



};

#endif
```

5.14 Файл System.cpp

Листинг 14 – System.cpp

```
#include "System.h"
#include <iostream>
#include <string>
#include <vector>
using namespace std;

void System::SetSettings(vector <Base*>& allobj){
           cin>>number_of_sorts;
           this->sorts=new string[number_of_sorts];
           this->costs=new int[number_of_sorts];


           for (int i=0;i<number_of_sorts;i++){
                      cin>>sorts[i];
           }

           for (int i=0;i<number_of_sorts;i++){
                      cin>>costs[i];
```

```
            }

            int change5, change10;
            cin>>change5>>change10;
            string msg=to_string(change5)+" "+to_string(change10);
            this->EmitSignal(SIGNAL_D(System::Signal1),msg);
            string msg2="Ready to work\n";
            this->EmitSignal(SIGNAL_D(System::SignalReady),msg2);

            vector <Base*> ukaz=allobj;

            string vvod;

            Base* ob1=ObjectByName("//money_taker",ukaz,this);
            MoneyTaker* obj1=(MoneyTaker*)ob1;

            Base* ob2=ObjectByName("//buttons",ukaz,this);
            MoneyChecker* obj2=(MoneyChecker*)ob2;

            Base* ob3=ObjectByName("//money_checker",ukaz,this);
            ChangeChecker* obj3=(ChangeChecker*)ob3;

            while (vvod!="Cancel"){
                        getline(cin,vvod);
                        //cin.ignore(32767,'\n');
                        if (vvod.find("5")!=-1 || vvod.find("1")!=-1){ //ввод денег
                                    obj1-
>TakeMoney(atoi(vvod.c_str()),change5,change10);
                        }
                        else
                        if (vvod.find("Coffee")!=-1){//выбор кофе
                                    string coffee=vvod.substr(vvod.find(" ")+1);
                                    for (int i=0;i<number_of_sorts;i++)
                                                if        (sorts[i]==coffee)        obj2-
>CheckMoney(sorts[i],costs[i]);
                        }
                        else
                        if (vvod=="Refund money"){ // возврат сдачи
                                    obj3->RefundMoney();
                        }
                        else
                        if (vvod=="SHOWTREE"){ // иерархия
                                    cout<<ObjectByName("/",allobj,this)->GetName();
                                    ObjectByName("/",allobj,this)-
>PrintTreeAndStatus();
                                    return;
                        }
            }
            cout<<"Turned off";
}

void System::Signal1(string& msg){
}

void System::SignalReady(string& msg){
            cout<<msg;
}
```

```
//void System::Signal3(string& msg){
//         cout<<msg;
//}
```

5.15 Файл System.h

Листинг 15 – System.h

```
#ifndef SYSTEM_H
#define SYSTEM_H

#include "Application.h"
#include "MoneyTaker.h"
#include "MoneyChecker.h"
#include "ChangeChecker.h"
class System : public Base{
           using Base::Base;
           protected:
                      int number_of_sorts=0;
                      string* sorts;
                      int* costs;

           public:
                      void SetSettings(vector <Base*>& allobj);

                      void Signal1(string& msg);
                      void SignalReady(string& msg);

};

#endif
```

6 ТЕСТИРОВАНИЕ

Результат тестирования программы представлен в таблице 38.

Таблица 38 – Результат тестирования программы

Входные данные	Ожидаемые выходные данные	Фактические выходные данные
3 Espresso Americano Cappuchino 25 50 50 3 5 50 Coffee Cappuchino 10 10 10 Coffee Espresso 5 5 Refund money 5 100 Cancel	Ready to work The amount: 50 Take the coffee Cappuchino Ready to work The amount: 10 The amount: 20 The amount: 30 Take the coffee Espresso Take the change: 10 * 0 rub., 5 * 1 rub. Ready to work The amount: 5 The amount: 10 Take the money: 10 * 1 rub., 5 * 0 rub. Ready to work The amount: 5 Take the money back, no change Turned off	Ready to work The amount: 50 Take the coffee Cappuchino Ready to work The amount: 10 The amount: 20 The amount: 30 Take the coffee Espresso Take the change: 10 * 0 rub., 5 * 1 rub. Ready to work The amount: 5 The amount: 10 Take the money: 10 * 1 rub., 5 * 0 rub. Ready to work The amount: 5 Take the money back, no change Turned off
3 a b c 80 60 70 5 5 50 Coffee b 10 10 5 Coffee c 5 Refund money 50 Cancel	Ready to work The amount: 50 There is not enough money The amount: 60 The amount: 70 The amount: 75 Take the coffee c Take the change: 10 * 0 rub., 5 * 1 rub. Ready to work The amount: 5 Take the money: 10 * 0 rub., 5 * 1 rub. Ready to work The amount: 50 Turned off	Ready to work The amount: 50 There is not enough money The amount: 60 The amount: 70 The amount: 75 Take the coffee c Take the change: 10 * 0 rub., 5 * 1 rub. Ready to work The amount: 5 Take the money: 10 * 0 rub., 5 * 1 rub. Ready to work The amount: 50 Turned off
1 a 10 1 1 SHOWTREE	Ready to work app system is ready settings_and_commands is ready buttons is ready money_taker is ready money_checker is	Ready to work app system is ready settings_and_commands is ready buttons is ready money_taker is ready money_checker is

Входные данные	Ожидаемые выходные данные	Фактические выходные данные
	ready coffee_giver is ready	ready coffee_giver is ready

ЗАКЛЮЧЕНИЕ

В настоящей курсовой работе было разработано и реализовано решение задачи по моделированию работы кофемашины. Был описан метод решения, составлен дискретный, понятный, определенный, результативный, массовый алгоритм выполнения поставленной задачи, на его основе построены блок-схемы, представляющие графическую интерпретацию алгоритма. На основе вышеупомянутого алгоритма был сформирован код задачи, разделенный на подзадачи, которые распределены по соответствующим файлам. Также была проведена отладка, проверка работоспособности и тестирование разработанного кода.

В ходе выполнения работы были получены навыки проектирования и реализации задач в стиле ООП, произведено знакомство с основными принципами ООП, а также были получены навыки по работе с сигналами и обработчиками и установки связей между ними.

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_laboratornyh_rabot_3.pdf (дата обращения 20.05.2022).

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 20.05.2022).

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).