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 программирование (ООП) - это методология, которая представляет программу как совокупность объектов, обладающих свойствами и методами, и образующих иерархию наследования. ООП важно, поскольку оно обеспечивает инкапсуляцию (скрытие деталей реализации), наследование (создание новых классов на основе существующих), полиморфизм (присваивание различных значений одному и тому же имени метода или оператора в различных контекстах) и абстракцию (моделирование реального мира). Это делает программы более модульными, понятными, гибкими и эффективными в плане повторного использования кода.

Получение практических навыков разработки и написания программного продукта с использованием парадигмы ООП является ключевым шагом в становлении успешным программистом. Это позволяет глубже понять, как проектировать и строить эффективные системы. Научиться создавать модули, которые можно легко повторно использовать и интегрировать, используя принципы инкапсуляции. Также получение навыков в области наследования и полиморфизма, что позволит создавать более гибкие и масштабируемые приложения. В конечном итоге, практический опыт работы с ООП поможет стать более компетентным и уверенным разработчиком.

Целью данной курсовой работы является разработка системы имитирующей работу вендингово автомата, с применением обьектно-ориентированной парадигмы.

1 ПОСТАНОВКА ЗАДАЧИ

Надо моделировать работу вендингово автомата следующей конструкции. Вендинговый автомат состоит из следующих элементов:

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

Пульт управления содержит кнопки:

- выбора продукта по его номеру (множество кнопок);
- возврата денег.

Правила работы с вендинговым автоматом.

Вендинговый автомат готовиться к работе следующим образом:

Задается количество типов продуктов.

1. Загружается исходное количество монет для выдачи сдачи с достоинством пять и десять рублей.

2. Каждому типу продукта соответствует: номер, количество экземпляров продукта для загрузки, стоимость, название. Стоимость кратное 5 рублям. Загружаются продукты.

3. После этого выводиться сообщение о готовности вендингового автомата к работе.

4. После готовности вендингового автомата, для покупки продукта выполняются следующие действия:

- ввод денег (монет, купюр) достоинством 5, 10, 50 или 100 рублей. При вводе денег осуществляется суммирование, сумма отображается на экране;
- выбор продукта по номеру, если денег достаточно, то выдается продует и

сдача, при наличии. Выводиться сообщение о готовности автомата к работе.

- после выбора продукта, если денег недостаточно, автомат сообщает о недостаточности средств.
- возврат денег, возвращаются все внесенные средства и выводиться сообщение о готовности автомата к работе.

Устройство возврата сдачи может вернуть только монеты с 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.
- Product «номер продукта» - последовательное нажатие соответствующих кнопок пульта управления (выбор продукта).
- Refund money – нажатие кнопки «вернуть деньги».
- Turn off the system – завершение работы системы.

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

Построить систему, которая использует объекты:

1. Объект «система».

2. Объект для чтения данных для подготовки и команд. Считывает данные для подготовки, загрузки и настройки вендингового автомата. После чтения очередной порции данных (строки) для настройки или данных команды, объект выдает сигнал с текстом полученных данных. Все данные настройки и данные команды синтаксический корректны.

3. Объект пульта управления, для отработки нажатия кнопок выбора продукта — номер продукта. Объект после нажатия кнопок (номер продукта двухзначное натуральное число) анализирует: корректность набора номера, наличие продукта данного типа, достаточность средств для выбранного продукта и выдает соответствующий сигнал.

4. Объект, моделирующий устройство приема денег. После принятия очередной монеты или 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 выводиться сообщение о готовности вендингового автомата к работе.

6. Объект, моделирующий устройство выдачи продукта. Выдает сигнал, содержащий текст о выдаче продукта. После выдачи продукта выдает сигнал о готовности вендингового автомата к работе.

7. Объект для вывода состояния или результата выполнения действия вендингового автомата на консоль.

Написать программу, реализующую следующий алгоритм:

1. Вызов метода объекта «система» build_tree_objects ( ).

1.1. Построение дерева иерархии объектов.

1.2. Установка связей сигналов и обработчиков между объектами.

2. Вызов метода объекта «система» exec_app ( ).

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 данных и команд.

2.3. После ввода команды «Turn off the system» 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.В состав тестов разработчика добавить тест, который вводит данные настройки и далее выдает команду «SHOWTREE».

1.1 Описание входных данных

Данные для первоначальной загрузки и настройки вендингового автомата.

Первая строка.

```
«натуральное число»
```

Задает количество типов загружаемых продуктов. Значение не более 90.

Вторая строка

```
«натуральное число 1» «натуральное число 2»
```

Строка содержит исходное количество монет для выдачи сдачи.

```
«натуральное число 1» - количество монет с достоинством 5 рублей.
«натуральное число 2» - количество монет с достоинством 10 рублей.
```

Выполняется операция первоначальной загрузки монет для сдачи.

Строки начиная с третьей в количестве типов загружаемых продуктов содержат:

```
«натуральное число 1» «натуральное число 2» «натуральное число 3» «название продукта»
«натуральное число 1» - номер продукта, значение от 10 до 99.
«натуральное число 2» - количество загружаемых продуктов данного типа, не более 20.
«натуральное число 3» - цена продукта, кратное 5.
«название продукта» - строка, наименование продукта.
```

После ввода каждой строки выполняется операция загрузки продукта, настройки количества и цены.

Завершается подготовка автомата и он переводится в состояние готовности.

Последующие строки содержат команды для функционирования вендингового автомата (нажатия на кнопки или подача денег).

Подача денег, монет и купюр с достоинством 5, 10, 50 или 100 рублей.

```
«натуральное число»
```

Возврат денег

```
Refund money
```

Выбор продукта

```
Product «номер продукта»
```

Последняя команда присутствует всегда

```
Turn off the system
```

Пример ввода:

```
10
13 50
10 20 50 Water Holy Spring
11 20 45 Water Shishkin forest
12 15 115 Juice Gardens of the Don region
13 11 40 Mineral water Essentuki № 4
14 18 150 Juice Gardens of the Don region
25 20 85 Chocolate Alyonka
19 20 130 Ritter Sport dark chocolate with mint filling
77 9 50 Chocolate Kinder
31 20 65 Baunty
44 1 35 Nougat Stylish things with almonds and cranberries in milk chocolate
glaze
50
100
Product 14
10
10
10
Product 44
10
Product 44
5
5
Refund money
100
Product 44
Product 45
Product 25
50
10
Turn off the system
```

1.2 Описание выходных данных

Шаблоны текстов, которые отображаются на консоли:

Первый раз сообщение о готовности к работе автомата отображается после завершения загрузки исходной сдачи и продуктов. Это сообщение отображается после завершения множества команд от одного клиента. Сообщение сигнализирует о готовности автомата для обслуживания нового клиента.

```
Ready to work
```

Сумма после ввода очередной монеты или купюры.

```
The amount: «сумма денег»
```

Сообщение о выдачи продукта

```
Take the product «наименование продукта»
```

Сообщения для получения сдачи:

```
Take the change: 10 * «количество десяти рублевых монет» rub., 5 * «количество пяти рублевых монет» rub.
Take the change: 10 * «количество десяти рублевых монет» rub.
Take the change: 5 * «количество пяти рублевых монет» rub.
```

Сообщение об отсутствия продукта с указанным номером

```
There is no product with this number
```

Сообщение об отсутствия продукта

```
There is no product
```

Сообщение о недостаточности средств

```
There is not enough money
```

Сообщение для получения введенных денег обратно:

```
Take the change: 10 * «количество десяти рублевых монет» rub., 5 * «количество пяти рублевых монет» rub.
Take the change: 10 * «количество десяти рублевых монет» rub.
Take the change: 5 * «количество пяти рублевых монет» rub.
```

Сообщение о возврате 50 или 100 рублевой купюры :

```
Take the money back, no change
```

Сообщение о завершении работы автомата:

```
Turned off
```

Пример вывода

```
Ready to work
The amount: 50
The amount: 150
Take the product Juice Gardens of the Don region
Ready to work
The amount: 10
The amount: 20
The amount: 30
There is not enough money
The amount: 40
Take the product Nougat Stylish things with almonds and cranberries in milk
chocolate glaze
Take the change: 5 * 1 rub.
Ready to work
The amount: 5
The amount: 10
Take the change: 10 * 1 rub.
Ready to work
The amount: 100
There is no product
There is no product with this number
Take the product Chocolate Alyonka
Take the change: 10 * 1 rub.,  5 * 1 rub.
Ready to work
The amount: 50
The amount: 60
Turned off
```

2 МЕТОД РЕШЕНИЯ

Для решения задачи используется:

- объект класса Class_base предназначен для ;
- объект класса Class_Get_Change предназначен для ;
- объект класса Class_Get_Product предназначен для ;
- объект класса Class_Print предназначен для ;
- объект класса Class_Pult предназначен для ;
- объект класса Class_Read предназначен для ;
- объект класса Class_Set_Money предназначен для ;
- объект класса Class_System предназначен для ;
- функция delete для удаления объекта и очистки памяти;
- функция getline для позволяет считать строки, включая пробелы;
- cin - объект стандартного потока ввода с клавиатуры;
- cout - объект стандартного потока вывода на экран;
- if, else - условный оператор(оператор ветвления);
- for - оператор цикла;
- while - оператор цикла;
- для работы с входными и выходными данными используется библиотека iostream;
- для работы со строками используется бибиотека string;
- для работы с векторами используется библиотека vector;
- структура object_sh с полями b_signal типа указатель на метод signal_f, b_target типа указатель на объект класса Class_base, b_handler типа указатель на метод handler_f;
- структура product_info с целочисленными полями numbber_product, count_product, price_product, строковой переменной name_product.

Класс Class_base:

- свойства/поля:
  - поле хранение имени объекта:
    - наименование — b_name;
    - тип — string;
    - модификатор доступа — private;
  - поле хранение указателя на головной объект:
    - наименование — b_head_name;
    - тип — указатель;
    - модификатор доступа — private;
  - поле хранение подчинённых объектов головного объекта:
    - наименование — b_sub_objects;
    - тип — vector;
    - модификатор доступа — private;
  - поле хранение состояния объекта:
    - наименование — b_state;
    - тип — int;
    - модификатор доступа — private;
  - поле хранение связей с другими объектами:
    - наименование — connects;
    - тип — vector;
    - модификатор доступа — private;
  - поле хранение количества доступных денег для сдачи пяти рублёвыми монетами:
    - наименование — change_five;
    - тип — int;
    - модификатор доступа — public;

- поле хранение количества доступных денег для сдачи десяти рублёвыми монетами:
  - наименование — change_ten;
  - тип — int;
  - модификатор доступа — public;
- поле хранение суммы денег внесённых пользователем в автомат:
  - наименование — sum;
  - тип — int;
  - модификатор доступа — public;

- функционал:
  - метод Class_base — создание объекта;
  - метод ~Class_base — удаление объекта;
  - метод setName — установка имени из параметра;
  - метод getName — возврат имени объекта;
  - метод getHead — возврат указателя на головной объект;
  - метод search_object_sub_by_name — метод осуществляет поиск объекта по имени от корня и возвращает указатель на первое вхождение объекта с требуемым именем, то есть поиск по ветке;
  - метод search_Object_branch — метод осуществляет поиск объекта по имени от корня и возвращает указатель с требуемым именем, то есть поиск по lthtde;
  - метод setObjectState — установка готовности объекта, в качестве параметра передаётся переменная целого типа, содержит номер состояния;
  - метод showObjectAndSubObjects — вывод иерархии объектов(дерева или ветки) от текущего объекта;
  - метод getSubObject — поиск подчинённого объекта по имени;

- метод deleteObjectName — метод удаления подчинённого объекта по имени;
- метод getObjectCoordinate — метод получения указателя на любой объект в составе дерева иерархии объектов согласно пути(координате);
- метод showObjectState — вывод иерархии объектов(дерева или ветки) и отметок их готовности от текущего объекта;
- метод changeHead — метод переопределения головного объекта для текущего в дереве иерархии;
- метод set_connection — метод установки связи между сигналом текущего объекта и обработчиком целевого объекта;
- метод delete_connection — метод удаления(разрыва) связи между сигналом текущего объекта и обработчиком целевого объекта;
- метод emit_signal — метод выдачи сигнала от текущего объекта с передачей строковой переменной;
- метод getPath — метод определения абсолютного пути до текущего объекта;
- метод setStateAll — метод установки состояния переданного в параметр всем объектам дерева или ветки.

Класс Class_System:

- функционал:
  - метод exec_app — запуск приложения и установка связей сигналов и обработчиков между объектами;
  - метод build_tree_objects — построение дерева иерархии объектов и установка связей сигналов и обработчиков между объектами.

Класс Class_Get_Change:

- функционал:

- метод signal_f — сигнал с количеством возвращаемой суммы;
- метод handler_f — обработка полученных данных.

Класс Class_Get_Product:

- функционал:
  - метод signal_f — сигнал с текстом выдачи товара;
  - метод handler_f — обработка сигнала.

Класс Class_Print:

- функционал:
  - метод handler_f — вывод полученных данных.

Класс Class_Pult:

- функционал:
  - метод signal_f — сигнал пульта управления;
  - метод handler_f — обработа данных и проверка на возможность покупки товара.

Класс Class_Read:

- функционал:
  - метод signal_f — сигнал с данными, после обработки команд;
  - метод handler_f — обработка входных данных и команд;
  - метод signal_start — сигнал начала работы вендингово автомата.

Класс Class_Set_Money:

- функционал:
  - метод signal_f — сигнал с количеством внесённой в автомат суммы денежных средств;
  - метод handler_f — обработка полученных данных.

Таблица 1 – Иерархия наследования классов

№	Имя класса	Классы-наследники	Модификатор доступа при наследовании	Описание	Номер
1	Class_base			Базовый класс с основными полями и методами	
		Class_System	public		2
		Class_Get_Change	public		3
		Class_Get_Product	public		4
		Class_Print	public		5
		Class_Pult	public		6
		Class_Read	public		7
		Class_Set_Money	public		8
2	Class_System			Класс отвечающий за начало работы автомата	
3	Class_Get_Change			Класс отвечающий за возврат денег и выдачу сдачи	
4	Class_Get_Product			Класс отвечающий за выдачу товара	
5	Class_Print			Класс отвечающий за вывод состояния и информации	
6	Class_Pult			Класс пульта управления вендинговым автоматом	
7	Class_Read			Класс обрабатывающий входные данные и команды	
8	Class_Set_Money			Класс устройства приёма денег	

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: нет.

Возвращаемое значение: целочисленное - индикатор успешности выполнения программы.

Алгоритм функции представлен в таблице 2.

Таблица 2 – Алгоритм функции main

№	Предикат	Действия	№ перехода
1		Создание объекта ob_cl_application класса Class_System с передачей в параметр пустой указатель	2
2		Вызов метода build_tree_objects класса Class_System через обращение к объекту ob_cl_application класса Class_System	3
3		Вызов метода exec_app класса Class_System через обращение к объекту ob_cl_application класса Class_System	4
4		Возврат 0 - успешное выполнение программы	∅

3.2 Алгоритм метода exec_app класса Class_System

Функционал: Запуск приложения и установка связей сигналов и обработчиков между объектами.

Параметры: нет.

Возвращаемое значение: целочисленный - индикатор успешности запуска программы.

Алгоритм метода представлен в таблице 3.

Таблица 3 – Алгоритм метода exec_app класса Class_System

№	Предикат	Действия	№ перехода
1		Вызов метода setStateAll с параметром 1 объекта base_s	2
2		Объявление строковой переменной cmd	3
3		Ввод с клавиатуры значения переменной cmd	4
4		Вызов метода emit_signal объекта System с параметрами указателя на метод signal_start класса Class_Read, переменной cmd, указателем на объект "Print"	5
5	Бесконечный	Считывание всей строки с клавиатуры в переменную cmd с помощью getline	6
			9
6	cmd равно "Turn off the system" или cmd равно "SHOWTREE"	Вызов метода emit_signal объекта System с параметрами указателя на метод signal_f класса Class_Read, переменной cmd, указателем на объект "Read"	7
			8
7		Завершение работы цикла	9
8		Вызов метода emit_signal объекта System с параметрами указателя на метод signal_f класса Class_Read, переменной cmd, указателем на объект "Read"	6
9		Возврат 0 - успешный запуск программы	∅

3.3 Алгоритм метода build_tree_objects класса Class_System

Функционал: Построение дерева иерархии объектов и установка связей сигналов и обработчиков между объектами.

Параметры: нет.

Возвращаемое значение: Без возвращаемого значения.

Алгоритм метода представлен в таблице 4.

Таблица 4 – Алгоритм метода build_tree_objects класса Class_System

№	Предикат	Действия	№ перехода
1		Вызов метода setName с параметром "System"	2
2		Инициализация вектора типа TYPE_SIGNAL указателями на сигналы(signal_f) всех классов	3
3		Инициализация вектора типа TYPE_HANDLER указателями на обработчики(handler_f) всех классов	4
4		Создание объекта класса Class_Read через параметризированный конструктор с параметрами указателя на текущий объект и "Read"	5
5		Создание объекта класса Class_Pult через параметризированный конструктор с параметрами указателя на текущий объект и "Pult"	6
6		Создание объекта класса Class_Set_Money через параметризированный конструктор с параметрами указателя на текущий объект и "SetMoney"	7
7		Создание объекта класса Class_Get_Change через параметризированный конструктор с параметрами указателя на текущий объект и "GetChange"	8
8		Создание объекта класса Class_Print через параметризированный конструктор с параметрами указателя на текущий объект и "Print"	9
9		Создание объекта класса Class_Get_Product через параметризированный конструктор с параметрами указателя на текущий объект и "GetProduct"	10

№	Предикат	Действия	№ перехода
10		Вызов метода set_connection объекта "System" для установки связи с объектом "Read" через методы signal_f класса Class_Read и handler_f класса Class_Read	11
11		Вызов метода set_connection объекта "Read" для установки связи с объектом "SetMoney" через методы signal_f класса Class_Read и handler_f класса Class_Set_Money	12
12		Вызов метода set_connection объекта "SetMoney" для установки связи с объектом "Print" через методы signal_f класса Class_Set_Money и handler_f класса Class_Print	13
13		Вызов метода set_connection объекта "Read" для установки связи с объектом "Print" через методы signal_f класса Class_Read и handler_f класса Class_Print	14
14		Вызов метода set_connection объекта "System" для установки связи с объектом "Print" через методы signal_f класса Class_Read и handler_f класса Class_Print	15
15		Вызов метода set_connection объекта "Read" для установки связи с объектом "GetChange" через методы signal_f класса Class_Read и handler_f класса Class_Get_Change	16
16		Вызов метода set_connection объекта "GetChange" для установки связи с объектом "Print" через методы signal_f класса Class_Get_Change и handler_f класса Class_Print	17
17		Вызов метода set_connection объекта "Read" для установки связи с объектом "GetChange" через методы signal_f класса Class_Read и handler_f класса Class_Get_Change	18
18		Вызов метода set_connection объекта "Pult" для установки связи с объектом "Print" через методы signal_f класса Class_Pult и handler_f класса Class_Print	19
19		Вызов метода set_connection объекта "Pult" для установки связи с объектом "GetProduct" через методы signal_f класса Class_Pult и	20

№	Предикат	Действия	№ перехода
		handler_f класса Class_Get_Product	
20		Вызов метода set_connection объекта "GetProduct" для установки связи с объектом "Print" через методы signal_f класса Class_Get_Product и handler_f класса Class_Print	21
21		Вызов метода set_connection объекта "GetProduct" для установки связи с объектом "GetChange" через методы signal_f класса Class_Get_Product и handler_f класса Class_Get_Change	∅

3.4 Алгоритм метода signal_f класса Class_Get_Change

Функционал: Сигнал с количеством возвращаемой суммы.

Параметры: Строковый, ссылка на переменную msg.

Возвращаемое значение: Без возвращаемого значения.

Алгоритм метода представлен в таблице 5.

Таблица 5 – Алгоритм метода signal_f класса Class_Get_Change

№	Предикат	Действия	№ перехода
1	в переменной msg символ "/" не найден		2
			5
2	в переменной msg от 0 до индекса символа "$" значение строки равно "0" и в переменной msg до индекса символа "$" плюс 1 значение строки не равно "0"	Присвоение переменной msg переход на новую строку, "Take the change: 5 * ", msg до индекса символа "$" плюс 1, " rub."	∅
			3
3	в переменной msg от 0 до индекса символа "$"	Присвоение переменной msg переход на новую строку, "Take the change: 10 * ",	∅

№	Предикат	Действия	№ перехода
	значение строки не равно "0" и в переменной msg до индекса символа "$" плюс 1 значение строки равно "0"	msg от 0 до индекса символа "$", " rub."	
			4
4	в переменной msg от 0 до индекса символа "$" значение строки не равно "0" и в переменной msg до индекса символа "$" плюс 1 значение строки не равно "0"	Присвоение переменной msg переход на новую строку, "Take the change: 10 * ", msg от 0 до индекса символа "$", " rub., 5 * ", msg до индекса символа "$" плюс 1, " rub."	∅
			∅
5	в переменной msg от 0 до индекса символа "/" значение строки равно "0" и в переменной msg до индекса символа "/" плюс 1 значение строки не равно "0"	Присвоение переменной msg переход на новую строку, "Take the change: 5 * ", msg до индекса символа "/" плюс 1, " rub.", переход на новую строку, "Ready to work"	∅
			6
6	в переменной msg от 0 до индекса символа "/" значение строки не равно "0" и в переменной msg до индекса символа "/" плюс 1 значение строки равно "0"	Присвоение переменной msg переход на новую строку, "Take the change: 10 * ", msg от 0 до индекса символа "/", " rub.", переход на новую строку, "Ready to work"	∅
			7
7	в переменной msg от 0 до индекса символа "/" значение строки не равно "0" и в переменной msg до индекса	Присвоение переменной msg переход на новую строку, "Take the change: 10 * ", msg от 0 до индекса символа "/", " rub., 5 * ", msg до индекса символа "/" плюс 1, " rub.", переход на	∅

№	Предикат	Действия	№ перехода
	символа "/" плюс 1 значение строки не равно "0"	новую строку, "Ready to work"	
			∅

3.5 Алгоритм метода handler_f класса Class_Get_Change

Функционал: Обработка полученных данных.

Параметры: Строковый, msg - переданное сообщение.

Возвращаемое значение: Без возвращаемого значения.

Алгоритм метода представлен в таблице 6.

Таблица 6 – Алгоритм метода handler_f класса Class_Get_Change

№	Предикат	Действия	№ перехода
1		Вызов метода emit_signal с параметрами указателя на метод signal_f класса Class_Get_Change, значение msg, указателем на объект "Print"	∅

3.6 Алгоритм метода signal_f класса Class_Get_Product

Функционал: Сигнал с текстом выдачи товара.

Параметры: Строковый, ссылка на переменную msg.

Возвращаемое значение: Без возвращаемого значения.

Алгоритм метода представлен в таблице 7.

Таблица 7 – Алгоритм метода signal_f класса Class_Get_Product

№	Предикат	Действия	№ перехода
1	msg не равно "Ready to work"	присвоение переменной msg "Take the product", msg	∅
			∅

3.7 Алгоритм метода handler_f класса Class_Get_Product

Функционал: Обработка сигнала.

Параметры: Строковый, msg - переданное сообщение.

Возвращаемое значение: Без возвращаемого значения.

Алгоритм метода представлен в таблице 8.

Таблица 8 – Алгоритм метода handler_f класса Class_Get_Product

№	Предикат	Действия	№ перехода
1	символ "/" в переменной msg не найден	Вызов метода emit_signal с параметрами указателя на метод signal_f класса Class_Get_Product, значение msg, указателем на объект "Print"	4
			2
2		Вызов метода emit_signal с параметрами указателя на метод signal_f класса Class_Get_Product, значение msg от 0 индекса до индекса символа "/", указателем на объект "Print"	3
3		Вызов метода emit_signal с параметрами указателя на метод signal_f класса Class_Read, значение msg от 0 индекса до индекса символа "/" плюс 1, указателем на объект "GetChange"	4
4		Вызов метода emit_signal с параметрами указателя на метод signal_f класса Class_Get_Product, "\nReady to work", указателем на объект "Print"	∅

3.8 Алгоритм метода handler_f класса Class_Print

Функционал: Вывод полученных данных.

Параметры: Строковый, ссылка на msg - переданное сообщение.

Возвращаемое значение: Без возвращаемого значения.

Алгоритм метода представлен в таблице 9.

Таблица 9 – Алгоритм метода handler_f класса Class_Print

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

3.9 Алгоритм метода signal_f класса Class_Pult

Функционал: Сигнал пульта управления.

Параметры: Строковый, ссылка на переменную msg.

Возвращаемое значение: Без возвращаемого значения.

Алгоритм метода представлен в таблице 10.

Таблица 10 – Алгоритм метода signal_f класса Class_Pult

№	Предикат	Действия	№ перехода
1	msg равно "money"	msg присвоение "\nThere is not enough money"	2
			2
2	msg равно "product"	msg присвоить "\nThere is no product"	3
			3
3	msg равно "number"	msg присвоить "\nThere is no product with this number"	∅
			∅

3.10 Алгоритм метода handler_f класса Class_Pult

Функционал: Обработа данных и проверка на возможность покупки товара.

Параметры: Строковый, msg - переданное сообщение.

Возвращаемое значение: Без возвращаемого значения.

Алгоритм метода представлен в таблице 11.

Таблица 11 – Алгоритм метода handler_f класса Class_Pult

№	Предикат	Действия	№ перехода
1		Инициализация переменной flag логического типа значением true	2
2		Инициализация переменной - счётчика i равно 0	3
3	i меньше значения вектора product объекта System		4
			9
4	значение msg равно значению number_product в векторе product по индексу i и стоимость товара в векторе product по индексу i меньше или равно значению переменной sum объекта System и count_product в векторе product по индексу i больше 0	Вычитание из переменной sum объекта System значение price_product в векторе product по индексу i	5
			8
5		Присвоение значению flag значение false	6
6		Вычитание единицы из значения cout_product в векторе product по индексу i	7
7	Значение переменной sum объекта System равно 0	Вызов метода emit_signal с параметрами указателя на метод signal_f класса Class_Pult, значение name_product в векторе product по индексу i, указателем на объект "GetProduct" Возврат - завершение работы метода	∅
		Вызов метода emit_signal с параметрами указателя на метод signal_f класса Class_Pult, значение name_product в векторе product по индексу i + "/" +	∅

№	Предикат	Действия	№ перехода
		"change", указателем на объект "GetProduct" Возврат - завершение работы метода	
8		Инкремент i	3
9	Значение переменной flag равно true	Инициализация переменной - счётчика i равно 0	10
			∅
10	i меньше значения вектора product объекта System		11
			15
11	значение msg равно значению number_product в векторе product по индексу i		12
			14
12	count_product в векторе product по индексу i меньше или равен 0	Вызов метода emit_signal с параметрами указателя на метод signal_f класса Class_Pult, "product", указателем на объект "Print"	14
			13
13	стоимость товара в векторе product по индексу i больше значеня переменной sum объекта System	Вызов метода emit_signal с параметрами указателя на метод signal_f класса Class_Pult, "money", указателем на объект "Print" Возврат - завершение работы метода	∅
			14
14		Инкремент i	11
15		Вызов метода emit_signal с параметрами указателя на метод signal_f класса Class_Pult, "number", указателем на объект "Print"	∅

3.11 Алгоритм метода signal_f класса Class_Read

Функционал: Сигнал с данными, после обработки команд.

Параметры: Строковый, ссылка на переменную msg.

Возвращаемое значение: Без возвращаемого значения.

Алгоритм метода представлен в таблице 12.

Таблица 12 – Алгоритм метода signal_f класса Class_Read

№	Предикат	Действия	№ перехода
1	значение msg равно "Refund money"	Объявление переменных fife, ten целого типа	2
			9
2		Присвоение ten значение переменной sum объекта "System" деленного на 10	3
3		Вычитание из переменной sum объекта "System" значения ten умноженного на 10	4
4		Присвоение ten значение переменной sum объекта "System" деленного на 10	5
5		Вычитание из переменной sum объекта "System" значения ten умноженного на 5	6
6		Вычитание из переменной change_five объекта "System" значения fife	7
7		Вычитание из переменной change_ten объекта "System" значения ten	8
8		Присвоение переменной msg значение "Refund money", значение ten приведённого к строковому типу с помощью функции to_string, "/", значение fife приведённого к строковому типу с помощью функции to_string	∅
9	значение msg равно "change"	Объявление переменных fife, ten целого типа	10

№	Предикат	Действия	№ перехода
			17
10		Присвоение ten значение переменной sum объекта "System" деленного на 10	11
11		Вычитание из переменной sum объекта "System" значения ten умноженного на 10	12
12		Присвоение ten значение переменной sum объекта "System" деленного на 10	13
13		Вычитание из переменной sum объекта "System" значения ten умноженного на 5	14
14		Вычитание из переменной change_five объекта "System" значения fife	15
15		Вычитание из переменной change_ten объекта "System" значения ten	16
16		Присвоение переменной msg значение ten приведённого к строковому типу с помощью функции to_string, "$", значение fife приведённого к строковому типу с помощью функции to_string	∅
17	значение msg равно "false"	Присвоение переменной msg значение "\nTake the money back, no change"	∅
			∅

3.12 Алгоритм метода handler_f класса Class_Read

Функционал: Обработка входных данных и команд.

Параметры: Строковый, msg - переданное сообщение.

Возвращаемое значение: Без возвращаемого значения.

Алгоритм метода представлен в таблице 13.

Таблица 13 – Алгоритм метода handler_f класса Class_Read

№	Предикат	Действия	№ перехода
1	Значение переменной msg с индекса 0 по 12 индекс равно "Refund money"	Вызов метода emit_signal с параметрами указателя на метод signal_f класса Class_Read, msg с индекса 12 по индекс длины строки msg минус 1, указателем на объект "GetChange"	∅
			2
2	Значение переменной msg равно "Turn off the system"	Вызов метода emit_signal с параметрами указателя на метод signal_f класса Class_Read, "\nTurned off", указателем на объект "Print"	∅
			3
3	Значение переменной msg равно "SHOWTREE"	Вызов метода showObjectsState объекта "System"	∅
			4
4	Значение переменной msg равно 50		5
			6
5	Значение переменной change_five объекта "System" умноженного на 5 плюс значение переменной change_ten объекта "System" умноженного на 10 меньше 50 плюс значение переменной sum объекта "System"	Вызов метода emit_signal с параметрами указателя на метод signal_f класса Class_Read, "false", указателем на объект "Print"	∅
		Вызов метода emit_signal с параметрами указателя на метод signal_f класса Class_Read, "msg", указателем на объект "SetMoney"	∅

№	Предикат	Действия	№ перехода
6	Значение переменной msg равно 100		7
			8
7	Значение переменной change_five объекта "System" умноженного на 5 плюс значение переменной change_ten объекта "System" умноженного на 10 меньше 100 плюс значение переменной sum объекта "System"	Вызов метода emit_signal с параметрами указателя на метод signal_f класса Class_Read, "false", указателем на объект "Print"	∅
		Вызов метода emit_signal с параметрами указателя на метод signal_f класса Class_Read, "msg", указателем на объект "SetMoney"	∅
8	Значение переменной msg равно "5"	Присвоение переменной change_five плюс 1	9
			9
9	Значение переменной msg равно "10"	Присвоение переменной change_ten плюс 1	∅
			∅

3.13 Алгоритм метода signal_start класса Class_Read

Функционал: Сигнал начала работы вендингово автомата.

Параметры: Строковый, ссылка на переменную msg.

Возвращаемое значение: Без возвращаемого значения.

Алгоритм метода представлен в таблице 14.

Таблица 14 – Алгоритм метода signal_start класса Class_Read

№	Предикат	Действия	№ перехода
1		Объявление переменных целого типа n, n_product, p_product	2
2		Объявление переменной строкового типа name_product	3
3		Присвоение переменной n значение переменной msg приведенной к целому типу	4
4		Ввод с клавиатуры значения переменной change_five объекта "System"	5
5		Ввод с клавиатуры значения переменной change_ten объекта "System"	6
6		Объявление указателя product_ на объект типа product_info	7
7		Инициализация переменной - счетчика i равной 0	8
8	i меньше значения переменной n	Ввод клавиатуры значений для переменных n_product, c_product, p_product	9
			17
9		Считывание всей строки с клавиатуры в переменную nameProduct с помощью getline	10
10		Создание объекта типа product_info	11
11		Присвоение полю number_product по адресу указателя product_ значение переменной n_product	12
12		Присвоение полю count_product по адресу указателя product_ значение переменной c_product	13
13		Присвоение полю price_product по адресу указателя product_ значение переменной p_product	14
14		Присвоение полю name_product по адресу указателя product_ значение переменной name_product с индекса 1 по размер длины строки	15

№	Предикат	Действия	№ перехода
		name_product	
15		Вызов метода push_back добавление в конец вектора product объекта "System" значение по адресу указателя product_	16
16		Инкремент i	17
17		Вызов метода emit_signal с параметрами указателя на метод signal_f класса Class_Read, "Ready work", указателем на объект "Print"	∅

3.14 Алгоритм метода signal_f класса Class_Set_Money

Функционал: Сигнал с количеством внесённой в автомат суммы денежных средств.

Параметры: Строковый, ссылка на переменную msg.

Возвращаемое значение: Без возвращаемого значения.

Алгоритм метода представлен в таблице 15.

Таблица 15 – Алгоритм метода signal_f класса Class_Set_Money

№	Предикат	Действия	№ перехода
1		Присвоение переменной msg значение "\nThe amount: ", msg	∅

3.15 Алгоритм метода handler_f класса Class_Set_Money

Функционал: Обработка полученных данных.

Параметры: Строковый, msg - переданное сообщение.

Возвращаемое значение: Без возвращаемого значения.

Алгоритм метода представлен в таблице 16.

Таблица 16 – Алгоритм метода handler_f класса Class_Set_Money

№	Предикат	Действия	№ перехода
1		Присвоение переменной sum объекта "System" значения sum плюс значение переменной msg приведённой к челому типу	2
2		Вызов метода emit_signal с параметрами указателя на метод signal_f класса Class_Set_Money, значение переменной sum объекта "System" приведенной к строковому типу с помощью функции to_string, указателем на объект "Print"	∅

3.16 Алгоритм конструктора класса Class_base

Функционал: Создание объекта.

Параметры: нет.

Алгоритм конструктора представлен в таблице 17.

Таблица 17 – Алгоритм конструктора класса Class_base

№	Предикат	Действия	№ перехода
1			∅

3.17 Алгоритм деструктора класса Class_base

Функционал: Удаление объекта.

Параметры: нет.

Алгоритм деструктора представлен в таблице 18.

Таблица 18 – Алгоритм деструктора класса Class_base

№	Предикат	Действия	№ перехода
1			∅

3.18 Алгоритм метода setName класса Class_base

Функционал: Установка имени из параметра.

Параметры: нет.

Возвращаемое значение: нет.

Алгоритм метода представлен в таблице 19.

Таблица 19 – Алгоритм метода setName класса Class_base

№	Предикат	Действия	№ перехода
1			∅

3.19 Алгоритм метода getName класса Class_base

Функционал: Возврат имени объекта.

Параметры: нет.

Возвращаемое значение: нет.

Алгоритм метода представлен в таблице 20.

Таблица 20 – Алгоритм метода getName класса Class_base

№	Предикат	Действия	№ перехода
1			∅

3.20 Алгоритм метода getHead класса Class_base

Функционал: Возврат указателя на головной объект.

Параметры: нет.

Возвращаемое значение: нет.

Алгоритм метода представлен в таблице 21.

Таблица 21 – Алгоритм метода getHead класса Class_base

№	Предикат	Действия	№ перехода
1			∅

3.21 Алгоритм метода search_object_sub_by_name класса Class_base

Функционал: Метод осуществляет поиск объекта по имени от корня и возвращает указатель на первое вхождение объекта с требуемым именем, то есть поиск по ветке.

Параметры: нет.

Возвращаемое значение: нет.

Алгоритм метода представлен в таблице 22.

Таблица 22 – Алгоритм метода search_object_sub_by_name класса Class_base

№	Предикат	Действия	№ перехода
1			∅

3.22 Алгоритм метода search_Object_branch класса Class_base

Функционал: Метод осуществляет поиск объекта по имени от корня и возвращает указатель с требуемым именем, то есть поиск по lthtde.

Параметры: нет.

Возвращаемое значение: нет.

Алгоритм метода представлен в таблице 23.

Таблица 23 – Алгоритм метода search_Object_branch класса Class_base

№	Предикат	Действия	№ перехода
1			∅

3.23 Алгоритм метода setObjectState класса Class_base

Функционал: Установка готовности объекта, в качестве параметра передаётся переменная целого типа, содержит номер состояния.

Параметры: нет.

Возвращаемое значение: нет.

Алгоритм метода представлен в таблице 24.

Таблица 24 – Алгоритм метода setObjectState класса Class_base

№	Предикат	Действия	№ перехода
1			∅

3.24 Алгоритм метода showObjectAndSubObjects класса Class_base

Функционал: Вывод иерархии объектов(дерева или ветки) от текущего объекта.

Параметры: нет.

Возвращаемое значение: нет.

Алгоритм метода представлен в таблице 25.

Таблица 25 – Алгоритм метода showObjectAndSubObjects класса Class_base

№	Предикат	Действия	№ перехода
1			∅

3.25 Алгоритм метода getSubObject класса Class_base

Функционал: Поиск подчинённого объекта по имени.

Параметры: нет.

Возвращаемое значение: нет.

Алгоритм метода представлен в таблице 26.

Таблица 26 – Алгоритм метода getSubObject класса Class_base

№	Предикат	Действия	№ перехода
1			∅

3.26 Алгоритм метода deleteObjectName класса Class_base

Функционал: Метод удаления подчинённого объекта по имени.

Параметры: нет.

Возвращаемое значение: нет.

Алгоритм метода представлен в таблице 27.

Таблица 27 – Алгоритм метода deleteObjectName класса Class_base

№	Предикат	Действия	№ перехода
1			∅

3.27 Алгоритм метода getObjectCoordinate класса Class_base

Функционал: Метод получения указателя на любой объект в составе дерева иерархии объектов согласно пути(координате).

Параметры: нет.

Возвращаемое значение: нет.

Алгоритм метода представлен в таблице 28.

Таблица 28 – Алгоритм метода getObjectCoordinate класса Class_base

№	Предикат	Действия	№ перехода
1			∅

3.28 Алгоритм метода showObjectState класса Class_base

Функционал: Вывод иерархии объектов(дерева или ветки) и отметок их готовности от текущего объекта.

Параметры: нет.

Возвращаемое значение: нет.

Алгоритм метода представлен в таблице 29.

Таблица 29 – Алгоритм метода showObjectState класса Class_base

№	Предикат	Действия	№ перехода
1			∅

3.29 Алгоритм метода changeHead класса Class_base

Функционал: Метод переопределения головного объекта для текущего в дереве иерархии.

Параметры: нет.

Возвращаемое значение: нет.

Алгоритм метода представлен в таблице 30.

Таблица 30 – Алгоритм метода changeHead класса Class_base

№	Предикат	Действия	№ перехода
1			∅

3.30 Алгоритм метода set_connection класса Class_base

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

Параметры: нет.

Возвращаемое значение: нет.

Алгоритм метода представлен в таблице 31.

Таблица 31 – Алгоритм метода set_connection класса Class_base

№	Предикат	Действия	№ перехода
1			∅

3.31 Алгоритм метода delete_connection класса Class_base

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

Параметры: нет.

Возвращаемое значение: нет.

Алгоритм метода представлен в таблице 32.

Таблица 32 – Алгоритм метода delete_connection класса Class_base

№	Предикат	Действия	№ перехода
1			∅

3.32 Алгоритм метода emit_signal класса Class_base

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

Параметры: нет.

Возвращаемое значение: нет.

Алгоритм метода представлен в таблице 33.

Таблица 33 – Алгоритм метода emit_signal класса Class_base

№	Предикат	Действия	№ перехода
1			∅

3.33 Алгоритм метода getPath класса Class_base

Функционал: Метод определения абсолютного пути до текущего объекта.

Параметры: нет.

Возвращаемое значение: нет.

Алгоритм метода представлен в таблице 34.

Таблица 34 – Алгоритм метода getPath класса Class_base

№	Предикат	Действия	№ перехода
1			∅

3.34 Алгоритм метода setStateAll класса Class_base

Функционал: Метод установки состояния переданного в параметр всем объектам дерева или ветки.

Параметры: нет.

Возвращаемое значение: нет.

Алгоритм метода представлен в таблице 35.

Таблица 35 – Алгоритм метода setStateAll класса Class_base

№	Предикат	Действия	№ перехода
1			∅

4 БЛОК-СХЕМЫ АЛГОРИТМОВ

Представим описание алгоритмов в графическом виде на рисунках 1-22.

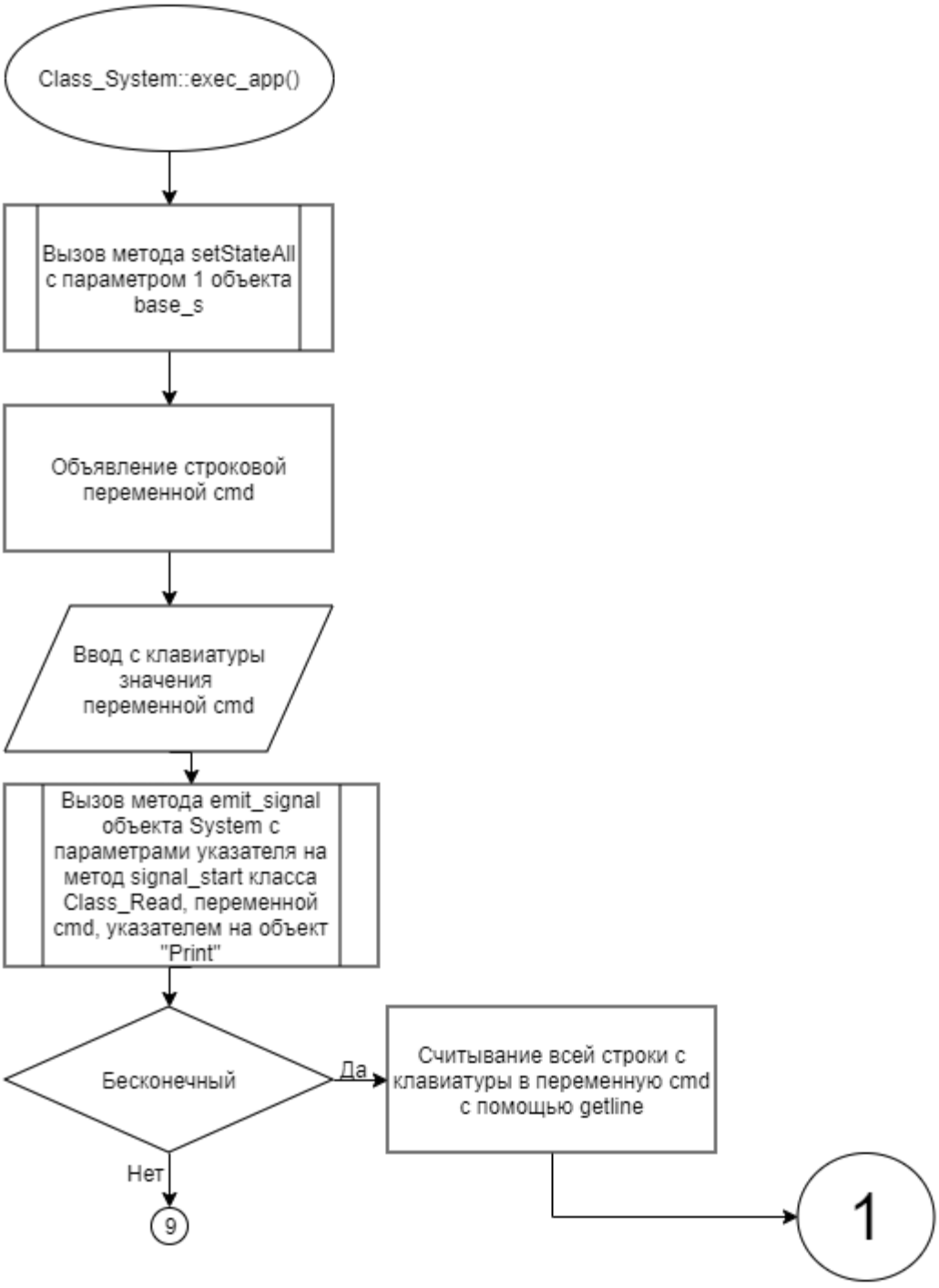

Рисунок 1 – Блок-схема алгоритма

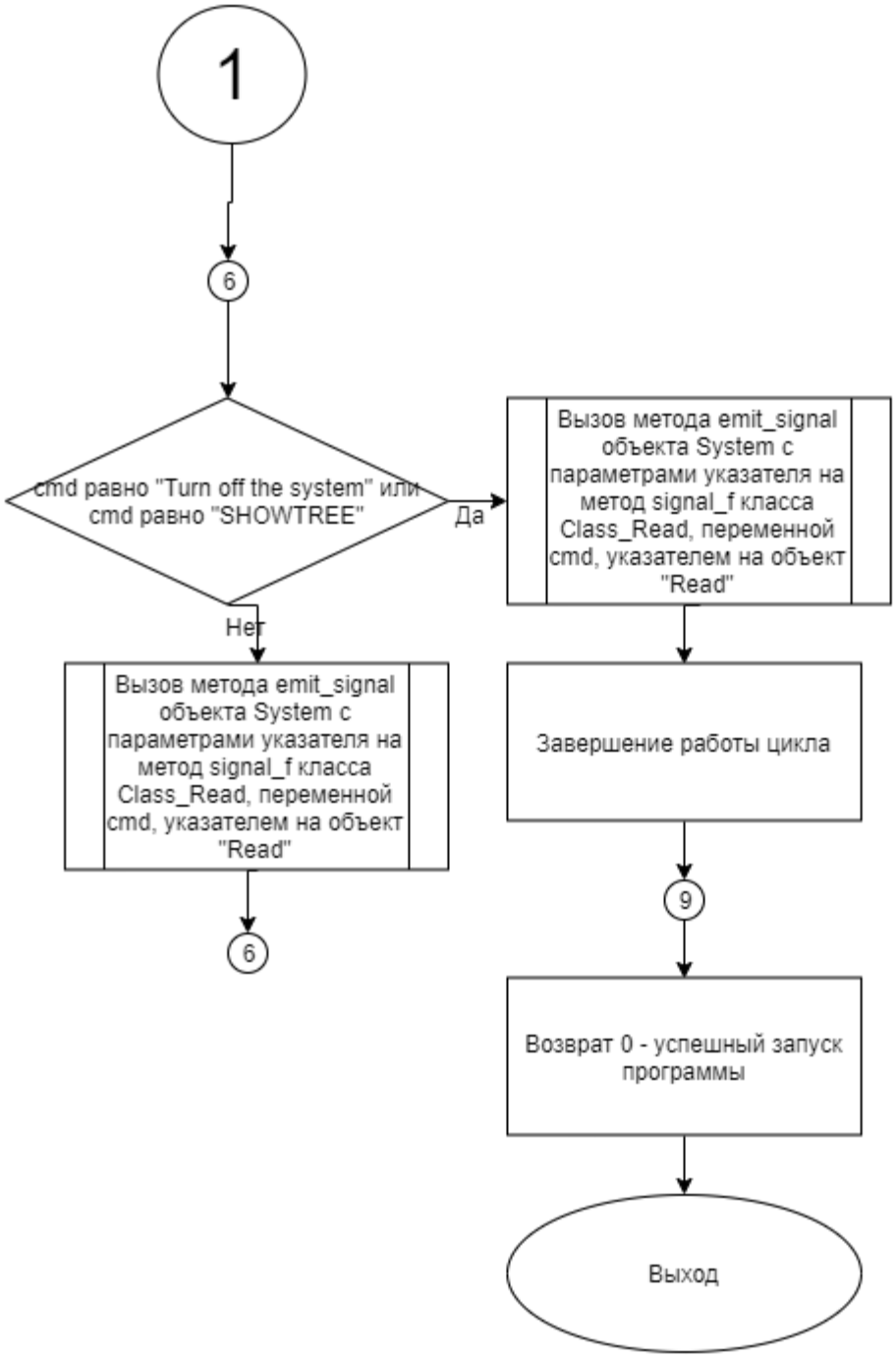

Рисунок 2 – Блок-схема алгоритма

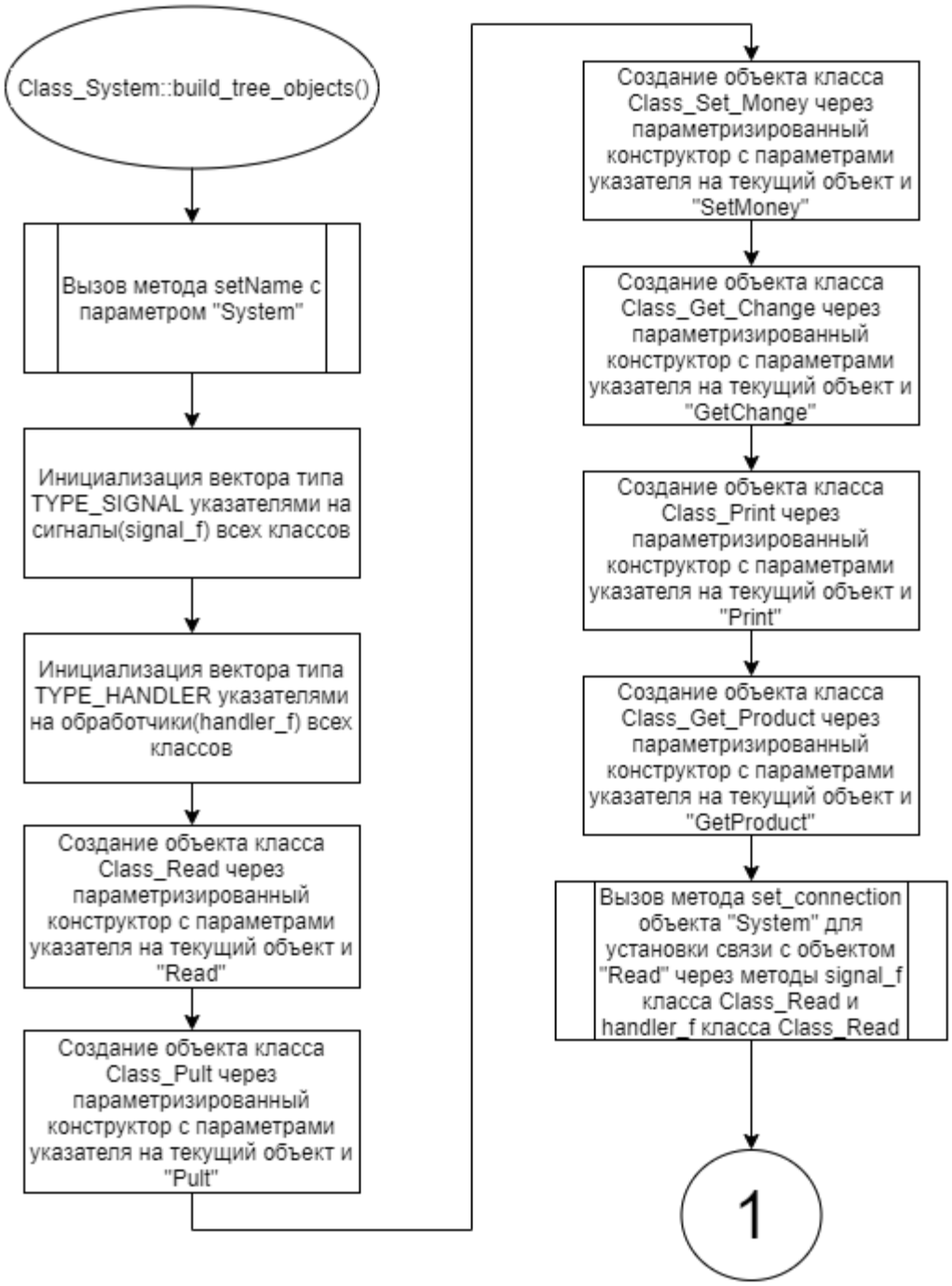

Рисунок 3 – Блок-схема алгоритма

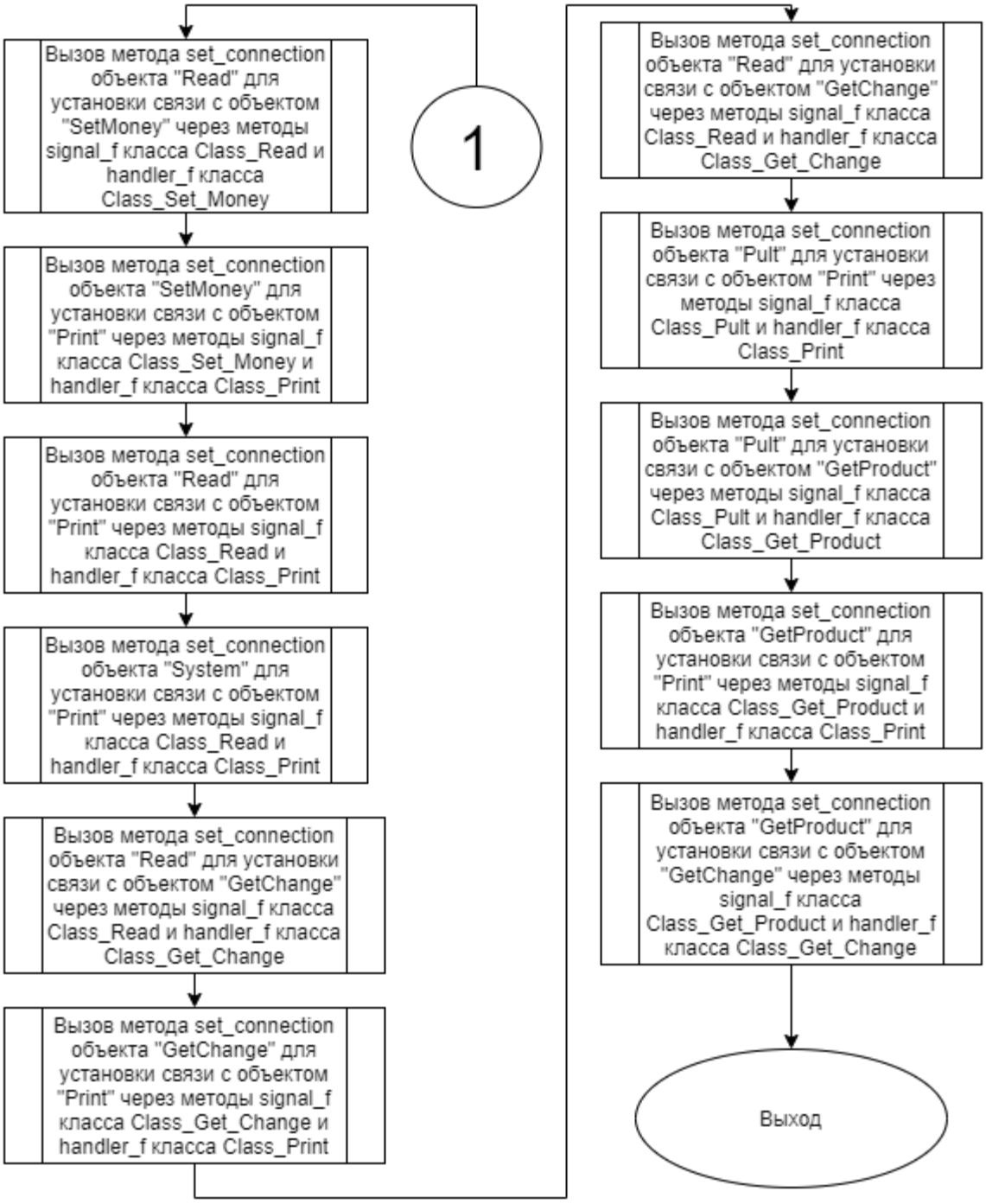

Рисунок 4 – Блок-схема алгоритма

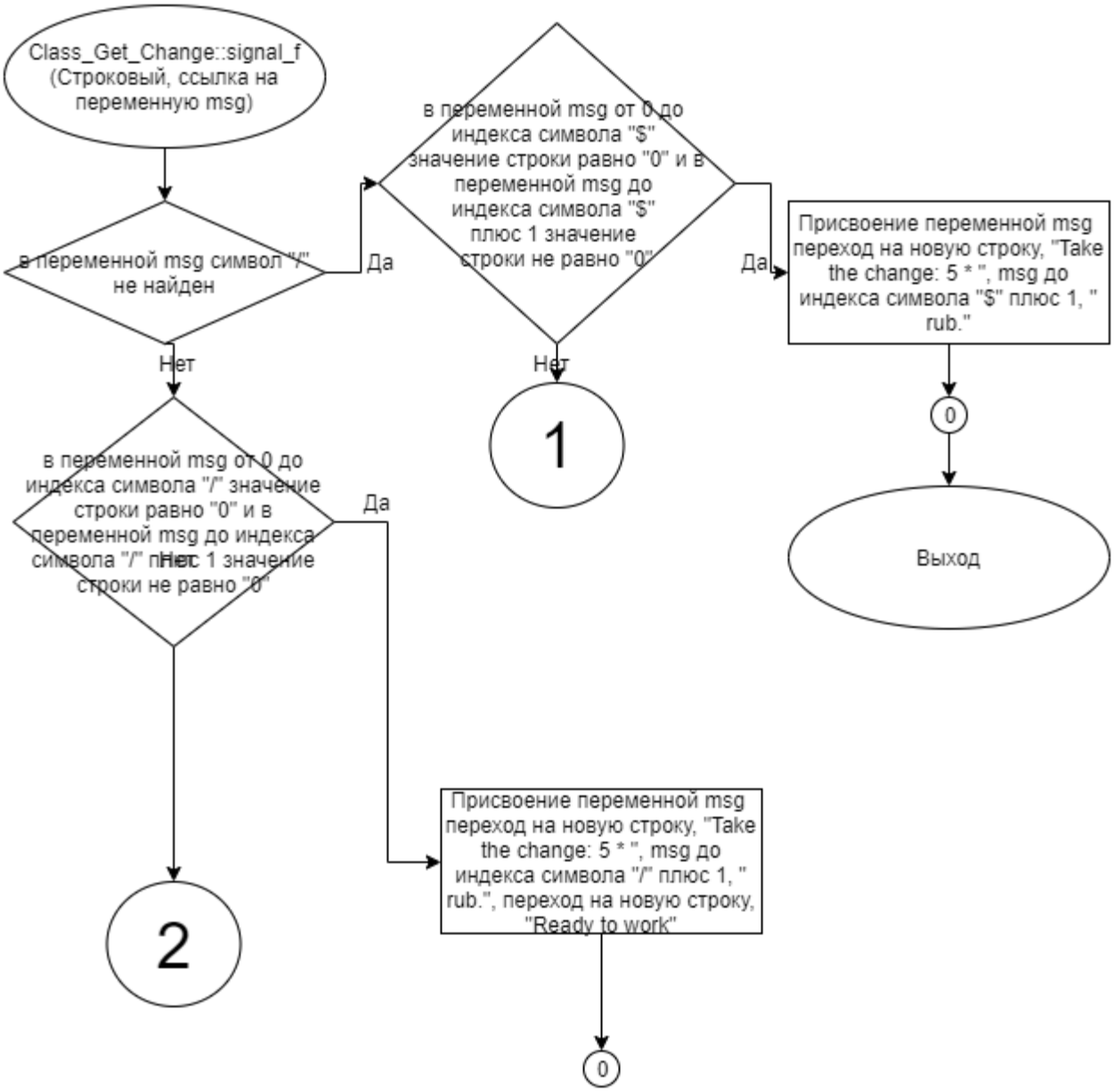

Рисунок 5 – Блок-схема алгоритма

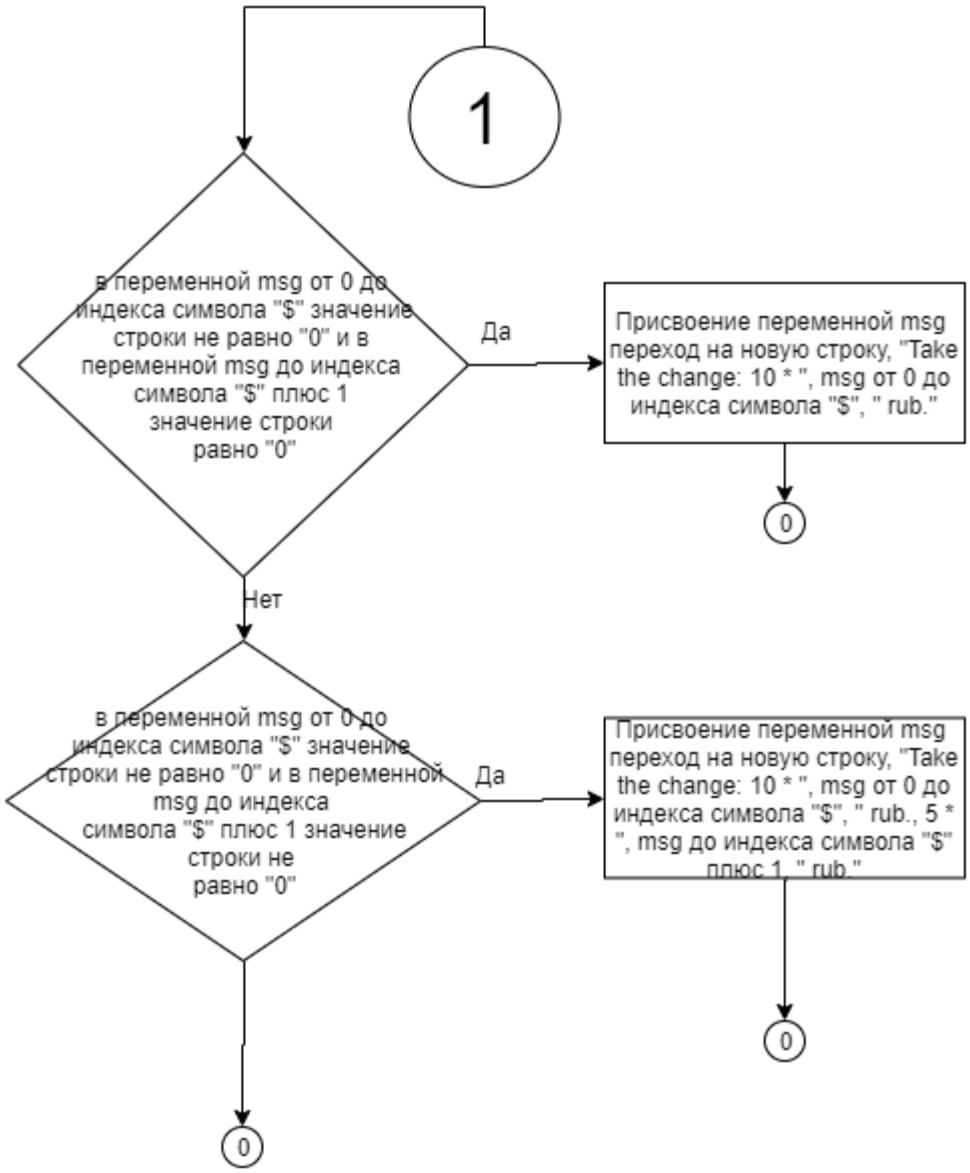

Рисунок 6 – Блок-схема алгоритма

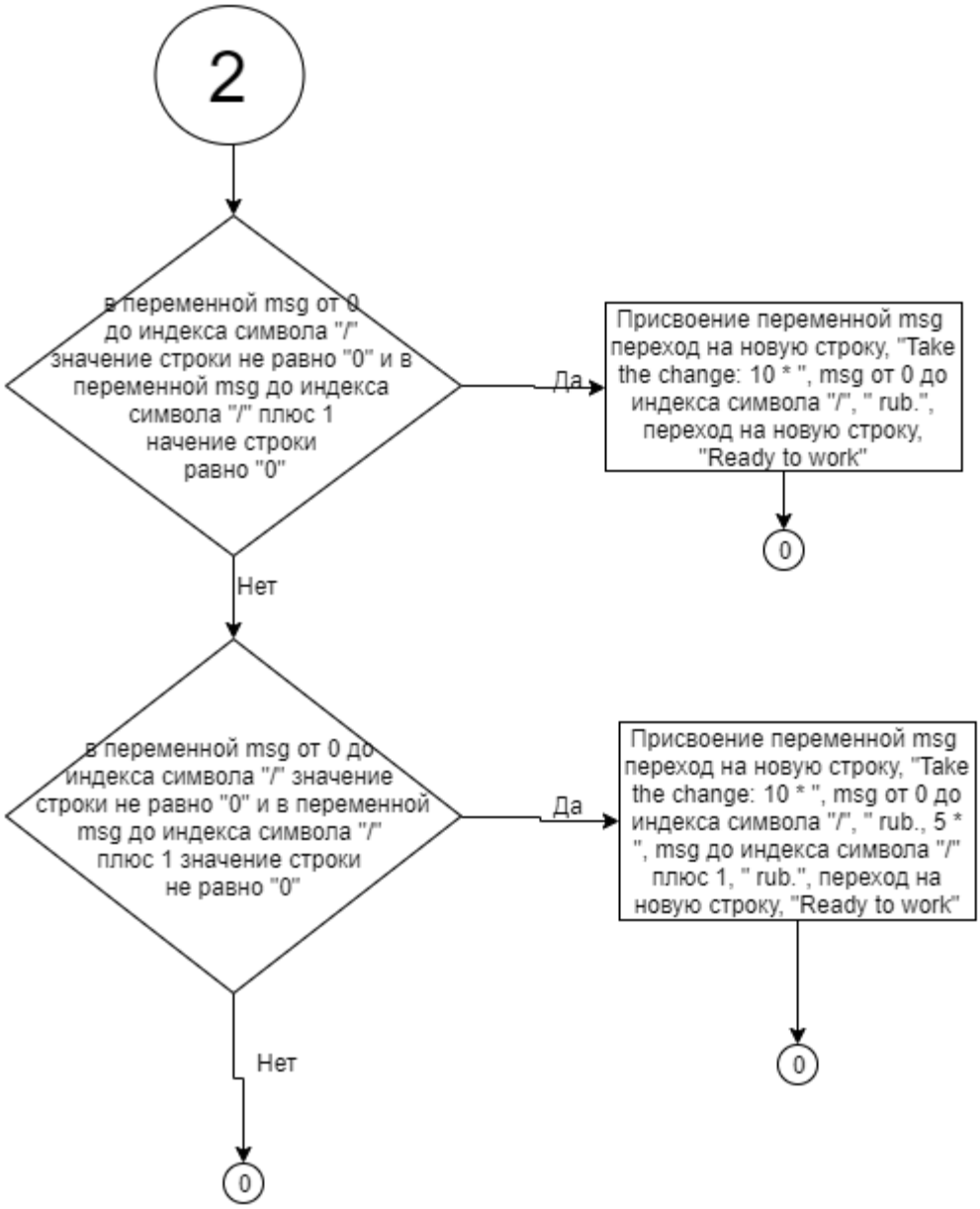

Рисунок 7 – Блок-схема алгоритма

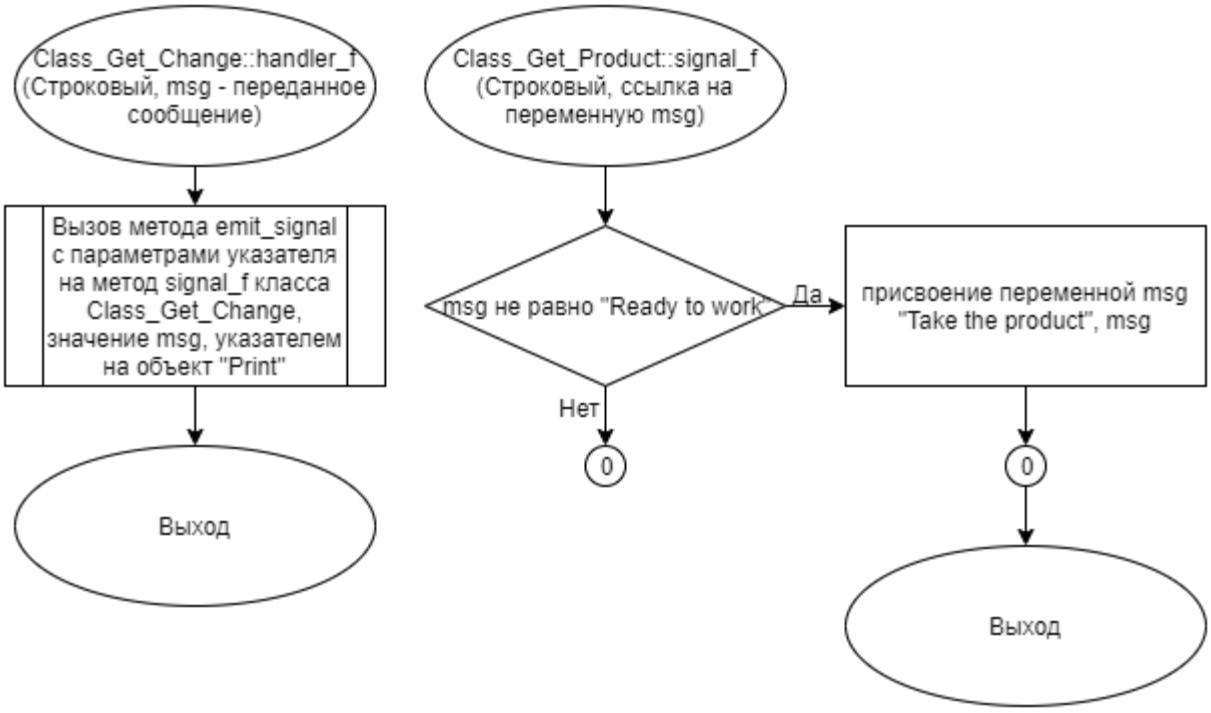

Рисунок 8 – Блок-схема алгоритма

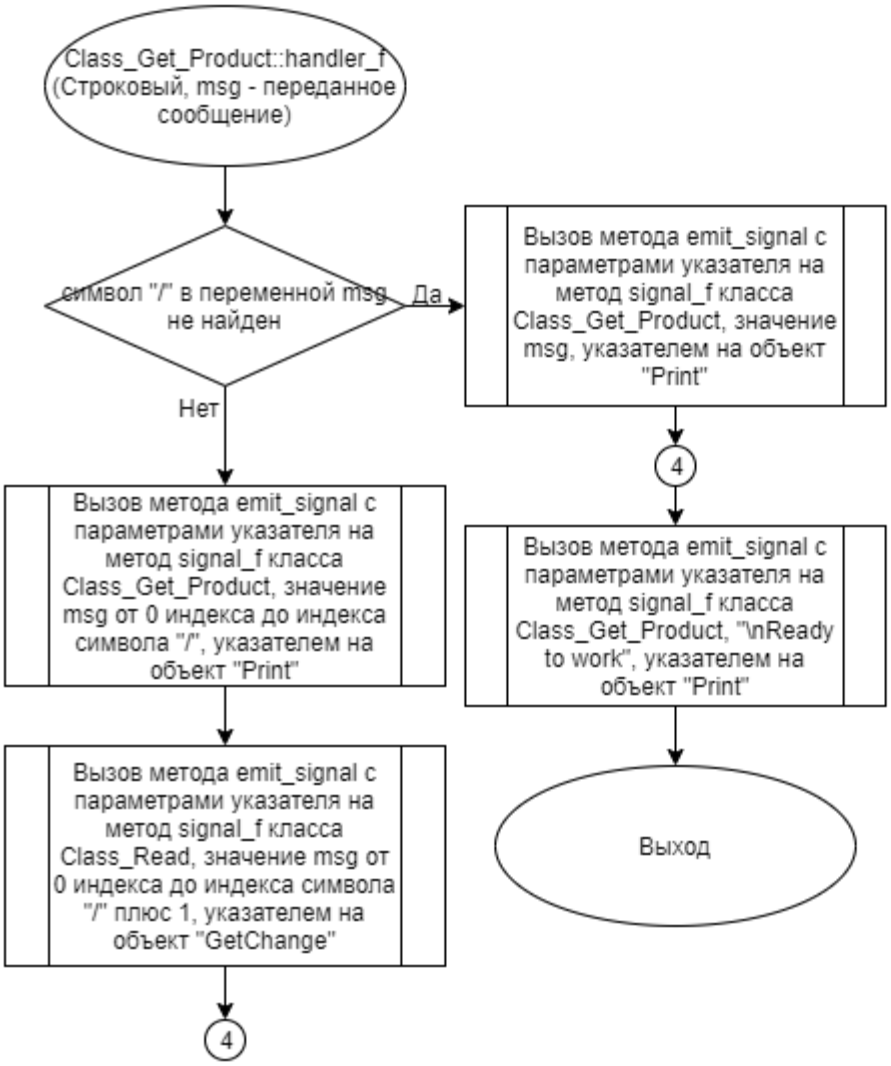

Рисунок 9 – Блок-схема алгоритма

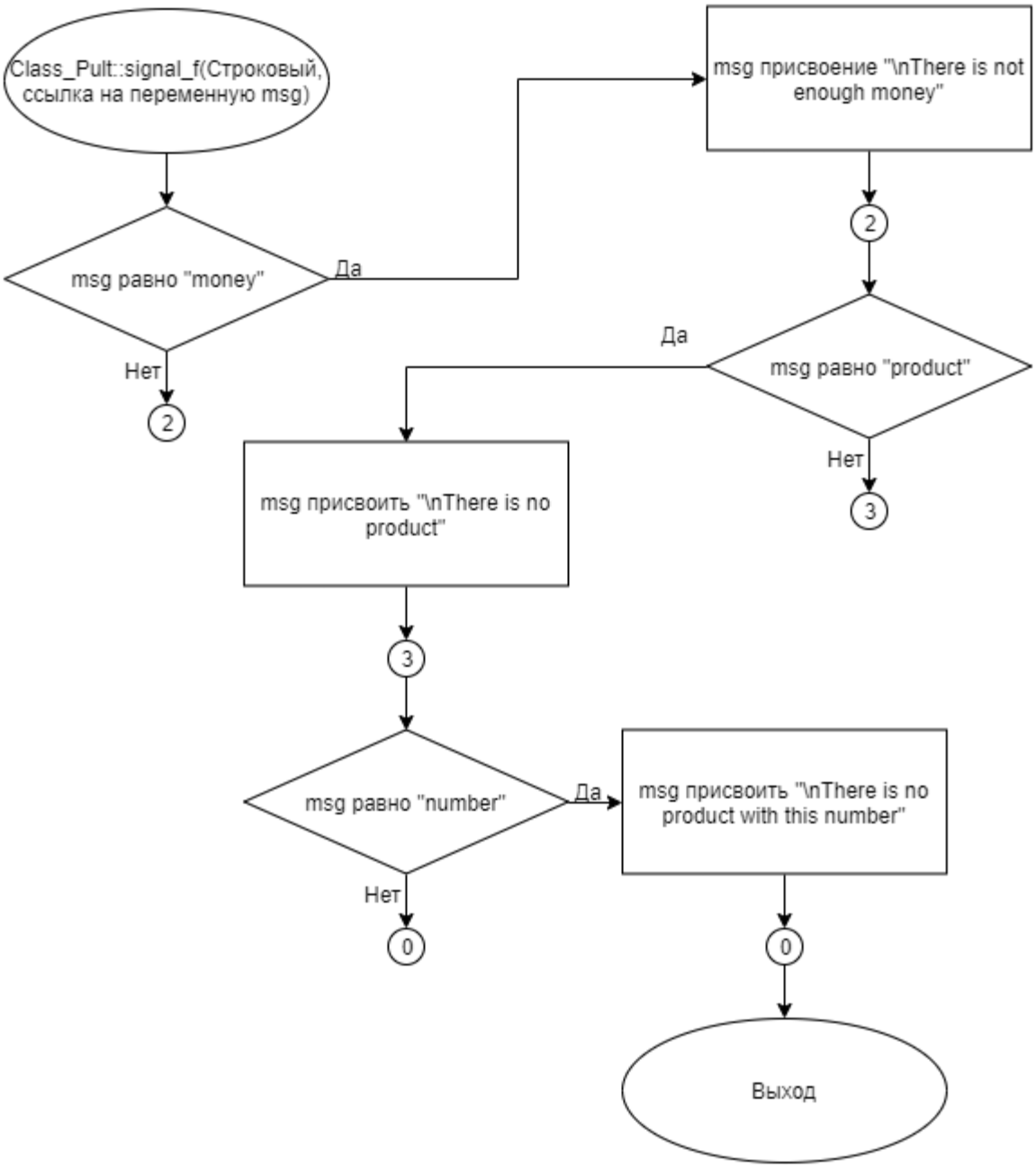

Рисунок 10 – Блок-схема алгоритма

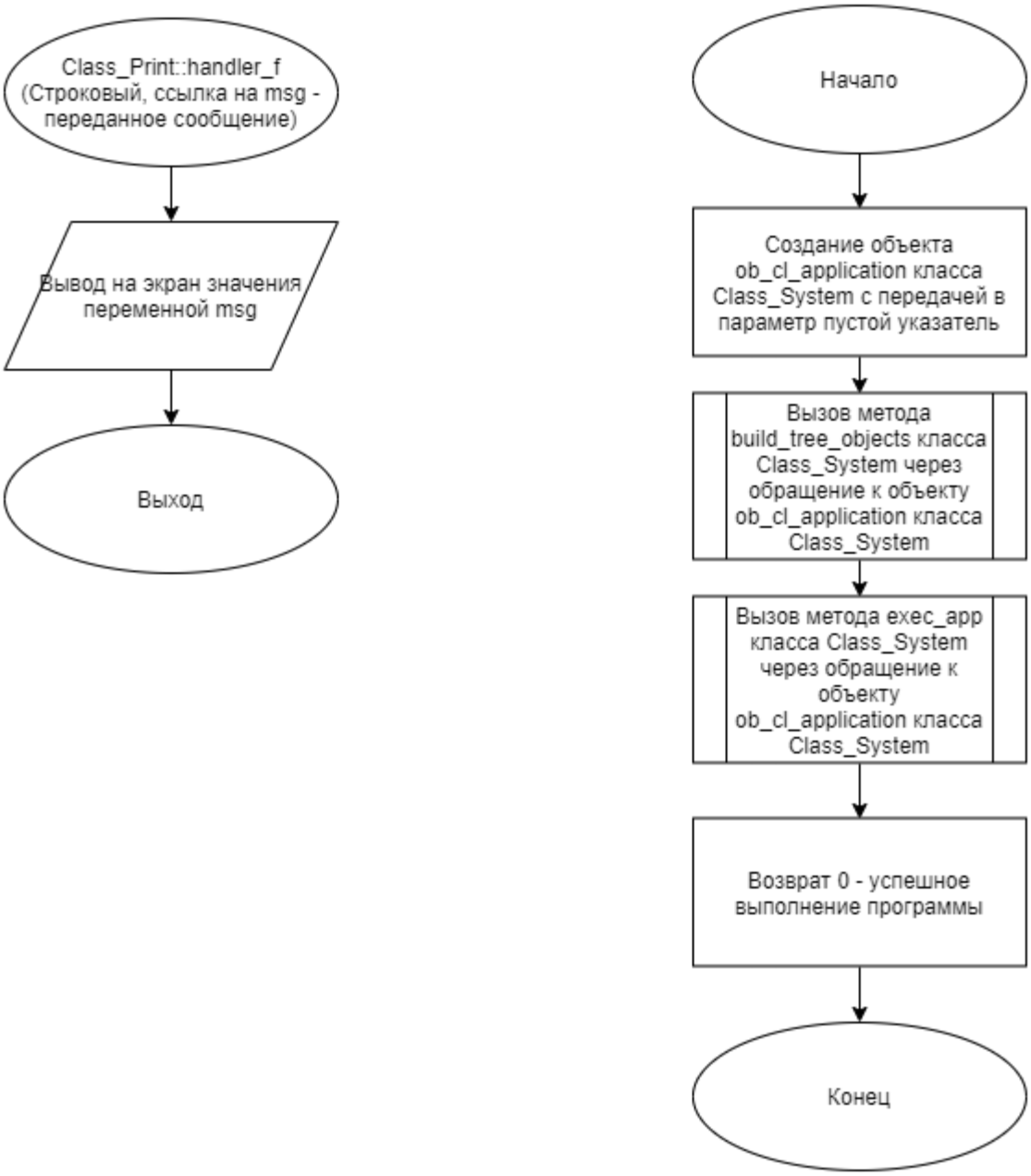

Рисунок 11 – Блок-схема алгоритма

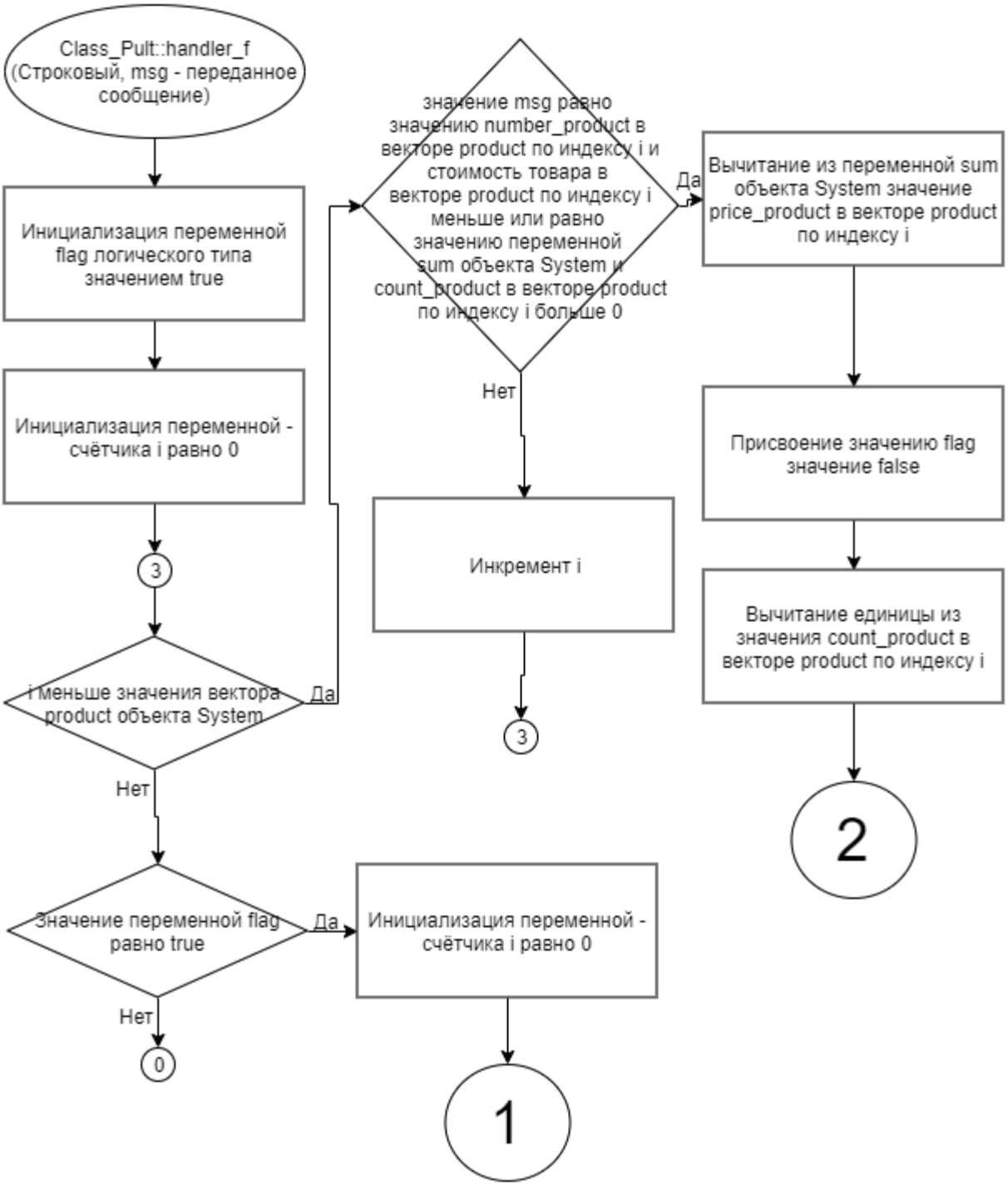

Рисунок 12 – Блок-схема алгоритма

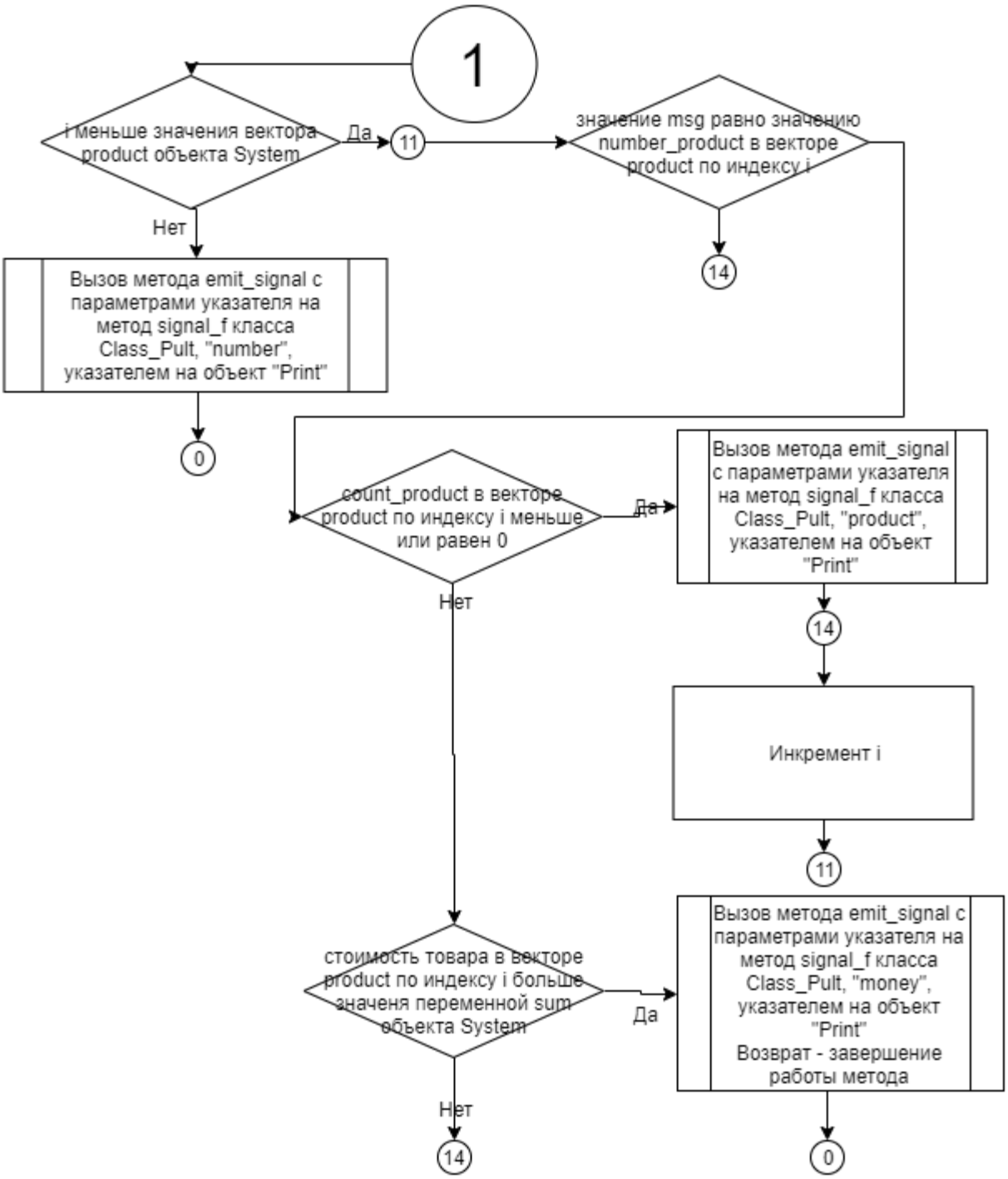

Рисунок 13 – Блок-схема алгоритма

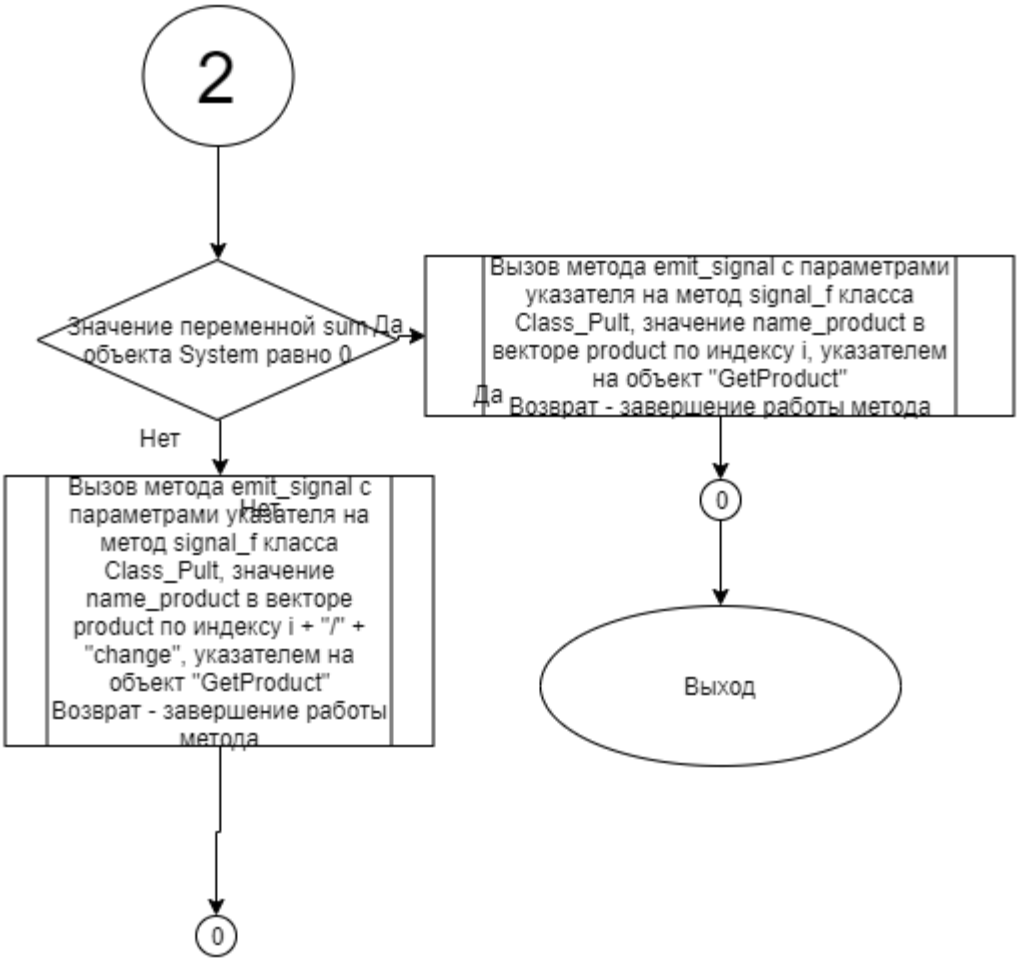

Рисунок 14 – Блок-схема алгоритма

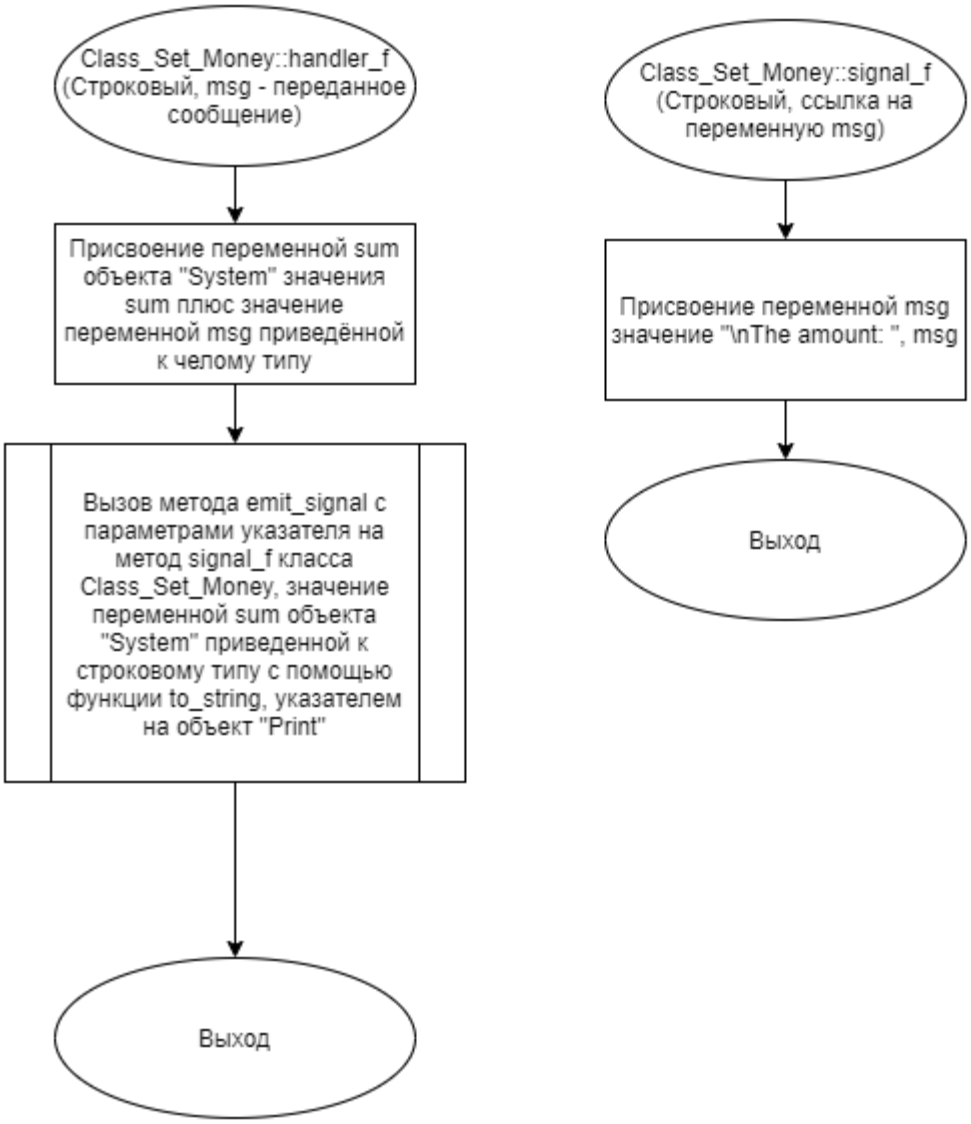

Рисунок 15 – Блок-схема алгоритма

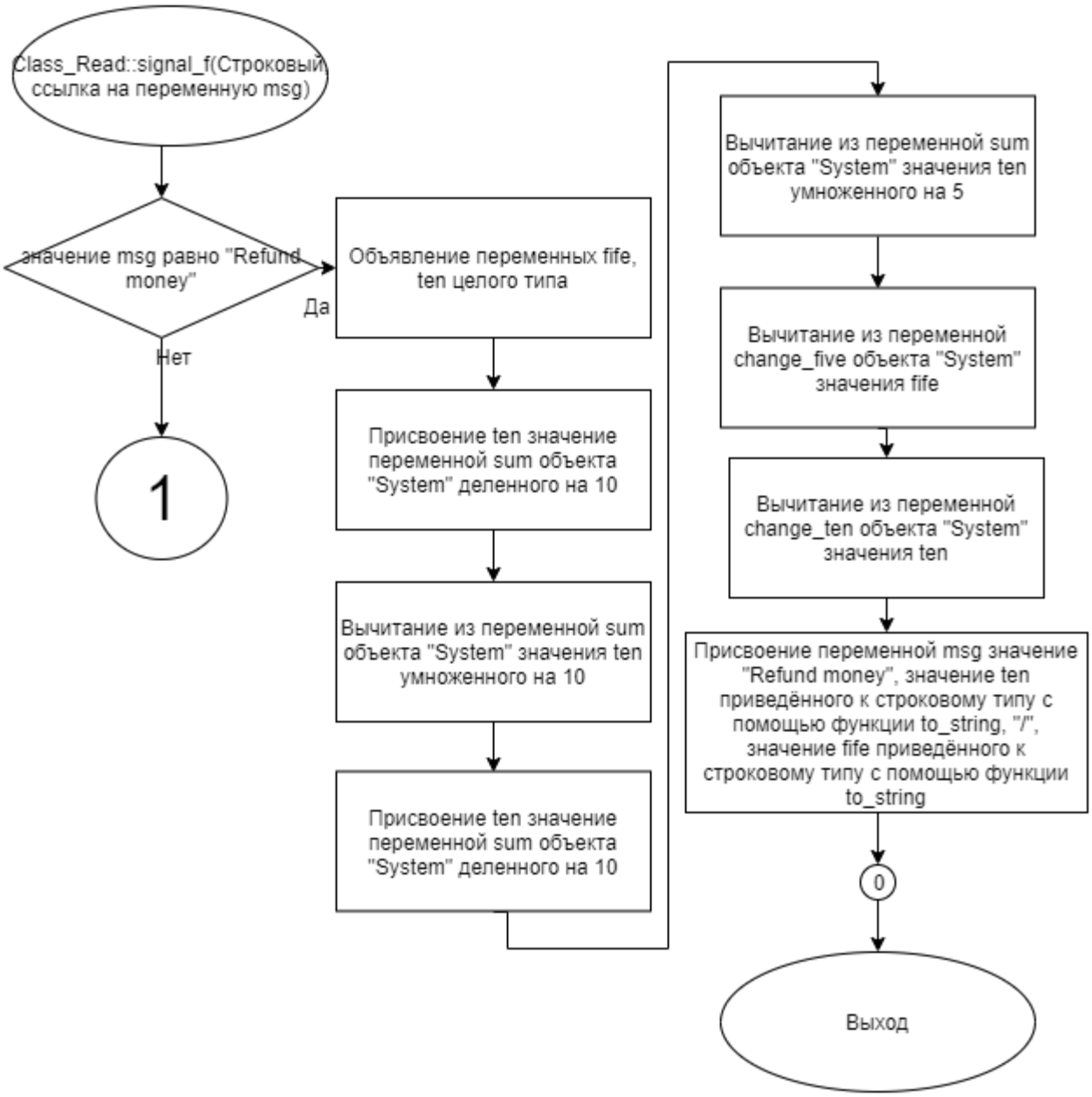

Рисунок 16 – Блок-схема алгоритма

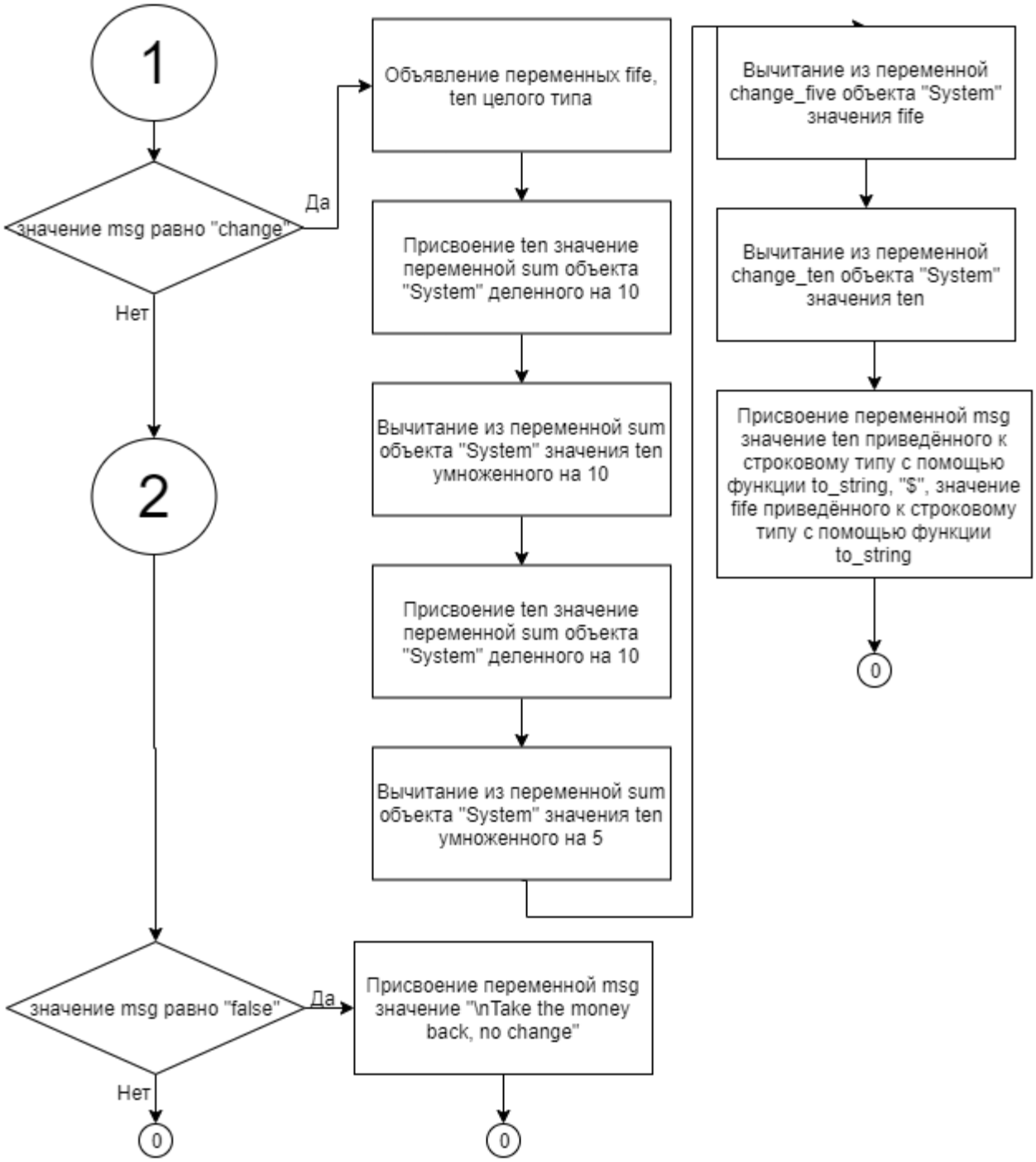

Рисунок 17 – Блок-схема алгоритма

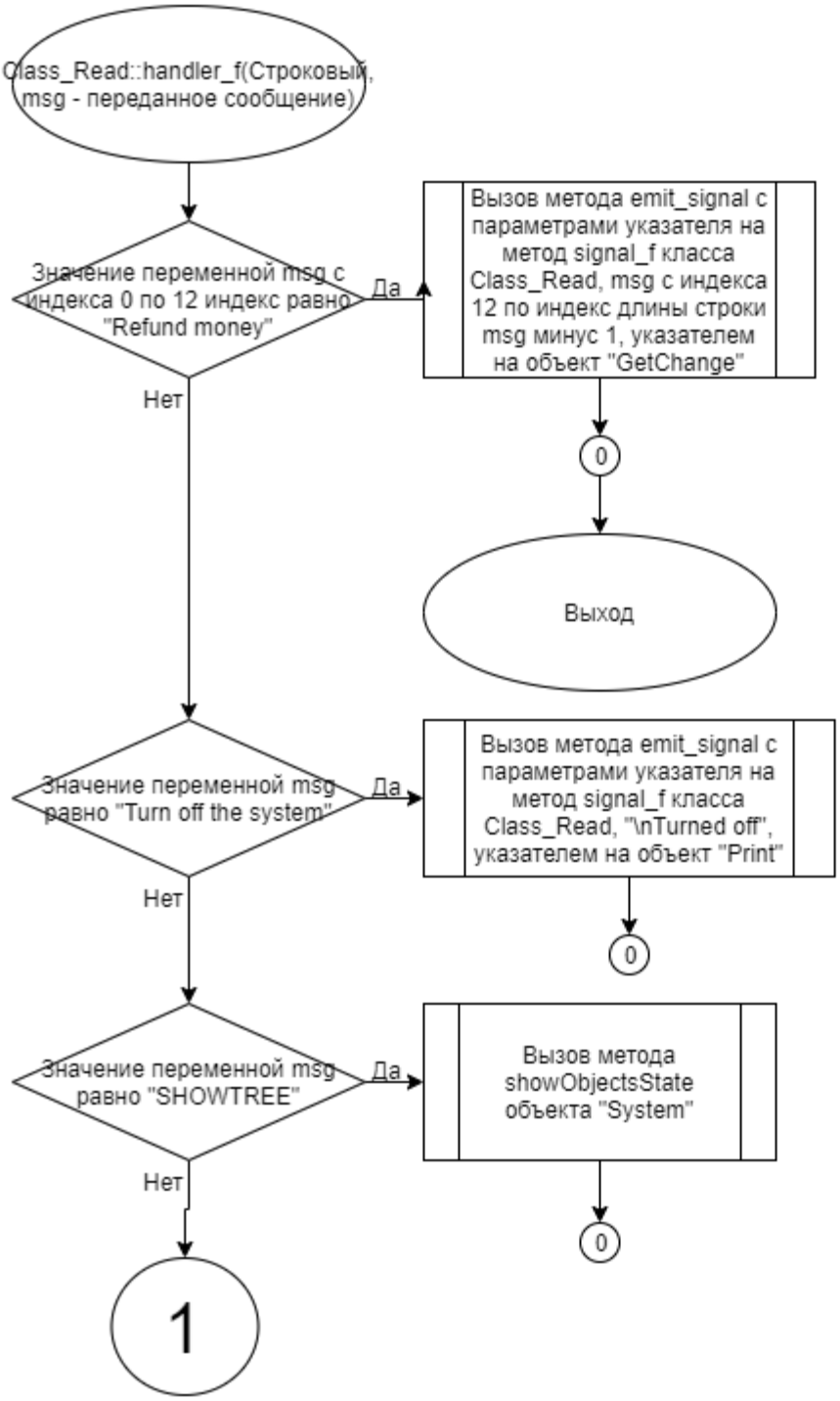

Рисунок 18 – Блок-схема алгоритма

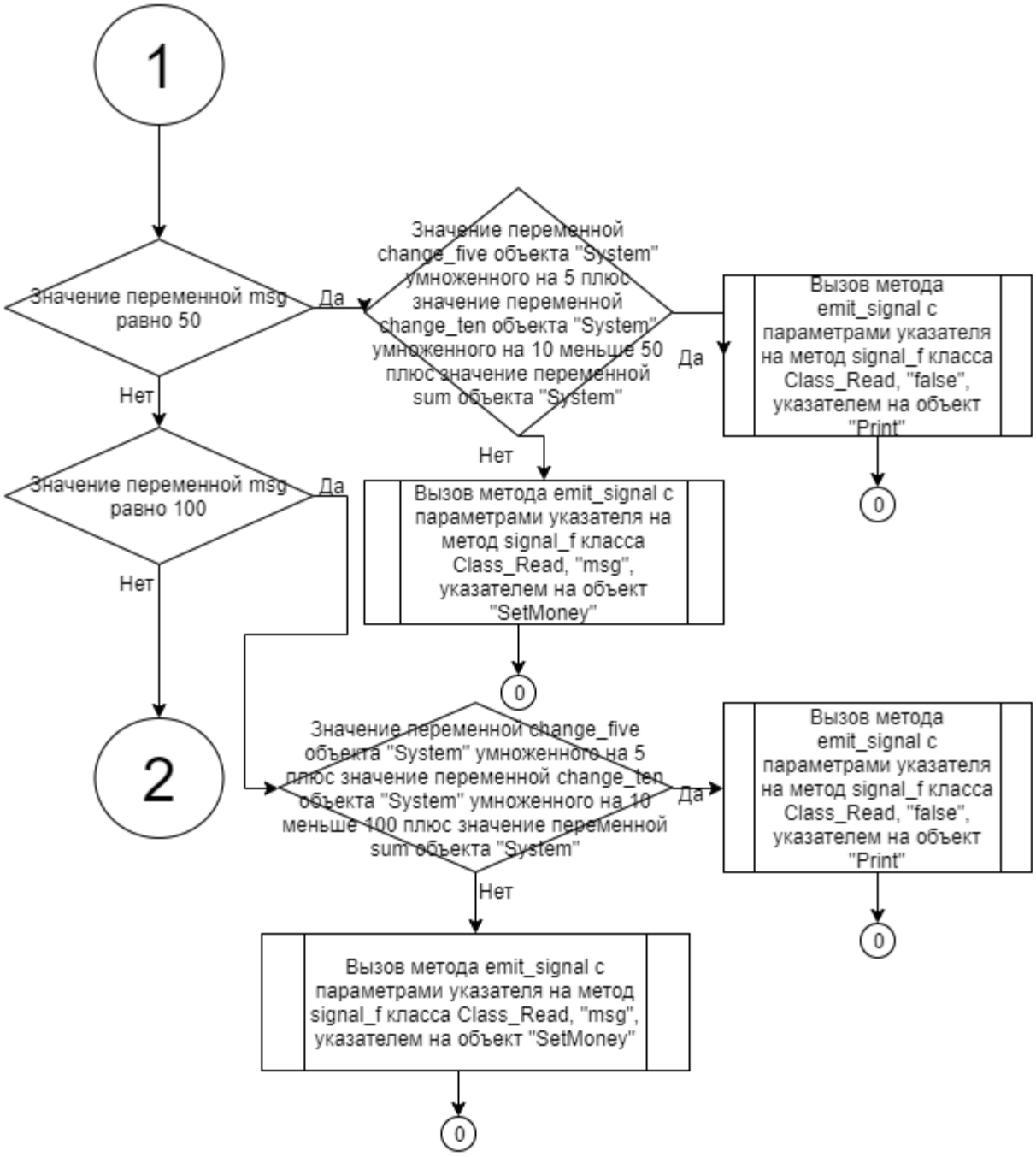

Рисунок 19 – Блок-схема алгоритма

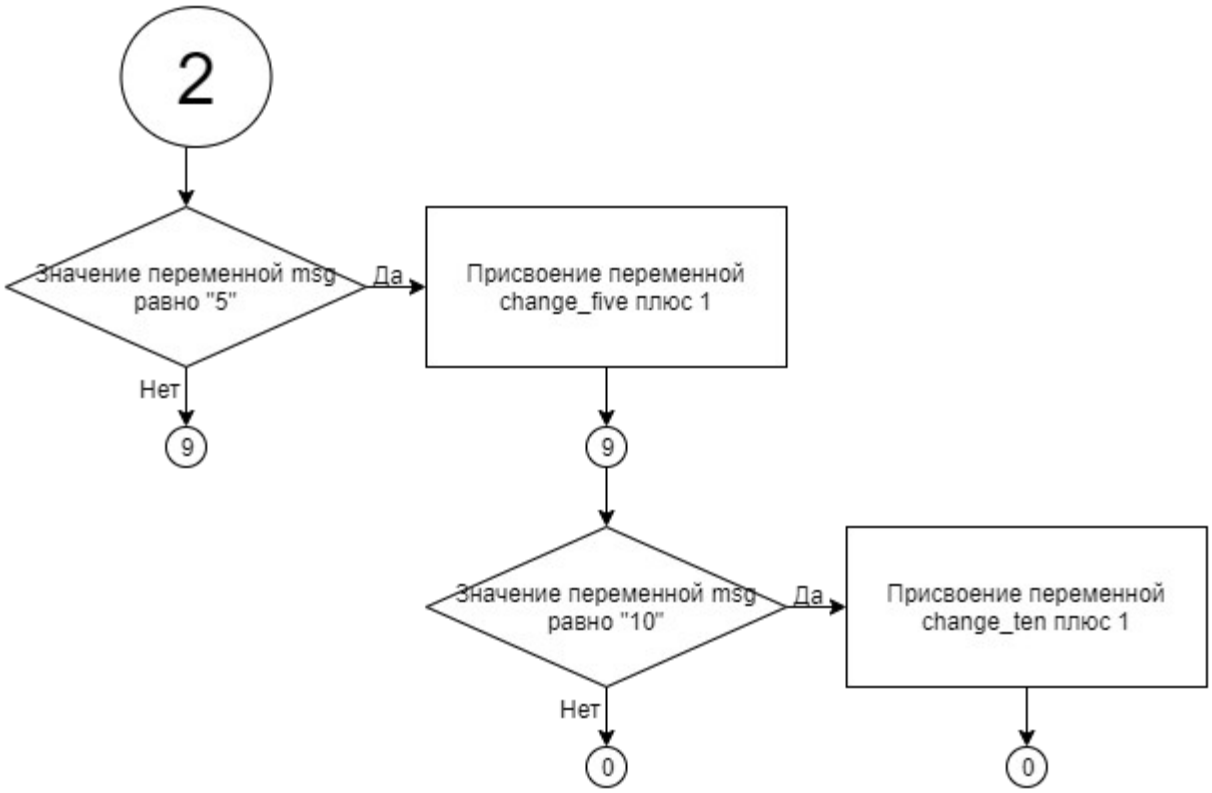

Рисунок 20 – Блок-схема алгоритма

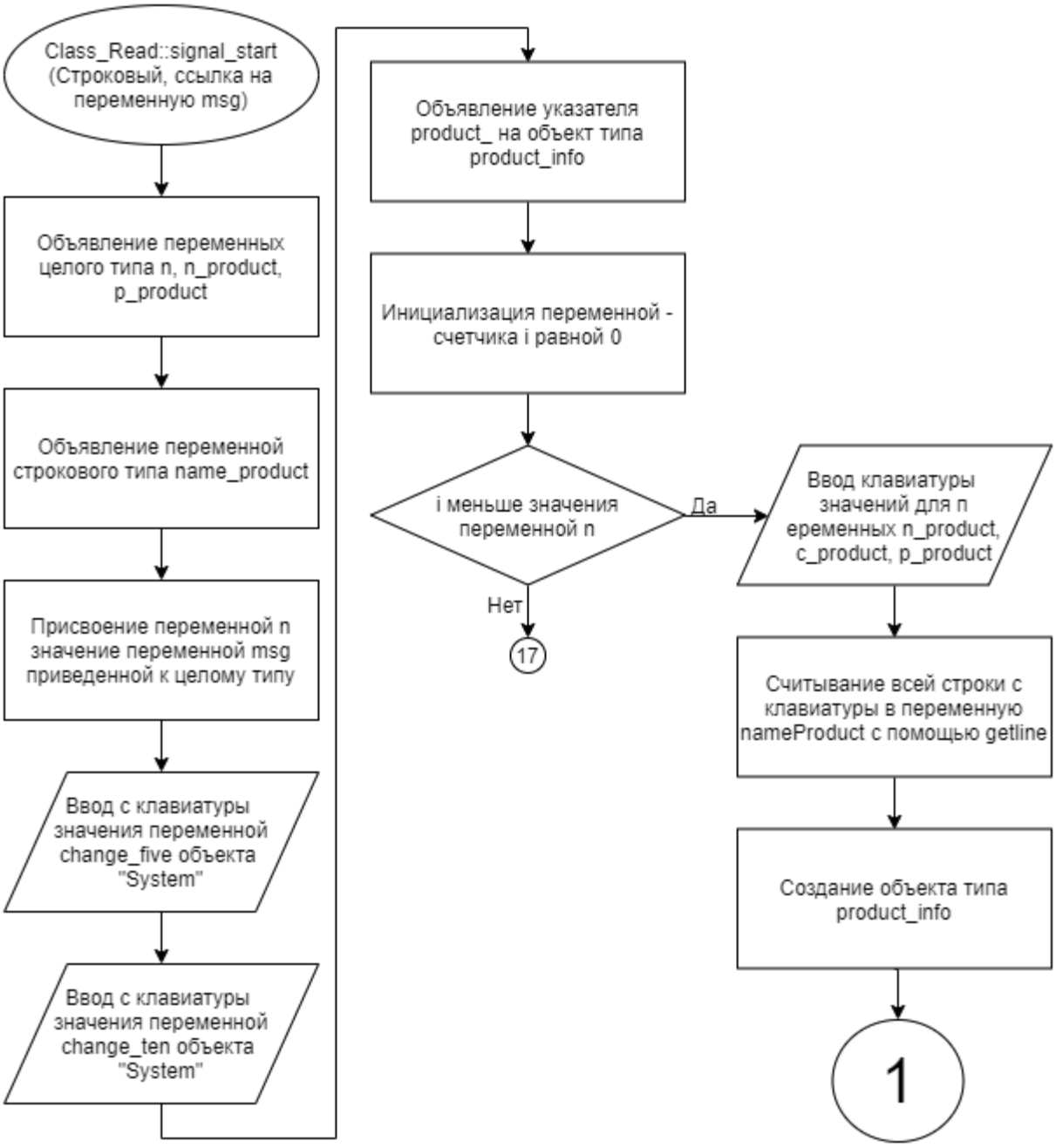

Рисунок 21 – Блок-схема алгоритма

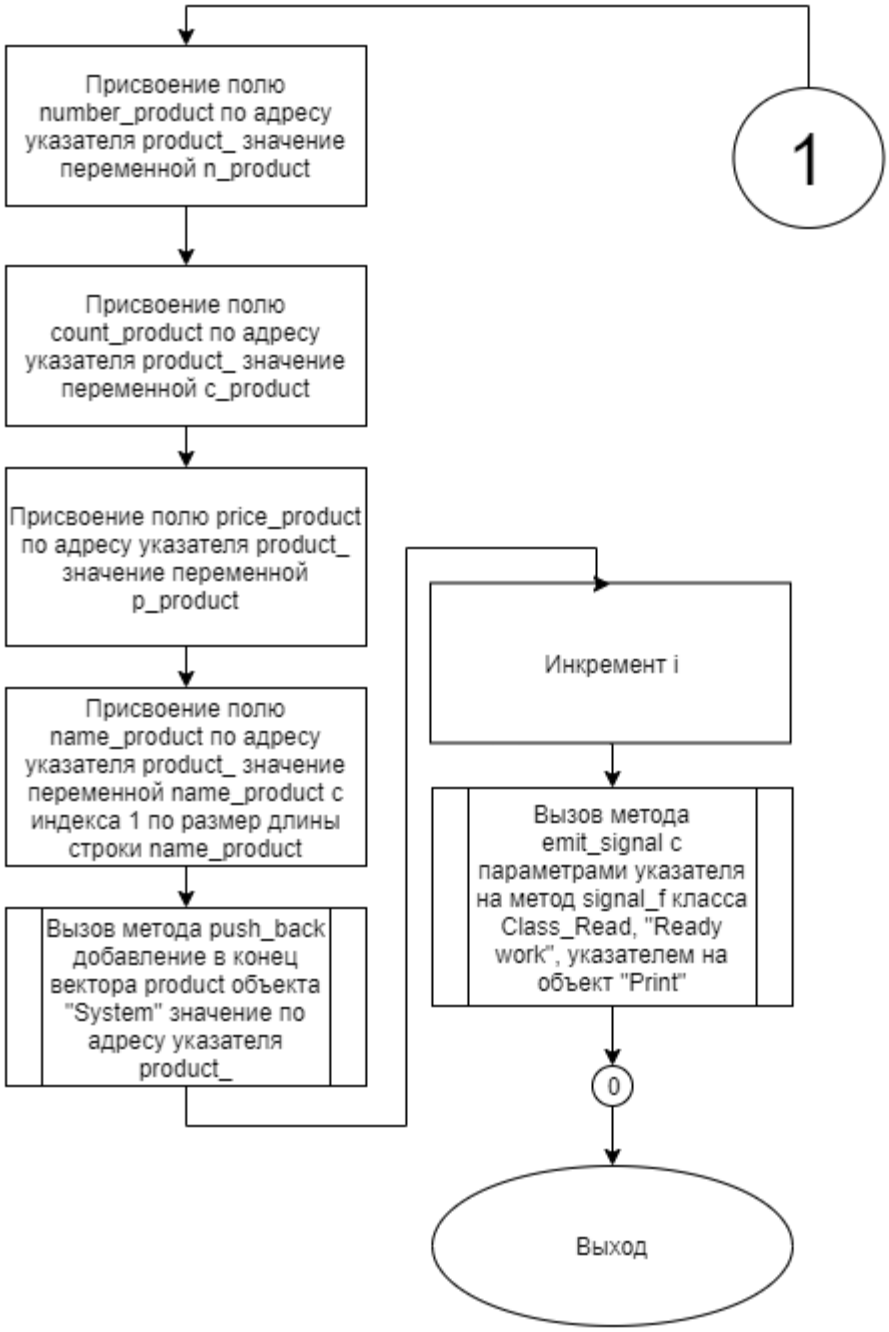

Рисунок 22 – Блок-схема алгоритма

5 КОД ПРОГРАММЫ

Программная реализация алгоритмов для решения задачи представлена ниже.

5.1 Файл Class_base.cpp

Листинг 1 – Class_base.cpp

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

Class_base :: Class_base(Class_base* b_head_object, string b_name) :
b_head_object(b_head_object), b_name(b_name) // присвоение свойствам класса
Class_base значения из параметра конструктора
{

   //-----------------------------------------------//
   // Конструктор класса Class_base создающий     //
   //        ветку подчинённых объектов           //
   //-----------------------------------------------//



   if(b_head_object != nullptr) // проверка на нулевой указатель
      b_head_object -> b_sub_objects.push_back(this); // добавляение в конец
вектора указателя на объект


}

Class_base*  Class_base  ::  base_s  =  new  Class_base();  //  переменная
указывающая на экземпляр класса

Class_base :: ~Class_base(){

   //-----------------------------------------------//
   // Деструктор класса Class_base удаляющий      //
   //        ветку подчинённых объектов           //
   //-----------------------------------------------//

   for(int  i  =  0;  i  <  b_sub_objects.size();  i++){  //  цикл  по  подчиненным
объектам
      delete b_sub_objects[i]; // удаление подчиненного объекта
   }
}
```

```
bool Class_base :: setName(string b_new_name){

   //-------------------------------------//
   // Установка нового имени из параметра //
   // если такое имя еще не существует    //
   //-------------------------------------//

   if (b_head_object != nullptr){ // проверка на не нулевую ссылку
      for(int i = 0; i < b_head_object -> b_sub_objects.size(); i++){ // по
указателю на головной объект вызываем его вектор с его подчиненными
объектами и функцию size
         if (b_head_object -> b_sub_objects[i] -> getName() == b_new_name) //
проверка на совпадение с именем
            return false; // вовзрат неудачной установки имени
      }
   }

   this -> b_name = b_new_name; // присвоение свойству b_name параметр
b_new_name
   return true; // возврат индикатора успешной установки имени
}

string Class_base :: getName(){

   //---------------------------------//
   // Метод возвращающий имя объекта //
   //---------------------------------//

   return this -> b_name; // возврат имени объекта
}

Class_base* Class_base :: getHead(){

   //---------------------------------//
   // Метод возвращающий указатель  //
   //      на головной объект       //
   //---------------------------------//

   return this -> b_head_object; // возврат указателя на головной объект
}

Class_base* Class_base :: getSubObject(string s_name){

   //-------------------------------------------------//
   // Метод поиска объекта по имени в подчинённых     //
   // объекта головного                               //
   //-------------------------------------------------//

   for(int i = 0; i < b_sub_objects.size(); i++) // проходим по всем
элементам массива
      if (b_sub_objects[i] -> getName() == s_name) // если имя объекта
совпало с тем, что мы ищем
         return b_sub_objects[i]; // возврат этого объекта

   for (int i = 0; i < b_sub_objects.size(); i++)
```

```
      if (b_sub_objects[i]->getSubObject(s_name)->getName() == s_name)
         return b_sub_objects[i]->getSubObject(s_name);


   return nullptr; // возврат нулевого указателя
}
/*
Class_base* Class_base :: search_Object_branch(string b_name_new){

   //---------------------------------------------//
   // Метод осуществляет поиск объекта по имени   //
   // от корня и возвращает указатель на          //
   // первое вхождение объекта с требуемым именем //
   // то есть поиск по дереву                     //
   //---------------------------------------------//

   int counter = 0; //счетчик для подсчёта количества объектов с именем
b_name_new
   Class_base* result = nullptr; // указатель на последний найденный объект с
заданным именем
   vector <Class_base*> vecto_r(1, this); // вектор для обхода, начиная с
текущего объекта

   while(vecto_r.size()>0){ // пока в векторе есть элементы

      Class_base* clas = vecto_r.back(); // получение последнего элемента из
вектора
      vecto_r.pop_back(); // удаление последнего элемента из вектора

      if (clas -> b_name == b_name_new){ // проверка, совпадает ли имя
текущего элемента с искомым
         counter++; // увеличение счетчика, если имя совпадает
         result = clas; // обновление result на текущий объект
      }

      for(int i = 0; i < clas -> b_sub_objects.size(); i++){ // перебор всех
подобъектов текущего объекта
         vecto_r.push_back(clas    ->    b_sub_objects[i]);    //    добавление
подобъектов в вектор для последующего обхода
      }
   }

   if (counter <= 1) // если объект с искомым именем встретился не более
одного раза
      return result; // возвращение найденного объекта

   return nullptr; // возвращение nullptr, если искомое имя встретилось более
одного раза

}
*/
/*
Class_base* Class_base :: search_object_sub_by_name(string b_name_new){

   //---------------------------------------------//
```

```
   // Метод осуществляет поиск объекта по имени   //
   // от текущего и возвращает указатель на       //
   // первое вхождение объекта с требуемым именем //
   // то есть поиск по ветке                      //
   //-------------------------------------------//

   if (b_head_object != nullptr) // проверка на существование объекта
      return  b_head_object  ->  search_object_sub_by_name(b_name_new);  //
рекурсивный вызов метода

   else // иначе
      return search_Object_branch(b_name_new); // вызов метода для поиска по
ветке
}
*/

void Class_base :: setObjectState(int state){

   //---------------------------//
   // Метод установки состояния //
   // объекта                   //
   //---------------------------//

   if(getHead() && !getHead() -> b_state || state == 0) // если есть головной
файл и он не имеет состояния или его состояние равно 0
      this -> b_state = false; // присвоение состояние not ready
   else // иначе
      this -> b_state = true; // присвоение состояние ready

   if (!state) // если состояние falst то есть notReady
      for (auto b_sub_object : b_sub_objects) // цикл по подчиненным объектам
         b_sub_object  ->  setObjectState(state);  //  все  подчиненные  объекты
получают состояние false то есть not ready

}

/*
void Class_base :: showObjectAndSubObjects(int spaces){

   //---------------------------------//
   // Метод вывода иерархии объектов //
   // (дерева или ветки) от текущего //
   // объекта                        //
   //---------------------------------//

   cout << this -> getName(); // вывод текущего имени объекта

   if (!b_sub_objects.empty()){ // если массив с подчиненными объектами не
пустой

      for  (  auto  b_sub_object  :  b_sub_objects){  //  цикл  по  подчиненным
объектам
         cout << endl; // переход на новую строку

         for(int i = 0; i < spaces; i++) // цикл пробелов
```

```
            cout << " "; // вывод пробела на экран

         b_sub_object -> showObjectAndSubObjects(spaces + 4); // вызов метода
showObjectAndSubObjects с параметром пробелов + 4 по ссылке
      }
   }
}
*/
void Class_base :: showObjectsState(int spaces){

   //-----------------------------------------------------//
   // Метод вывода иерархии объектов (дерева или ветки) //
   // и отметок их готовности от текущего объекта       //
   //-----------------------------------------------------//

   cout << this -> getName(); // вывод текущего имени объекта
   cout << (this -> b_state ? " is ready" : " is not ready"); // если
состояние объекта true - вывод is ready, если false - вывод is not ready

   if (!b_sub_objects.empty()){ // если массив с подчиненными объектами не
пустой

      for ( auto b_sub_object : b_sub_objects){ // цикл по подчиненным
объектам
         cout << endl; // переход на новую строку

         for(int i = 0; i < spaces; i++) // цикл попробелам
            cout << " "; // вывод пробела на экран

         b_sub_object -> showObjectsState(spaces + 4); // вызов метода
showObjectsState с параметром пробелов + 4 по ссылке
      }
   }
}

/*
bool Class_base :: changeHead(Class_base* new_b_head_object)
{

   //-----------------------------------------------------------------------
----//
   // Метод переопределения головного объекта для текущего в дереве иерархии.
//
   // Метод должен иметь один параметр, указатель на объект базового класса,
//
   // содержащий указатель на новый головной объект. Переопределение
головного   //
   // объект для корневого объекта недопустимо. Недопустимо создать второй
//
   // корневой объект. Недопустимо при переопределении, чтобы у нового
//
   // головного появились два подчиненных объекта с одинаковым наименованием.
//
   // Новый головной объект не должен принадлежать к объектам из ветки
текущего. //
   // Если переопределение выполнено, метод возвращает значение «истина»,
```

```
//
   //                                иначе                                «ложь»
//
   //-------------------------------------------------------------------------
----//


   if(new_b_head_object){ // проверка на существование объекта

      Class_base* test_object = new_b_head_object; // указатель для проверки
циклических ссылок в дереве

      while (test_object) // пока не станет nullptr
      {
         test_object  =  test_object  ->  getHead();  //  обновляется,  чтобы
указывать на родительский объект выше по иерархической лестнице
         if  (this  ==  test_object)  //  проверка  на  то,  не  стал  ли  текущий
объект равен test_object
            return false; // возврат
      }

      if (!new_b_head_object -> getSubObject(getName()) && b_head_object) //
проверка на уникальность имени и существует ли уже такой объект
      {
         b_head_object     ->     b_sub_objects.erase(find(b_head_object     ->
b_sub_objects.begin(),    b_head_object->b_sub_objects.end(),    this));    //
удаление указателя на текущий объект из подобъектов текущего родительского
объекта

         new_b_head_object   ->   b_sub_objects.push_back(this);   //   текущий
объект добавляется в список подобъектов нового родительского объекта
         b_head_object  =  new_b_head_object;  //  обновляется  указатель  на
текущий родительский объект

         return true; // возврат
      }
   }

   return false; // возврат
}

void Class_base :: deleteObjectName(string delete_b_name){

   //-------------------------------------------------------//
   // Метод удаления подчиненного объекта по наименованию. //
   // Если объект не найден, то метод завершает работу.    //
   // Один параметр строкового типа, содержит наименование //
   // удаляемого подчиненного объекта                      //
   //-------------------------------------------------------//

   if (delete_b_name.empty()) // проверка на не пустую строчку
      return; // завершение метода

   for(  int  i  =  0;  i  <  b_sub_objects.size();  i++)  //  цикл  по  подчиненным
объектам
      if  (b_sub_objects[i]  ->  getName()  ==  delete_b_name)  //  проверка  что
```

```
объект по индексу совпадает с удаляемым объектом
        b_sub_objects.erase(b_sub_objects.begin()   +   i);   //   удаление
подчиненных объектов объекта delete_b_name

}

Class_base* Class_base :: getObjectCoordinate(string b_name_coordinate)
{

   //-------------------------------------------------------------------------
-------------------------------------------------------------------------
--//
   // Метод получения указателя на любой объект в составе дерева иерархии
объектов              согласно              пути              (координаты).
//
   //  В  качестве  параметра  методу  передать  путь  (координату)  объекта.
Координата          задаться          в          следующем          виде:
//
   //             /             -             корневой             объект;
//
   // //«имя объекта» - поиск объекта по уникальной имени от корневого (для
однозначности     уникальность     требуется     в     рамках     дерева);
//
   //             .             -             текущий             объект;
/
   // .«имя объекта» - поиск объекта по уникальной имени от текущего (для
однозначности уникальность требуется в рамках ветви дерева от текущего
объекта); //
   // «имя объекта 1»[/«имя объекта 2»] . . . - относительная координата от
текущего    объекта,    «имя    объекта    1»    подчиненный    текущего;
//
   // /«имя объекта 1»[/«имя объекта 2»]  . . . - абсолютная координата от
корневого                                                          объекта.
//
   //-------------------------------------------------------------------------
-------------------------------------------------------------------------
--//

   string b_name_this; // объявление строки для хранения текущего названия
объекта или пути
   int index = 0; // инициализация переменной индекса, для навигации по
строке
   Class_base* b_root = this; // инициализация текущим объектом

   if  (b_name_coordinate[0]  ==  '/')  //  проверка  начинается  ли  строка
координат с символа /
   {
      while (b_root -> getHead()) // цикл для продвижения к корню дерева
объектов
        b_root = b_root -> getHead(); // присвоение указателю новый головной
объект
   }

   if (b_name_coordinate.length() == 1 && (b_name_coordinate[0] == '/' ||
b_name_coordinate[0] == '.')) // если строка состоит из одного символа и
```

```
( начинается с / или .)
      return b_root; // возврат корневого объекта

   if (b_name_coordinate[0] == '/' || b_name_coordinate[0] == '.') // если
строка начинается с / или .
      index = 1; // присвоить переменной индекс, чтобы начать со следующего
символа

   if (b_name_coordinate[0] == '/' && b_name_coordinate[1] == '/') // если
строка начинается с //
   {
      b_name_this  =  b_name_coordinate.substr(2);  //  получаем  подстроку
после //

      return  b_root->search_Object_branch(  b_name_this);  //  возвращаем
указатель на найденный объект в дереве по этому имени
   }

   while (index < b_name_coordinate.length()) // проходим по всем символам в
строке
   {
      if (b_name_coordinate[index] == '/') // если текущий символ /
      {
         b_root  =  b_root  ->  getSubObject(b_name_this);  //  присваиваем
указатель на найденный объект в списке подчиненных объектов

         if (!b_root) // если подобъект не был найден
            return nullptr; // возврат нулевого указателя

         b_name_this = ""; // очищаем имя для следующего возможного имени
объекта
      }

      else // иначе
         b_name_this += b_name_coordinate[index]; // накапливаем имя

      index++; // увеличиваем индекс
   }

   if (b_name_coordinate[0] == '.') // если путь начинается с .
      return  b_root  ->  search_Object_branch(b_name_this);  //  возвращаем
указатель на найденный объект на дереве

   else // иначе
      return b_root -> getSubObject(b_name_this); // возвращаем указатель на
найденный объект в списке подчиненных объектов
}
*/
void  Class_base  ::  set_connection(TYPE_SIGNAL  b_signal,  Class_base*
b_target, TYPE_HANDLER b_handler){

   //-------------------------------------------------------------------------
---------------//
   // Метод установки связи между сигналом текущего объекта и обработчиком
целевого объекта //
   //-------------------------------------------------------------------------
```

```
---------------//

  object_sh* b_value; // объявление указателя на объект типа object_sh, который будет использоваться для хранения деталей подключения (соединения).

  for (int i = 0; i < connects.size(); i++) // цикл по всем элементам вектора connects
     if (connects[i] -> b_signal == b_signal && connects[i] -> b_target == b_target && connects[i] -> b_handler == b_handler) // проверка на существует ли уже соединение с такими же парамтерамми b_signal, b_target и b_handler
        return; // если такое соединение найдено, завершаем выполнение метода

  b_value = new object_sh(); // выделяем память для нового объекта object_sh и присваиваем его адрес переменной b_value

  // Присваиваем соответствующие значения членам структуры нового объекта
  b_value -> b_signal = b_signal; // присваиваем значение b_signal члену b_signal объекта b_value
  b_value -> b_target = b_target; // присваиваем значение b_target члену b_target объекта b_value
  b_value -> b_handler = b_handler; // присваиваем значение b_handler члену b_handler объекта b_value

  connects.push_back(b_value); // добавляем указатель на объект b_value в конец вектора connects
}

void Class_base :: delete_connction(TYPE_SIGNAL b_signal, Class_base* b_target, TYPE_HANDLER b_handler){

  //-------------------------------------------------------------------------
------------------------//
  // Метод удаления (разрыва) связи между сигналом текущего объекта и обработчиком целевого объекта //
  //-------------------------------------------------------------------------
------------------------//

  bool found = false; // инициализация флага найденного элемента как "ложь"
  int index = 0; // инициализация переменной index, которая будет хранить индекс элемента при прохождении по вектору

  for (auto connection : connects) // цикл for, проходящий по каждому элементу вектора connects
     if (connection -> b_signal == b_signal && connection->b_target == b_target && connection->b_handler == b_handler){ // проверка условий соответствия текущего объекта connection искомым значениям b_signal, b_target и b_handler
        found = true; // установка флага в true (найдено)
        break; // завершение работы цикла
     }
     else // иначе
        index++; // если текущий элемент не соответствует условию, увеличиваем индекс для следующих проверок

  if (found) // после завершения цикла проверяем, был ли найден искомый
```

```
элемент
     connects.erase(connects.begin() + index); // удаляем его, используя
метод erase вектора начиная от смещения начала вектора на index позиций
}


void Class_base :: emit_signal (TYPE_SIGNAL b_signal, string message,
Class_base* p_object){ // заголовок метода

   //-------------------------------------------------------------------------
---//
   // Метод выдачи сигнала от текущего объекта с передачей строковой
переменной //
   //-------------------------------------------------------------------------
---//

   (this->*b_signal)(message); // вызов метода сигнала с сообщением
   for (int i = 0; i < connects.size(); i++) { // перебор всех элементов
подключения
     if((connects[i]->b_signal==b_signal       )       &&       (connects[i]-
>b_target==p_object)){ // найдено подключение
        TYPE_HANDLER  hendl  =  connects[i]->b_handler;  //  инициализация
обработчика

        if(connects[i]->b_target->b_state != 0) // проверка состояния
          (connects[i]->b_target->*hendl)(message);    //    Вызов    метода
обработчика
     }
   }
}
/*
string Class_base :: getPath(){

   //-----------------------------------------------------------//
   // Метод определения абсолютной пути до текущего объекта //
   //-----------------------------------------------------------//

   string  path  =  "";  //  инициализация  пустой  строки  для  хранения
результирующего пути
   Class_base*  current  =  this;  //  инициализация  указателя  текущим
объектом(начало обхода для цикла)

   while(current -> getHead()){ // пока у текущего объекта есть родительский
элемент (поднимаемся к корню)
     path = "/" + current -> getName() + path; // формирование пути путем
добавления имени текущего объекта и предыдущего пути, разделенных слешем
     current = current -> getHead(); // перемещение указателя к голове
(родителю) текущего объекта, продвигаясь вверх по иерархии
   }

   if (path.empty()) // проверка на заполнение строки
     return "/"; // возврат /, то есть у объекта нет родителей и он это
корень

   return path; // возврат сформированного пути к текущему объекту от корня
```

```
}
*/

void Class_base :: setStateAll(int s){ // заголовок метода
   setObjectState(s); // вызов метода установки значения
   for(int i = 0; i < b_sub_objects.size(); i++) // перебор всех элементов
вектора
      b_sub_objects[i] -> setStateAll(s); // рекурсивный вызов
}
```

5.2 Файл Class_base.h

Листинг 2 – Class_base.h

```
#ifndef __CLASS_BASE__H
#define __CLASS_BASE__H
#include <iostream>
#include <vector>
using namespace std;

class Class_base; // объявление без определения, то есть у нас есть такой
класс, а определим его где - то ниже

#define SIGNAL_D( signal_f ) ( TYPE_SIGNAL ) ( & signal_f ) // для получения
указателя на метод сигнала объекта

#define HANDLER_D( hendler_f ) ( TYPE_HANDLER ) ( & hendler_f ) // для
получения указателя на метод обработчика объекта

typedef void (Class_base :: *TYPE_SIGNAL) (string & msg); // задаём новый
тип - указатель на сигнал

typedef void (Class_base :: *TYPE_HANDLER) (string msg); // задаём новый тип
- указатель на обработчик

struct object_sh{
   TYPE_SIGNAL b_signal; // указатель на метод класса Class_base для сигнала
   Class_base* b_target; // указатель на целевой объект
   TYPE_HANDLER b_handler; // указатель на метод класса Class_base для
обработки сообщения
};

struct product_info{ // характеристики продукта
   int number_product; // номер
   int count_product; // количество
   int price_product; // цена
   string name_product; // название
};
```

```
// базовый класс
class Class_base{
   private:
      string b_name;
      Class_base* b_head_object; // указатель на головной объект для каждого
объекта
      vector <Class_base*> b_sub_objects; // динамический массив хранящий
подчиненные объекты головного объекта
      int b_state; // состояние объекта
      vector <object_sh*> connects; // массив для хранения связей

   public:

      Class_base(Class_base* b_head_object = base_s, string b_name = "Base");
// конструктор
      ~Class_base(); // деструктор
      bool setName(string b_new_name); // установка имени из аргумента
      string getName(); // возврат свойства имя
      Class_base* getHead(); // возврат указателя на головной объекта
      Class_base* getSubObject(string s_name);

      // Class_base* search_object_sub_by_name(string b_name);
      // Class_base* search_Object_branch(string b_name); // поиск объекта с
заданным именем на дереве
      void setObjectState(int state); // установка готовности объекта, в
качестве параметра передается переменная целого типа, содержит номер
состояния.
      // void showObjectAndSubObjects(int = 4); // метод вывода иерархии
объектов (дерева или ветки) от текущего объекта с параметром пробелов
заданным по умолчанию
      void showObjectsState(int = 4); // метод вывода иерархии объектов
(дерева или ветки) и отметок их готовности от текущего объекта
      void setStateAll(int a); // установка всем объектам одно состояние
      //   bool   changeHead(Class_base*   new_b_head_object);   //   метод
переопределения головного объекта для текущего в дереве иерархии. Метод
должен иметь один параметр, указатель на объект базового класса, содержащий
указатель на новый головной объект. Переопределение головного объект для
корневого объекта недопустимо. Недопустимо создать второй корневой объект.
Недопустимо при переопределении, чтобы у нового головного появились два
подчиненных объекта с одинаковым наименованием. Новый головной объект не
должен принадлежать к объектам из ветки текущего. Если переопределение
выполнено, метод возвращает значение «истина», иначе «ложь»
      // void deleteObjectName(string delete_b_name); // метод удаления
подчиненного объекта по наименованию. Если объект не найден, то метод
завершает работу. Один параметр строкового типа, содержит наименование
удаляемого подчиненного объекта
      // Class_base* getObjectCoordinate(string b_name_coordinate); // метод
получения указателя на любой объект в составе дерева иерархии объектов
согласно пути (координаты)

      void   set_connection(TYPE_SIGNAL   b_signal,   Class_base*   b_target,
TYPE_HANDLER b_handler); // установка соединения
      void   delete_connction(TYPE_SIGNAL   b_signal,   Class_base*   b_target,
TYPE_HANDLER b_handler); // удаление соеднинения
      void emit_signal(TYPE_SIGNAL b_signal, string message, Class_base*
p_object); // запуск сигнала в объекте и проходит по вектору соединений и
```

```
смотрит у каких объектов есть обработчик конкретных сигналов
     // string getPath(); // метод определения абсолютной пути до текущего
объекта

     static Class_base* base_s; // вспомогательный указатель
     int sum = 0, change_five = 0, change_ten = 0; // переменные хранящие
ввёденную сумму денег польхователем и деньги доступные для сдачи
     vector <product_info*> product; // вектор продуктов
};

#endif
```

5.3 Файл Class_Get_Change.cpp

Листинг 3 – Class_Get_Change.cpp

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

Class_Get_Change :: Class_Get_Change(Class_base* b_head, string b_name) :

//-------------------------------------------------------------------//
// Конструктор класса Class_5 с присвоением свойствам класса Class_5 //
// значения параметров b_head и b_name. Вызов конструктора класса    //
// Class_base с аргументами b_head и b_name                          //
//-------------------------------------------------------------------//

Class_base(b_head,  b_name)//  вызов  конструктора  класса  Class_base  с
параметрами b_head и b_name для создания объекта
{}


void Class_Get_Change::signal_f(string& msg) { // заголовок метода
   if(msg.find("/")==-1){ // проверка чтоб не было / в строке
     if(msg.substr(0,msg.find("$")) == "0" && msg.substr(msg.find("$")+1) !=
"0") msg = "\nTake the change: 5 * "+msg.substr(msg.find("$")+1)+" rub."; //
возврат сдачи
     else if(msg.substr(0,msg.find("$")) != "0" && msg.substr(msg.find("$")
+1) == "0") msg = "\nTake the change: 10 * "+msg.substr(0, msg.find("$"))+"
rub."; // возврат сдачи
     else if(msg.substr(0,msg.find("$")) != "0" && msg.substr(msg.find("$")
+1) != "0") msg="\nTake the change: 10 * "+msg.substr(0,msg.find("$"))+"
rub.,  5 * "+msg.substr(msg.find("$")+1)+" rub."; // возврат сдачи

   }else
     if(msg.substr(0,msg.find("/")) == "0" && msg.substr(msg.find("/")+1) !=
"0") msg = "\nTake the change: 5 * "+msg.substr(msg.find("/")+1)+" rub." +"\
nReady to work"; // возврат сдачи
     else if(msg.substr(0,msg.find("/")) != "0" && msg.substr(msg.find("/")
```

```
+1) == "0") msg = "\nTake the change: 10 * "+msg.substr(0, msg.find("/"))+"
rub." +"\nReady to work"; // возврат сдачи
     else if(msg.substr(0,msg.find("/")) != "0" && msg.substr(msg.find("/")
+1) != "0") msg="\nTake the change: 10 * "+msg.substr(0,msg.find("/"))+"
rub., 5 * "+msg.substr(msg.find("/")+1)+" rub." +"\nReady to work"; //
возврат сдачи

     }
void Class_Get_Change::handler_f(string msg) { // заголовок метода
  this->emit_signal((TYPE_SIGNAL)(&Class_Get_Change::signal_f),msg,base_s-
>getSubObject("Print")); // отправка сообщения с класса Class_Get_Change
классу объекта Print
}
```

5.4 Файл Class_Get_Change.h

Листинг 4 – Class_Get_Change.h

```
#ifndef __CLASS_5__H
#define __CLASS_5__H
#include "Class_base.h"
#include <iostream>
using namespace std;

class Class_Get_Change : public Class_base{ // наследование от класса
Class_base с модификатором доступа public

  public: // открытый модификатор доступа
     Class_Get_Change(Class_base* b_head, string b_name); // конструктор
     void signal_f(string & msg); // метод сигнала
     void handler_f(string msg); // метод обработчика
};

#endif
```

5.5 Файл Class_Get_Product.cpp

Листинг 5 – Class_Get_Product.cpp

```
#include "Class_Get_Product.h"
#include "Class_Read.h"
using namespace std;
Class_Get_Product :: Class_Get_Product(Class_base* b_head, string b_name) :

//-------------------------------------------------------------------//
```

```
// Конструктор класса Class_6 с присвоением свойствам класса Class_6 //
// значения параметров b_head и b_name. Вызов конструктора класса    //
// Class_base с аргументами b_head и b_name                          //
//-------------------------------------------------------------------//

Class_base(b_head,  b_name)//  вызов  конструктора  класса  Class_base  с
параметрами b_head и b_name для создания объекта
{}

void Class_Get_Product::signal_f(string& msg){ // заголовок метода

   if(msg!="\nReady to work") // проверка чтоб не было равно
      msg="\nTake the product "+msg; // присвоение значения
}

void Class_Get_Product::handler_f(string msg){ // заголовок метода
   if(msg.find("/")== -1){ // в строке нет символа /
      this->emit_signal((TYPE_SIGNAL)
(&Class_Get_Product::signal_f),msg,base_s->getSubObject("Print"));        //
отправка сообщения между сигналом и обработчиком
   }
   else{ // иначе
      this->emit_signal((TYPE_SIGNAL)
(&Class_Get_Product::signal_f),msg.substr(0,msg.find("/")),base_s-
>getSubObject("Print")); // отправка сообщения между сигналом и обработчиком
      this->emit_signal((TYPE_SIGNAL)
(&Class_Read::signal_f),msg.substr(msg.find("/")        +1),base_s        ->
getSubObject("GetChange"));   //   отправка   сообщения   между   сигналом   и
обработчиком
   }
   this->emit_signal((TYPE_SIGNAL)(&Class_Get_Product::signal_f),"\nReady  to
work",base_s -> getSubObject("Print")); // отправка сообщения между сигналом
и обработчиком о готовности работать дальше
}
```

5.6 Файл Class_Get_Product.h

Листинг 6 – Class_Get_Product.h

```
#ifndef __CLASS_7__H
#define __CLASS_7__H
#include "Class_base.h"
#include <vector>
#include <iostream>

class  Class_Get_Product  :  public  Class_base  {  //  наследование  от  класса
Class_base с модификатором доступа public
   public: // открытый модификатор доступа
      Class_Get_Product(Class_base* pt, string name); // конструктор с двумя
параметрами
```

```
        void signal_f (string&); // метод сигнала
        void handler_f(string); // метод обработчика
};



#endif
```

5.7 Файл Class_Print.cpp

Листинг 7 – Class_Print.cpp

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

Class_Print :: Class_Print(Class_base* b_head, string b_name) :

//-------------------------------------------------------------------//
// Конструктор класса Class_6 с присвоением свойствам класса Class_6 //
// значения параметров b_head и b_name. Вызов конструктора класса    //
// Class_base с аргументами b_head и b_name                          //
//-------------------------------------------------------------------//

Class_base(b_head,  b_name)//  вызов  конструктора  класса  Class_base  с
параметрами b_head и b_name для создания объекта
{}


void Class_Print :: handler_f(string& msg){ // заголовок метода со строковым
параметром сообщения

   //-------------------//
   // Метод обработчика //
   //-------------------//

   cout << msg; // вывод на экран
}
```

5.8 Файл Class_Print.h

Листинг 8 – Class_Print.h

```
#ifndef __CLASS_6__H
```

```
#define __CLASS_6__H
#include "Class_base.h"
#include <iostream>
using namespace std;

class Class_Print : public Class_base{ // наследование от класса Class_base
с модификатором доступа public

   public: // открытый модификатор доступа
      Class_Print(Class_base* b_head, string b_name); // конструктор
      void handler_f(string &msg); // метод обработчика
};

#endif
```

5.9 Файл Class_Pult.cpp

Листинг 9 – Class_Pult.cpp

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

Class_Pult :: Class_Pult(Class_base* b_head, string b_name) :

//--------------------------------------------------------------------//
// Конструктор класса Class_3 с присвоением свойствам класса Class_3 //
// значения параметров b_head и b_name. Вызов конструктора класса    //
// Class_base с аргументами b_head и b_name                           //
//--------------------------------------------------------------------//

Class_base(b_head, b_name) // вызов конструктора класса Class_base с
параметрами b_head и b_name для создания объекта
{}


void Class_Pult::signal_f(string& msg) { // заголовок метода
   if(msg=="money") // проверка на не хватку денег
      msg="\nThere is not enough money"; // присвоение соответсвующего
сообщения

   if (msg == "product") // проверка на не хватку продукта
      msg = "\nThere is no product"; // присвоение соответсвующего сообщения

   if (msg == "number") // проверка на неверный ввод номера продукта
      msg = "\nThere is no product with this number"; // присвоение
соответсвующего сообщения
}

void Class_Pult::handler_f(string msg) { // заголовок метода
```

```
   bool flag=true; // индикатор
   for  (int  i=0;i<base_s->getSubObject("System")->product.size();i++){  //
цикл по каждому элементу
     if     ((stoi(msg.c_str())==base_s->getSubObject("System")->product[i]-
>number_product) &&
        ((base_s->getSubObject("System")->product[i]                    ->
price_product)<=(base_s->getSubObject("System")->sum)) &&
        (base_s->getSubObject("System")->product[i]  ->  count_product  >  0))
{ // проверка условий для выдачи товара
        base_s->getSubObject("System")->sum-=base_s->getSubObject("System")-
>product[i]  ->  price_product;  //  вычитание  из  суммы  денег  внесённых
пользоватателем значение стоимости товара
        flag=false; // изменение значения индикатора

        base_s->getSubObject("System")->product[i] -> count_product -= 1; //
вычитание из количества продуктов 1, так как его купили
        if(base_s->getSubObject("System")->sum==0){ // внесенная сумма равно
0
           this->emit_signal((TYPE_SIGNAL)(&Class_Pult::signal_f),base_s   ->
getSubObject("System")     ->     product[i]     ->     name_product,base_s-
>getSubObject("GetProduct")); // вывод на экран названия продукта
           return;} // возврат
        else{
           this->emit_signal((TYPE_SIGNAL)(&Class_Pult::signal_f),base_s   ->
getSubObject("System")  ->  product[i]  ->  name_product+"/"+"change",base_s-
>getSubObject("GetProduct")); // вывод на экран названия продукта и выдача
сдачи
           return;} // возврат
     }
   }
   if (flag){ // индикатор выдачи продукта
     for (int i = 0; i < base_s -> getSubObject("System") -> product.size();
i++){ // пройти по всем элементам
        if   (stoi(msg.c_str())==base_s->getSubObject("System")->product[i]-
>number_product){ // введененный номер и номер товара совпали
           if(base_s->getSubObject("System")->product[i] -> count_product <=
0) // товара нет в наличии
              this->emit_signal((TYPE_SIGNAL)
(&Class_Pult::signal_f),"product",base_s->getSubObject("Print"));  //  вывод
сообщения о том, что продукта нет
           else       if((base_s->getSubObject("System")->product[i]       ->
price_product)>(base_s->getSubObject("System")->sum))   //   сумма   меньше
стоимости товара
              this->emit_signal((TYPE_SIGNAL)
(&Class_Pult::signal_f),"money",base_s->getSubObject("Print"));   //   вывод
сообщения о том, что денег недостаточно
           return; // возврат
        }
     }
     this->emit_signal((TYPE_SIGNAL)(&Class_Pult::signal_f),"number",base_s-
>getSubObject("Print")); // вывод соообщения о том, что неверно набран номер
товара
}
}
```

5.10 Файл Class_Pult.h

Листинг 10 – Class_Pult.h

```
#ifndef __CLASS_3__H
#define __CLASS_3__H
#include "Class_base.h"
#include <iostream>
using namespace std;

class Class_Pult : public Class_base{ // наследование от класса Class_base с
модификатором доступа public

   public: // открытый модификатор доступа
      Class_Pult(Class_base* b_head, string b_name); // конструктор
      void signal_f(string & msg); // метод сигнала
      void handler_f(string msg); // метод обработчика
};

#endif
```

5.11 Файл Class_Read.cpp

Листинг 11 – Class_Read.cpp

```
#include "Class_Read.h"
#include "Class_base.h"
#include <iostream>
#include <string>

using namespace std;

Class_Read :: Class_Read(Class_base* b_head, string b_name) :

//-------------------------------------------------------------------//
// Конструктор класса Class_2 с присвоением свойствам класса Class_2 //
// значения параметров b_head и b_name. Вызов конструктора класса    //
// Class_base с аргументами b_head и b_name                          //
//-------------------------------------------------------------------//

Class_base(b_head,  b_name)  //  вызов  конструктора  класса  Class_base  с
параметрами b_head - указатель на головной объект(родителя) и b_name - имя
объекта для создания объекта
{}

void Class_Read::signal_f(string& msg){ // заголовок метода

   if (msg=="Refund money"){ // сообщение о возврат денег
      int  fife,ten;  //  объявление  переменнхы  хранящих  количество  5  и  10
```

```
рублевых монет
     ten=(base_s->getSubObject("System")->sum)/10; // присваиваем количество
монет номиналом 10 в ten
     base_s->getSubObject("System")->sum-=ten*10;  //  вычитание  из  суммы
внесенной пользователем колчиество монет номиналом 10
     fife=base_s->getSubObject("System")->sum/5;  //  присваиваем  количество
монет номиналом 5 в five
     base_s->getSubObject("System")->sum-=fife*5;  //  вычитание  из  суммы
внесенной пользователем колчиество монет номиналом 5
     base_s->getSubObject("System")->change_five-=fife;   //   вычитание   из
переменной хранящей сдачу 5 рублевыми монетами значение five
     base_s->getSubObject("System")->change_ten-=ten;   //   вычитание   из
переменной хранящей сдачу 10 рублевыми монетами значение ten
     msg="Refund money" + to_string(ten)+"/"+to_string(fife); // присвоение
о количество возвращенных денег
   }

   else if(msg=="change"){ // сообщение о возврат сдачи
     int  fife,ten;  //  объявление  переменнхы  хранящих  количество  5  и  10
рублевых монет
     ten   =   (base_s->getSubObject("System")->sum)/10;   //   присваиваем
количество монет номиналом 10 в ten
     base_s  ->  getSubObject("System")->sum-=ten*10;  // вычитание из суммы
внесенной пользователем колчиество монет номиналом 10
     fife = base_s->getSubObject("System")->sum/5; // присваиваем количество
монет номиналом 5 в five
     base_s  ->  getSubObject("System")->sum-=fife*5;  // вычитание из суммы
внесенной пользователем колчиество монет номиналом 5
     base_s  ->  getSubObject("System")->change_five-=fife;  //  вычитание  из
переменной хранящей сдачу 5 рублевыми монетами значение five
     base_s  ->  getSubObject("System")->change_ten-=ten;  //  вычитание  из
переменной хранящей сдачу 10 рублевыми монетами значение ten
     msg  =  to_string(ten)+"$"+to_string(fife);  //  присвоение  значение
количество 5 и 10 а между ними спец сигнал ля обработки $
   }

   else if(msg=="false") // значение false
     msg="\nTake the money back, no change"; // присвоение, что сдачи нет
для введенной суммы
}

void Class_Read::handler_f(string msg){

   if (msg.substr(0, 12)=="Refund money"){ // если введена команда возврата
средств
     this->emit_signal((TYPE_SIGNAL)(&Class_Read::signal_f),msg.substr(12,
msg.size()-1),base_s->getSubObject("GetChange"));} // делаем возврат
   else if (msg.substr(0, 7) == "Product"){ // если задана команда выдачи
продукта
     this->emit_signal((TYPE_SIGNAL)(&Class_Read::signal_f),msg.substr(8,
msg.size()-1),base_s->getSubObject("Pult")); // отправка на пулт управления
   }

   else if (msg=="Turn off the system"){ // если введена команда Turn off the
system
     this->emit_signal((TYPE_SIGNAL)(&Class_Read::signal_f),"\nTurned
```

```
off",base_s->getSubObject("Print")); // Отправляем сообщение в объект Print
для и завершаем работу автомата
   }

   else if (msg=="SHOWTREE"){ // если введена команда SHOWTREE
      cout << endl;
      base_s->getSubObject("System")->showObjectsState();  //  вызов  метода
вывода состояния объектов в иерархии
   }

   else{ // иначе
      if (stoi(msg.c_str())==50){ // внесенная сумма равно 50
         if        (((base_s->getSubObject("System")->change_five*5)+(base_s-
>getSubObject("System")->change_ten*10))<50            +            base_s-
>getSubObject("System")->sum) // проверка на возможность выдачи сдачи
            this->emit_signal((TYPE_SIGNAL)
(&Class_Read::signal_f),"false",base_s->getSubObject("Print"));  //  отправка
с сигнала значение false на печать
         else // иначе
            this->emit_signal((TYPE_SIGNAL)
(&Class_Read::signal_f),msg,base_s->getSubObject("SetMoney")); // внесенение
этой суммы на баланс
      }
      else if (stoi(msg.c_str())==100){
         if        (((base_s->getSubObject("System")->change_five*5)+(base_s-
>getSubObject("System")->change_ten*10))<100            +            base_s-
>getSubObject("System")->sum)// проверка на возможность выдачи сдачи
            this->emit_signal((TYPE_SIGNAL)
(&Class_Read::signal_f),"false",base_s->getSubObject("Print"));  //  отправка
с сигнала значение false на печать
         else // иначе
            this->emit_signal((TYPE_SIGNAL)
(&Class_Read::signal_f),msg,base_s->getSubObject("SetMoney")); // внесенение
этой суммы на баланс
      }
      else{
         this->emit_signal((TYPE_SIGNAL)(&Class_Read::signal_f),msg,base_s-
>getSubObject("SetMoney")); // внесенение этой суммы на баланс
         if(msg=="5") // пользователь внес 5 рублей
            base_s->getSubObject("System")->change_five+=1;   //   внесение   в
переменную хранящую количество 5 рублевых монет + 1
         if(msg=="10") // пользователь внес 10 рублей
            base_s->getSubObject("System")->change_ten+=1;   //   внесение   в
переменную хранящую количество 10 рублевых монет + 1
      }
   }
}

void Class_Read :: signal_start(string& msg){ // заголовок метода


   int   n,   n_product,c_product,p_product;   //   объявление   переменных
характеристик товара
   string name_product; // название товара
```

```
   n = atoi(msg.c_str()); // внесение количества товаров для ввода

   cin >> base_s -> getSubObject("System") -> change_five; // ввод сдачи 5
рублевыми монетами
   cin >> base_s -> getSubObject("System") -> change_ten; // ввод сдачи 10
рублевыми монетами

   product_info* product_; // указатель для добавления объекта в вектор
продуктов
   for (int i=0;i<n;i++){ // массив пока не достигнет количества n
      cin >> n_product >> c_product >> p_product; // ввод характеристик
продукта
      getline(cin, name_product); // ввод названия продукта
      product_ = new product_info(); // создание объекта
      product_ -> number_product = n_product; // присвоение полю объекта
product_ значение номера продукта
      product_ -> count_product = c_product; // присвоение полю объекта
product_ значение количества продукта
      product_ -> price_product = p_product; // присвоение полю объекта
product_ значение стоиости продукта
      product_ -> name_product = name_product.substr(1, name_product.size());
// присвоение полю объекта product_ название прродукта
      base_s->getSubObject("System")   ->   product.push_back(product_);   //
добавление продукта в вектор продуктов
   }

   this->emit_signal((TYPE_SIGNAL)(&Class_Read::signal_f),"Ready          to
work",base_s->getSubObject("Print")); // запуск работы автомата
}
```

5.12 Файл Class_Read.h

Листинг 12 – Class_Read.h

```
#ifndef __CLASS_2__H
#define __CLASS_2__H
#include "Class_base.h"
#include <iostream>
using namespace std;

class Class_Read : public Class_base{ // наследование от класса Class_base с
модификатором доступа public

   public: // открытый модификатор доступа
      Class_Read(Class_base* b_head, string b_name); // конструктор
      void signal_f(string & msg); // метод сигнала
      void handler_f(string msg); // метод обработчика
      void  signal_start(string&  msg);  //  метод  сигнала  для  начала  работы
автомата
};
```

```
#endif
```

5.13 Файл Class_Set_Money.cpp

Листинг 13 – Class_Set_Money.cpp

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

Class_Set_Money :: Class_Set_Money(Class_base* b_head, string b_name) :

//-------------------------------------------------------------------//
// Конструктор класса Class_4 с присвоением свойствам класса Class_4 //
// значения параметров b_head и b_name. Вызов конструктора класса    //
// Class_base с аргументами b_head и b_name                          //
//-------------------------------------------------------------------//

Class_base(b_head,  b_name)//  вызов  конструктора  класса  Class_base  с
параметрами b_head и b_name для создания объекта
{}

void Class_Set_Money::signal_f(string& msg) { // заголовок метода
   msg ="\nThe amount: " + msg; // вывод значения баланса
}
void Class_Set_Money::handler_f(string msg) { // заголовок метода
   base_s->getSubObject("System") -> sum +=stoi(msg.c_str()); // присвоение
общей сумме денег значение новых денег
   emit_signal((TYPE_SIGNAL)(&Class_Set_Money::signal_f),to_string(base_s-
>getSubObject("System")->sum),base_s->getSubObject("Print"));  //  вывод  на
экран
}
```

5.14 Файл Class_Set_Money.h

Листинг 14 – Class_Set_Money.h

```
#ifndef __CLASS_4__H
#define __CLASS_4__H
#include "Class_base.h"
#include <iostream>
using namespace std;

class  Class_Set_Money  :  public  Class_base{  //  наследование  от  класса
```

```
Class_base с модификатором доступа public

   public: // открытый модификатор доступа
      Class_Set_Money(Class_base* b_head, string b_name); // конструктор
      void signal_f(string & msg); // метод сигнала
      void handler_f(string msg); // метод обработчика
};

#endif
```

5.15 Файл Class_System.cpp

Листинг 15 – Class_System.cpp

```
#include "Class_System.h"
#include "Class_Pult.h"
#include "Class_Set_Money.h"
#include "Class_Get_Change.h"
#include "Class_Print.h"
#include "Class_Read.h"
#include "Class_Get_Product.h"
#include <iostream>
using namespace std;

Class_System :: Class_System(Class_base* p_head) :

//-------------------------------------------------------------------------//
// Конструктор класса Class_application с присвоением свойствам класса //
// Class_application значения параметров b_head и b_name.              //
// Вызов конструктора класса Class_base с аргументом b_head            //
//-------------------------------------------------------------------------//

Class_base(base_s) // вызов конструктора класса Class_base с параметром
p_head - указатель
{}



void Class_System :: build_tree_objects(){ // заголовок метода

   //-------------------------------//
   // Метод ввода иерархии объектов //
   //-------------------------------//

   setName("System"); // установка имени корневому объекту

   vector <TYPE_SIGNAL> signals = {SIGNAL_D(Class_Read :: signal_f ),
SIGNAL_D( Class_Pult :: signal_f ), SIGNAL_D( Class_Set_Money :: signal_f ),
SIGNAL_D( Class_Get_Change :: signal_f ), SIGNAL_D( Class_Get_Product ::
signal_f ) }; // инициализация вектора сигналов для каждого класса
   vector <TYPE_HANDLER> handler = {HANDLER_D(Class_Read :: handler_f ),
HANDLER_D( Class_Pult :: handler_f ), HANDLER_D( Class_Set_Money ::
handler_f ), HANDLER_D( Class_Get_Change :: handler_f ),
HANDLER_D( Class_Print :: handler_f ), HANDLER_D( Class_Get_Product ::
handler_f ) }; // инициализация вектора обработчика для каждого класса


   new Class_Read(this, "Read"); // создание объекта
   new Class_Pult(this, "Pult");// создание объекта
   new Class_Set_Money(this, "SetMoney");// создание объекта
   new Class_Get_Change(this, "GetChange");// создание объекта
   new Class_Print(this, "Print");// создание объекта
   new Class_Get_Product(this, "GetProduct");// создание объекта
```

```
   base_s->getSubObject("System")->set_connection(signals[0],         base_s-
>getSubObject("Read"),handler[0]); // установка связи между объектами

   base_s->getSubObject("Read")->set_connection(signals[0],base_s-
>getSubObject("SetMoney"),handler[2]); // установка связи между объектами

   base_s->getSubObject("SetMoney")->set_connection(signals[2],base_s-
>getSubObject("Print"),handler[4]); // установка связи между объектами

   base_s->getSubObject("Read")->set_connection(signals[0],base_s-
>getSubObject("Print"),handler[4]);// установка связи между объектами

   base_s->getSubObject("System")->set_connection(signals[0],base_s-
>getSubObject("Print"),handler[4]);// установка связи между объектами

   base_s->getSubObject("Read")->set_connection(signals[0],base_s-
>getSubObject("GetChange"),handler[3]);// установка связи между объектами

   base_s->getSubObject("GetChange")->set_connection(signals[3],base_s-
>getSubObject("Print"),handler[4]);// установка связи между объектами

   base_s->getSubObject("Read")->set_connection(signals[0],base_s-
>getSubObject("Pult"),handler[1]);// установка связи между объектами

   base_s->getSubObject("Pult")->set_connection(signals[1],base_s-
>getSubObject("Print"),handler[4]);// установка связи между объектами

   base_s->getSubObject("Pult")->set_connection(signals[1],base_s-
>getSubObject("GetProduct"),handler[5]);// установка связи между объектами

   base_s->getSubObject("GetProduct")->set_connection(signals[4],base_s-
>getSubObject("Print"),handler[4]);// установка связи между объектами

   base_s->getSubObject("GetProduct")->set_connection(signals[0],base_s-
>getSubObject("GetChange"),handler[3]);// установка связи между объектами



}

int Class_System :: exec_app(){ // заголовок метода

   //--------------------------//
   // Метод запуска приложения //
   //--------------------------//

   base_s -> setStateAll(1); // установка всем объектам состояние 1, активное

   string cmd; // объявление переменной команды
   cin >> cmd; // ввод команды

   base_s->getSubObject("System")->emit_signal((TYPE_SIGNAL)
(&Class_Read::signal_start),cmd,base_s->getSubObject("Print"));   //   запуск
первого сигнала который вводит продукты и количества сдачи
```

```
   while(true){ // бесконечный
      getline(cin, cmd); // получение всей строки
      if(cmd == "Turn off the system" || cmd == "SHOWTREE"){ // если равно
чему - то из команд завершения или
         base_s->getSubObject("System")->emit_signal((TYPE_SIGNAL)
(&Class_Read::signal_f),cmd,base_s->getSubObject("Read"));    //    отправка
команды
         break; // остановка цикла
      }
      base_s->getSubObject("System")->emit_signal((TYPE_SIGNAL)
(&Class_Read::signal_f),cmd,base_s->getSubObject("Read"));    //    отправка
команды
   }

   return(0); // возврат 0

}
```

5.16 Файл Class_System.h

Листинг 16 – Class_System.h

```
#ifndef __CLASS_APPLICATION__H
#define __CLASS_APPLICATION__H
#include "Class_base.h"
#include <iostream>
using namespace std;

class Class_System : public Class_base{ // наследование от класса Class_base
с модификатором доступа public

   public: // открытый модификатор доступа
      Class_System(Class_base* p_head); // конструктор

      void build_tree_objects(); // метод постройки дерева, то есть ввод с
клавиатуры данных
      int exec_app(); // запуск приложения
};

#endif
```

5.17 Файл main.cpp

Листинг 17 – main.cpp

```
#include <stdlib.h>
#include <stdio.h>
#include "Class_System.h"
#include <iostream>
using namespace std;

int main()
{
   Class_System ob_cl_application (nullptr);  // создание объекта приложение
   ob_cl_application.build_tree_objects  (  );              //  конструирование
системы,
   return ob_cl_application.exec_app ( );          // запуск системы
}
```

6 ТЕСТИРОВАНИЕ

Результат тестирования программы представлен в таблице 36.

Таблица 36 – Результат тестирования программы

Входные данные	Ожидаемые выходные данные	Фактические выходные данные
10 13 50 10 20 50 Water Holy Spring 11 20 45 Water Shishkin forest 12 15 115 Juice Gardens of the Don region 13 11 40 Mineral water Essentuki № 4 14 18 150 Juice Gardens of the Don region 25 20 85 Chocolate Alyonka 19 20 130 Ritter Sport dark chocolate with mint filling 77 9 50 Chocolate Kinder 31 20 65 Baunty 44 1 35 Nougat Stylish things with almonds and cranberries in milk chocolate glaze 50 100 Product 14 10 10 10 Product 44 10 Product 44 5 5 Refund money 100 Product 44 Product 45 Product 25 50 10	Ready to work The amount: 50 The amount: 150 Take the product Juice Gardens of the Don region Ready to work The amount: 10 The amount: 20 The amount: 30 There is not enough money The amount: 40 Take the product Nougat Stylish things with almonds and cranberries in milk chocolate glaze Take the change: 5 * 1 rub. Ready to work The amount: 5 The amount: 10 Take the change: 10 * 1 rub. Ready to work The amount: 100 There is no product There is no product with this number Take the product Chocolate Alyonka Take the change: 10 * 1 rub., 5 * 1 rub. Ready to work The amount: 50 The amount: 60 Turned off	Ready to work The amount: 50 The amount: 150 Take the product Juice Gardens of the Don region Ready to work The amount: 10 The amount: 20 The amount: 30 There is not enough money The amount: 40 Take the product Nougat Stylish things with almonds and cranberries in milk chocolate glaze Take the change: 5 * 1 rub. Ready to work The amount: 5 The amount: 10 Take the change: 10 * 1 rub. Ready to work The amount: 100 There is no product There is no product with this number Take the product Chocolate Alyonka Take the change: 10 * 1 rub., 5 * 1 rub. Ready to work The amount: 50 The amount: 60 Turned off

Входные данные	Ожидаемые выходные данные	Фактические выходные данные
Turn off the system		
SHOWTREE	Ready to work System is ready Read is ready Pult is ready SetMoney is ready GetChange is ready Print is ready GetProduct is ready	Ready to work System is ready Read is ready Pult is ready SetMoney is ready GetChange is ready Print is ready GetProduct is ready
11 13 50 10 20 50 Water Holy Spring 11 20 45 Water Shishkin forest 12 15 115 Juice Gardens of the Don region 13 11 40 Mineral water Essentuki № 4 14 18 150 Juice Gardens of the Don region 25 20 85 Chocolate Alyonka 19 20 130 Ritter Sport dark chocolate with mint filling 77 9 50 Chocolate Kinder 31 20 65 Baunty 44 1 35 Nougat Stylish things with almonds and cranberries in milk chocolate glaze 45 1 40 Палпи 50 100 Product 14 10 10 10 Product 44 10 Product 44 5	Ready to work The amount: 50 The amount: 150 Take the product Juice Gardens of the Don region Ready to work The amount: 10 The amount: 20 The amount: 30 There is not enough money The amount: 40 Take the product Nougat Stylish things with almonds and cranberries in milk chocolate glaze Take the change: 5 * 1 rub. Ready to work The amount: 5 The amount: 10 Take the change: 10 * 1 rub. Ready to work The amount: 100 There is no product Take the product Палпи Take the change: 10 * 6 rub. Ready to work There is not enough money The amount: 50 The amount: 60 There is not enough money	Ready to work The amount: 50 The amount: 150 Take the product Juice Gardens of the Don region Ready to work The amount: 10 The amount: 20 The amount: 30 There is not enough money The amount: 40 Take the product Nougat Stylish things with almonds and cranberries in milk chocolate glaze Take the change: 5 * 1 rub. Ready to work The amount: 5 The amount: 10 Take the change: 10 * 1 rub. Ready to work The amount: 100 There is no product Take the product Палпи Take the change: 10 * 6 rub. Ready to work There is not enough money The amount: 50 The amount: 60 There is not enough

Входные данные	Ожидаемые выходные данные	Фактические выходные данные
5 Refund money 100 Product 44 Product 45 Product 25 50 10 Product 14 10 10 10 Product 44 10 Product 44 5 5 Refund money 100 Product 44 Product 45 Product 25 50 10 Turn off the system	The amount: 70 The amount: 80 The amount: 90 There is no product The amount: 100 There is no product The amount: 105 The amount: 110 Take the change: 10 * 11 rub. Ready to work The amount: 100 There is no product There is no product Take the product Chocolate Alyonka Take the change: 10 * 1 rub., 5 * 1 rub. Ready to work The amount: 50 The amount: 60 Turned off	money The amount: 70 The amount: 80 The amount: 90 There is no product The amount: 100 There is no product The amount: 105 The amount: 110 Take the change: 10 * 11 rub. Ready to work The amount: 100 There is no product There is no product Take the product Chocolate Alyonka Take the change: 10 * 1 rub., 5 * 1 rub. Ready to work The amount: 50 The amount: 60 Turned off
11 13 50 10 20 50 Water Holy Spring 11 20 45 Water Shishkin forest 12 15 115 Juice Gardens of the Don region 13 11 40 Mineral water Essentuki № 4 14 18 150 Juice Gardens of the Don region 25 20 85 Chocolate Alyonka 19 20 130 Ritter Sport dark chocolate with mint filling 77 9 50 Chocolate Kinder 31 20 65 Baunty 44 1 35 Nougat Stylish things with almonds and cranberries in milk	Ready to work System is ready Read is ready Pult is ready SetMoney is ready GetChange is ready Print is ready GetProduct is ready	Ready to work System is ready Read is ready Pult is ready SetMoney is ready GetChange is ready Print is ready GetProduct is ready

Входные данные	Ожидаемые выходные данные	Фактические выходные данные
chocolate glaze 45 1 40 Палпи SHOWTREE		

ЗАКЛЮЧЕНИЕ

Осуществлено моделирование работы системы вендингово автомата, использующей следующий объекты: система, объект пульта управления, объект устройства приёма денег, объект устройства выдачи продукта, объект устройства возврата сдачи, объект экрана отображения состояния и информации. Реализован весь необходимый функционал программы, все упомянутые в постановке задачи сигналы, команды и обработчики. Корректность решения подтверждена тестами программы, произведёнными системой "Аврора", а также приведёнными непосредственно в тексте отчёта по выполненному заданию.

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

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).