



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение  
высшего образования

«МИРЭА – Российский технологический университет»

**РТУ МИРЭА**

Институт информационных технологий  
Кафедра вычислительной техники

## КУРСОВАЯ РАБОТА

По дисциплине

«Объектно-ориентированное программирование»  
(наименование дисциплины)

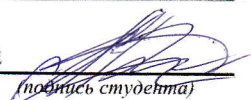
Тема курсовой работы

Моделирование работы инженерного арифметического  
калькулятора  
(наименование темы)

Студент группы

ИМБО-01-21  
(учебная группа)

Рутковская Анастасия Алексеевна  
(Фамилия Имя Отчество)

  
(подпись студента)

Руководитель курсовой работы

ст. преп. Грач Е.П.

(Должность, звание, ученая степень)

(подпись руководителя)

Консультант

доц. каф. ВТ Унгер А.Ю.

(Должность, звание, ученая степень)

(подпись консультанта)

Работа представлена к защите « » 2022 г.

Допущен к защите « » 2022 г.

Москва 2022 г.



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение  
высшего образования

«МИРЭА – Российский технологический университет»

**РТУ МИРЭА**

Институт информационных технологий

Кафедра вычислительной техники

**Утверждаю**

Заведующий кафедрой

Платонова О.В.

ФИО

« 14 » марта 2022г.

**ЗАДАНИЕ**

**На выполнение курсовой работы**

по дисциплине «Объектно-ориентированное программирование»

Студент Рутковская Анастасия Алексеевна

Группа ИМБО-01-21

**Тема**

Моделирование работы инженерного арифметического

**Исходные данные:**

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

**Перечень вопросов, подлежащих разработке, и обязательного графического материала:**

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

**Срок представления к защите курсовой работы:** до « 16 » мая 2022 г.

**Задание на курсовую работу выдал**

Подпись

( Нутуридзе З.Ш. )

ФИО консультанта

« 28 » февраля 2022 г.

**Задание на курсовую работу получил**

Подпись

( Рутковская А.А. )

ФИО исполнителя

« 28 » февраля 2022 г.

Москва 2022г.

# СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	8
1 ПОСТАНОВКА ЗАДАЧИ .....	9
1.1 Описание входных данных .....	11
1.2 Описание выходных данных.....	11
2 МЕТОД РЕШЕНИЯ .....	13
3 ОПИСАНИЕ АЛГОРИТМОВ.....	21
3.1 Алгоритм функции main.....	21
3.2 Алгоритм конструктора класса cl_1 .....	21
3.3 Алгоритм конструктора класса cl_2.....	22
3.4 Алгоритм метода handler_reader класса cl_2 .....	22
3.5 Алгоритм метода signal_operation класса cl_2 .....	23
3.6 Алгоритм конструктора класса cl_3.....	23
3.7 Алгоритм метода plus класса cl_3 .....	23
3.8 Алгоритм метода minus класса cl_3 .....	24
3.9 Алгоритм метода umn класса cl_3.....	24
3.10 Алгоритм метода del класса cl_3 .....	24
3.11 Алгоритм метода ost класса cl_3 .....	25
3.12 Алгоритм метода handler_operation класса cl_3.....	25
3.13 Алгоритм метода signal_print класса cl_3.....	31
3.14 Алгоритм метода signal_num_to_4 класса cl_3 .....	31
3.15 Алгоритм метода signal_oper_to_4 класса cl_3 .....	32
3.16 Алгоритм метода signal_res_to_4 класса cl_3.....	32
3.17 Алгоритм метода handler_num_to_3 класса cl_3.....	32
3.18 Алгоритм метода handler_oper_to_3 класса cl_3.....	33
3.19 Алгоритм метода handler_res_to_3 класса cl_3 .....	33
3.20 Алгоритм конструктора класса cl_4.....	33

3.21 Алгоритм метода <code>sdv_lev</code> класса <code>cl_4</code> .....	34
3.22 Алгоритм метода <code>sdv_prav</code> класса <code>cl_4</code> .....	34
3.23 Алгоритм метода <code>handler_operation</code> класса <code>cl_4</code> .....	35
3.24 Алгоритм метода <code>signal_print</code> класса <code>cl_4</code> .....	37
3.25 Алгоритм метода <code>signal_num_to_3</code> класса <code>cl_4</code> .....	38
3.26 Алгоритм метода <code>signal_oper_to_3</code> класса <code>cl_4</code> .....	38
3.27 Алгоритм метода <code>signal_res_to_3</code> класса <code>cl_4</code> .....	38
3.28 Алгоритм метода <code>handler_num_to_4</code> класса <code>cl_4</code> .....	39
3.29 Алгоритм метода <code>handler_oper_to_4</code> класса <code>cl_4</code> .....	39
3.30 Алгоритм метода <code>handler_res_to_4</code> класса <code>cl_4</code> .....	40
3.31 Алгоритм конструктора класса <code>cl_5</code> .....	40
3.32 Алгоритм метода <code>handler_operation</code> класса <code>cl_5</code> .....	40
3.33 Алгоритм метода <code>signal_res_5</code> класса <code>cl_5</code> .....	41
3.34 Алгоритм метода <code>signal_num_5</code> класса <code>cl_5</code> .....	41
3.35 Алгоритм метода <code>signal_oper_5</code> класса <code>cl_5</code> .....	42
3.36 Алгоритм конструктора класса <code>cl_6</code> .....	42
3.37 Алгоритм метода <code>handler_print</code> класса <code>cl_6</code> .....	43
3.38 Алгоритм метода <code>HEX_16</code> класса <code>cl_6</code> .....	45
3.39 Алгоритм метода <code>BIN_2</code> класса <code>cl_6</code> .....	48
3.40 Алгоритм метода <code>signal_res_6</code> класса <code>cl_6</code> .....	49
3.41 Алгоритм метода <code>signal_num_1_6</code> класса <code>cl_6</code> .....	50
3.42 Алгоритм конструктора класса <code>cl_application</code> .....	50
3.43 Алгоритм метода <code>build_tree_objects</code> класса <code>cl_application</code> .....	50
3.44 Алгоритм метода <code>exec_app</code> класса <code>cl_application</code> .....	54
3.45 Алгоритм метода <code>signal_reader</code> класса <code>cl_application</code> .....	55
3.46 Алгоритм метода <code>handler_operation</code> класса <code>cl_application</code> .....	55
3.47 Алгоритм конструктора класса <code>cl_base</code> .....	56

3.48 Алгоритм деструктора класса cl_base.....	56
3.49 Алгоритм метода get_name класса cl_base .....	57
3.50 Алгоритм метода set_name класса cl_base.....	57
3.51 Алгоритм метода get_parent класса cl_base.....	58
3.52 Алгоритм метода set_parent класса cl_base .....	58
3.53 Алгоритм метода get_ptr_by_name класса cl_base.....	59
3.54 Алгоритм метода get_state класса cl_base.....	60
3.55 Алгоритм метода set_state класса cl_base .....	60
3.56 Алгоритм метода print_state класса cl_base.....	61
3.57 Алгоритм метода set_connection класса cl_base.....	62
3.58 Алгоритм метода delete_connection класса cl_base.....	62
3.59 Алгоритм метода emit_signal класса cl_base .....	63
4 БЛОК-СХЕМЫ АЛГОРИТМОВ .....	65
5 КОД ПРОГРАММЫ.....	117
5.1 Файл cl_1.h .....	117
5.2 Файл cl_2.cpp .....	117
5.3 Файл cl_2.h .....	118
5.4 Файл cl_3.cpp .....	118
5.5 Файл cl_3.h .....	121
5.6 Файл cl_4.cpp .....	122
5.7 Файл cl_4.h .....	124
5.8 Файл cl_5.cpp .....	124
5.9 Файл cl_5.h .....	125
5.10 Файл cl_6.cpp .....	125
5.11 Файл cl_6.h.....	128
5.12 Файл cl_application.cpp .....	129
5.13 Файл cl_application.h .....	131

5.14 Файл cl_base.cpp .....	132
5.15 Файл cl_base.h.....	135
5.16 Файл main.cpp .....	136
6 ТЕСТИРОВАНИЕ .....	137
ЗАКЛЮЧЕНИЕ .....	139
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ .....	140

## **ВВЕДЕНИЕ**

Объектно-ориентированное программирование может использоваться во многих сферах, связанных со специальностью "Прикладная математика". Так, ООП во многих языках программирования можно использовать, например, для построения различных графиков или вычислений функций. Использование ООП в области хранения "больших данных" является актуальным в современном мире. Обладание навыками данной парадигмы программирования будет полезно для профессиональной деятельности.

# 1 ПОСТАНОВКА ЗАДАЧИ

Надо моделировать работу калькулятора следующей конструкции:

в вычислении участвуют целые числа объемом памяти 2 байта;

допустимые операции: +, -, \*, / (целочисленное деление), % (деление с остатком), << (побитовый сдвиг влево), >> (побитовый сдвиг в право);

операции выполняются последовательно, для выполнения операции необходимы два аргумента и знак операции;

после выполнения каждой операции фиксируется и выводится результат;

последовательность операций и аргументов образует выражение;

результат отображается в 16, 10 и 2-ой системе счисления;

при возникновении переполнения выдается Overflow;

при попытке деления на 0 выдается Division by zero;

при вводе знака “C” калькулятор приводится в исходное состояние, первый аргумент выражения принимает значение 0 и готов для ввода очередного выражения;

при вводе знака “Off” калькулятор завершает работу.

Нажатие на клавиши калькулятора моделируется посредством клавиатурного ввода. Ввод делится на команды:

1. «целое число» - первый аргумент выражения, целое не отрицательное число, можно последовательно вводить несколько раз, предыдущее значение меняется. При вводе не первым аргументом выражения - игнорируется;
2. «знак операции» «целое число» - второе и последующие операции выражения;
3. «C» - приведение калькулятора в исходное состояние;
4. «Off» - завершение работы калькулятора.

Вывод результата моделируется посредством вывода на консоли. Результат



выводиться в следующей форме:

«выражение»      HEX «16-ое число»    DEC «10-ое число»    BIN «2-ое число»

«16-ое число» выводиться в верхнем регистре с лидирующими нулями (пример 01FA).

«10-ое число» (пример 1765).

«2-ое число» выводиться разбивкой по четыре цифры с лидирующими нулями (пример 0000 0100 0111 0101).

Построить систему, которая использует объекты:

- Объект «система».
- Объект для чтения команд. После чтения очередной команды объект выдает сигнал с текстом, содержащим команду. Все команды синтаксически корректны (моделирует пульт управления калькулятора).
- Объект для выполнения арифметических операции. После завершения выдается сигнал с текстом результата. Если произошло переполнение или деление на ноль, выдается сигнал об ошибке. После выдачи сообщения калькулятор переводится посредством соответствующего сигнала в исходное положение.
- Объект для выполнения операции побитового сдвига. После завершения выдается сигнал с текстом результата.
- Объект для выполнения операции «С».
- Объект для вывода очередного результата 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 команд.

2.2.1. Выдача сигнала объекту для ввода команды.

2.2.2. Обработка команды.

2.3. После ввода команды «Off» завершить работу.

Все приведенные сигналы и соответствующие обработчики должны быть реализованы.

Все сообщения на консоль выводятся с новой строки.

В набор поддерживаемых команд добавить команду «SHOWTREE» и по этой команде вывести дерево иерархии объектов системы с отметкой о готовности и завершить работу программы.

### 1.1 Описание входных данных

Построчно множество команд, в любом количестве. Перечень команд:

«целое не отрицательное число»

«знак операции» «целое число»

C

Последняя команда присутствует всегда:

Off

Пример ввода

5 + 5 << 1 / 0 + 5 C 7 8 / -3 C 9 % -4 + 7 \* 11 Off

### 1.2 Описание выходных данных

Построчно выводиться результат каждой операции по форме:

«выражение» HEX «16-ое число» DEC «10-ое число» BIN «2-ое число»

Если произошло переполнение:

«выражение» Overflow

Если произошло переполнение:

«выражение» Division by zero

Пример вывода:

$5 + 5$     HEX 000A    DEC 10    BIN 0000 0000 0000 1010

$5 + 5 << 1$     HEX 0014    DEC 20    BIN 0000 0000 0001 0100

$5 + 5 << 1 / 0$     Division by zero

$0 + 5$     HEX 0005    DEC 5    BIN 0000 0000 0000 0101

$8 / -3$     HEX FFFE    DEC -2    BIN 1111 1111 1111 1110

$9 \% -4$     HEX 0001    DEC 1    BIN 0000 0000 0000 0001

$9 \% -4 + 7$     HEX 0008    DEC 8    BIN 0000 0000 0000 1000

$9 \% -4 + 7 * 11$     HEX 0058    DEC 88    BIN 0000 0000 0101 1000

## 2 МЕТОД РЕШЕНИЯ

Для решения задачи используется:

- Оператор цикла
- Условный оператор
- Стандартные объекты потока ввода/вывода `cin/cout`
- Объекты классов

Иерархия наследования отображена в таблице 1.

Таблица 1 – Иерархия наследования классов

№	Имя класса	Классы-наследники	Модификатор доступа при наследовании	Описание	Номер	Комментарии
1	cl_base			Базовый класс в иерархии		
		cl_1	public		2	
		cl_application	public		3	
		cl_2	public		4	
		cl_3	public		5	
		cl_4	public		6	
		cl_5	public		7	
		cl_6	public		8	
2	cl_1			Класс, наследуемый от cl_base		
3	cl_application			Класс для построения		

				дерева иерархии и выполнения команды «SHOWTREE»		
4	cl_2			Класс для чтения команд		
5	cl_3			Класс для выполнения арифметических операций		
6	cl_4			Класс для выполнения операции побитового сдвига		
7	cl_5			Класс для выполнения операции «C»		
8	cl_6			Класс для вывода очередного результата на консоль		

Класс `cl_base`:

- Поля/свойства:
  - Имя объекта:
    - Наименование - `name`
    - Тип данных - `string` (строковый)
    - Модификатор доступа - `private` (закрытый)
  - Указатель на родителя:
    - Наименование - `parent`
    - Тип данных - указатель на объект класса `cl_base`
    - Модификатор доступа - `private` (закрытый)
  - Вектор указателей на наследников:
    - Наименование - `children`
    - Тип данных - вектор указателей на объекты класса `cl_base`
    - Модификатор доступа - `private` (закрытый)
  - Глубина рекурсии:
    - Наименование - `num_rec`
    - Тип данных - `int` (целочисленный)
    - Модификатор доступа - `private` (закрытый)
  - Готовность объекта (состояние):
    - Наименование - `state`
    - Тип данных - `int` (целочисленный)
    - Модификатор доступа - `private` (закрытый)
  - Вектор указателей на соединения:
    - Наименование - `connections`
    - Тип данных - вектор указателей на объекты структуры `o_sh`
    - Модификатор доступа - `private` (закрытый)
  - Флаг для определения операции `C`:

- Наименование - flag\_C
  - Тип данных - bool (логический)
  - Модификатор доступа - protected
- Структура сигнала и обработчика:
  - Наименование - o\_sh
  - Тип данных - struct (структура)
  - Модификатор доступа - private (закрытый)
- Методы:
  - cl\_base(cl\_base \*parent, string name="")
    - Функционал - параметризованный конструктор
  - ~cl\_base()
    - Функционал - деструктор
  - get\_name()
    - Функционал - возврат имени
  - set\_name(string name)
    - Функционал - запись имени
  - get\_parent()
    - Функционал - возврат родителя
  - set\_parent(cl\_base \*parent)
    - Функционал - установка значения родителя
  - get\_ptr\_by\_name (string name)
    - Функционал - поиск указателя на объект по имени
  - get\_state()
    - Функционал - возврат готовности
  - set\_state(int state1)
    - Функционал - установка значения готовности
  - print\_state(int num\_rec2)

- Функционал - вывод иерархии с готовностью объектов
- set\_connection(TYPE\_SIGNAL signal, cl\_base\* target\_obj, TYPE\_HANDLER handler)
  - Функционал - установка связи
- delete\_connection (TYPE\_SIGNAL signal, cl\_base\* target\_obj, TYPE\_HANDLER handler)
  - Функционал - удаление связи
- emit\_signal(TYPE\_SIGNAL signal, string& msg)
  - Функционал - выдача сигнала от заданного по координате объекта

Класс cl\_application:

- Поля/свойства:
  - Флаг для ввода:
    - Наименование - flag\_input
    - Тип данных - bool (логический)
    - Модификатор доступа - private (закрытый)
- Методы:
  - cl\_application(cl\_base\*parent)
    - Функционал - параметризованный конструктор
  - build\_tree\_objects()
    - Функционал - построение дерева иерархии
  - exec\_app(bool b)
    - Функционал - начало работы калькулятора
  - signal\_reader(string &msg)
    - Функционал - сигнал для считывания данных
  - handler\_operation(string msg)
    - Функционал - обработчик сигнала для вывода иерархии



Класс cl\_1:

- Поля/свойства:
  - Отсутствуют
- Методы:
  - cl\_1(cl\_base \*parent, string name)
    - Функционал - параметризованный конструктор

Класс cl\_2:

- Поля/свойства:
  - Отсутствуют
- Методы:
  - cl\_2(cl\_base \*parent, string name)
    - Функционал - параметризованный конструктор
  - handler\_reader(string msg)
    - Функционал - обработчик сигнала для чтения данных
  - signal\_operation(string &msg)
    - Функционал - сигнал для выполнения операций

Класс cl\_3:

- Поля/свойства:
  - Текущее значение числа
    - Наименование - num\_1
    - Тип данных - int (целочисленный)
    - Модификатор доступа - private (закрытый)
  - Выражение, записанное в калькулятор
    - Наименование - result
    - Тип данных - string (строковый)
    - Модификатор доступа - private (закрытый)
  - Текущая операция

- Наименование - operation
- Тип данных - string (строковый)
- Модификатор доступа - private (закрытый)
- Методы:
  - cl\_3(cl\_base \*parent, string name)
    - Функционал - параметризованный конструктор
  - plus (int num\_1, int num\_2)
    - Функционал - сложение двух чисел
  - minus (int num\_1, int num\_2)
    - Функционал - разность двух чисел
  - umn (int num\_1, int num\_2)
    - Функционал - произведение двух чисел
  - del (int num\_1, int num\_2)
    - Функционал - целочисленное деление
  - ost (int num\_1, int num\_2)
    - Функционал - остаток от деления
  - handler\_operation(string msg)
    - Функционал - обработчик для выполнения операций
  - signal\_print(string &msg)
    - Функционал - сигнал для вывода
  - signal\_num\_to\_4(string &msg)
    - Функционал - сигнал для передачи числа в класс cl\_4
  - signal\_oper\_to\_4(string &msg)
    - Функционал - сигнал для передачи операции в класс cl\_4
  - signal\_res\_to\_4(string &msg)
    - Функционал - сигнал для передачи результата в класс cl\_4
  - handler\_num\_to\_3(string msg)

- Функционал - обработчик для переданного числа из класса cl\_4
- handler\_oper\_to\_3(string msg)
  - Функционал - обработчик для переданной операции из класса cl\_4
- handler\_res\_to\_3(string msg)
  - Функционал - обработчик для переданного результата из класса cl\_4

Класс cl\_4:

- Поля/свойства:
  - Текущее значение числа
    - Наименование - num\_1
    - Тип данных - int (целочисленный)
    - Модификатор доступа - private (закрытый)
  - Выражение, записанное в калькулятор
    - Наименование - result
    - Тип данных - string (строковый)
    - Модификатор доступа - private (закрытый)
  - Текущая операция
    - Наименование - operation
    - Тип данных - string (строковый)
    - Модификатор доступа - private (закрытый)
- Методы:
  - cl\_4(cl\_base \*parent, string name)
    - Функционал - параметризованный конструктор
  - sdv\_lev (int num\_1, int num\_2)
    - Функционал - побитовый сдвиг влево
  - sdv\_prav (int num\_1, int num\_2)

- Функционал - побитовый сдвиг вправо
- handler\_operation(string msg)
  - Функционал - обработчик для выполнения операций
- signal\_print(string &msg)
  - Функционал - сигнал для вывода
- signal\_num\_to\_3(string &msg)
  - Функционал - сигнал для передачи числа в класс cl\_3
- signal\_oper\_to\_3(string &msg)
  - Функционал - сигнал для передачи операции в класс cl\_3
- signal\_res\_to\_3(string &msg)
  - Функционал - сигнал для передачи результата в класс cl\_3
- handler\_num\_to\_4(string msg)
  - Функционал - обработчик для переданного числа из класса cl\_3
- handler\_oper\_to\_4(string msg)
  - Функционал - обработчик для переданной операции из класса cl\_3
- handler\_res\_to\_4(string msg)
  - Функционал - обработчик для переданного результата из класса cl\_3

Класс cl\_5:

- Поля/свойства:
  - Текущее значение числа
    - Наименование - num\_1
    - Тип данных - int (целочисленный)
    - Модификатор доступа - private (закрытый)
  - Выражение, записанное в калькулятор
    - Наименование - result

- Тип данных - string (строковый)
- Модификатор доступа - private (закрытый)
- Текущая операция
  - Наименование - operation
  - Тип данных - string (строковый)
  - Модификатор доступа - private (закрытый)
- Методы:
  - cl\_5(cl\_base \*parent, string name)
    - Функционал - параметризованный конструктор
  - handler\_operation(string msg)
    - Функционал - обработчик для выполнения операций
  - signal\_res\_5(string &msg)
    - Функционал - сигнал для передачи результата из класса cl\_5
  - signal\_num\_5(string &msg)
    - Функционал - сигнал для передачи числа из класса cl\_5
  - signal\_oper\_5(string &msg)
    - Функционал - сигнал для передачи операции из класса cl\_5

Класс cl\_6:

- Поля/свойства:
  - Флаг для вывода endl
    - Наименование - flagg
    - Тип данных - bool (логический)
    - Модификатор доступа - private (закрытый)
  - Текущее значение числа
    - Наименование - num\_1
    - Тип данных - int (целочисленный)
    - Модификатор доступа - private (закрытый)

- Методы:
  - `cl_6(cl_base *parent, string name)`
    - Функционал - параметризованный конструктор
  - `handler_print(string msg)`
    - Функционал - обработчик для вывода
  - `HEX_16(int number)`
    - Функционал - перевод для шестнадцатиричную систему счисления
  - `BIN_2(int number)`
    - Функционал - перевод в двоичную систему счисления
  - `signal_res_6(string &msg)`
    - Функционал - сигнал для передачи результата из класса `cl_6`
  - `signal_num_1_6(string &msg)`
    - Функционал - сигнал для передачи числа из класса `cl_6`

### 3 ОПИСАНИЕ АЛГОРИТМОВ

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

#### 3.0 Алгоритм функции main

Функционал: основная функция.

Параметры: Отсутствуют.

Возвращаемое значение: int.

Алгоритм функции представлен в таблице 2.

Таблица 2 – Алгоритм функции main

№	Предикат	Действия	№ перехода
1		Создание объекта cl_application_obj(nullptr) класса cl_application	2
2		Объявление логической переменной b	3
3		Присвоение переменной b результата вызова метода build_tree_objects для объекта cl_application_obj	4
4		Вызов метода exes_app с передачей параметра b	Ø

#### 3.1 Алгоритм конструктора класса cl\_1

Функционал: Вызов конструктора класса cl\_base.

Параметры: cl\_base \*parent, string name.

Алгоритм конструктора представлен в таблице 3.

Таблица 3 – Алгоритм конструктора класса cl\_1

№	Предикат	Действия	№ перехода
1		Вызов конструктора класса cl_base с передачей параметров cl_base *parent, string name	Ø

### 3.2 Алгоритм конструктора класса cl\_2

Функционал: Вызов конструктора класса cl\_base.

Параметры: cl\_base \*parent, string name.

Алгоритм конструктора представлен в таблице 4.

Таблица 4 – Алгоритм конструктора класса cl\_2

№	Предикат	Действия	№ перехода
1		Вызов конструктора класса cl_base с передачей параметров cl_base *parent, string name	∅

### 3.3 Алгоритм метода handler\_reader класса cl\_2

Функционал: Обработчик сигнала чтения данных.

Параметры: string msg.

Возвращаемое значение: Отсутствует.

Алгоритм метода представлен в таблице 5.

Таблица 5 – Алгоритм метода handler\_reader класса cl\_2

№	Предикат	Действия	№ перехода
1		Объявление скоровой переменной znak	2
2		Ввод znak	3
3		Вызов метода emit_signal с передачей параметров SIGNAL_D(cl_2::signal_operation), znak	∅

### 3.4 Алгоритм метода signal\_operation класса cl\_2

Функционал: Сигнал для выполнения операций.

Параметры: string &msg.

Возвращаемое значение: Отсутствует.

Алгоритм метода представлен в таблице 6.

Таблица 6 – Алгоритм метода signal\_operation класса cl\_2

№	Предикат	Действия	№ перехода
---	----------	----------	------------



№	Предикат	Действия	№ перехода
1			∅

### 3.5 Алгоритм конструктора класса cl\_3

Функционал: Вызов конструктора класса cl\_base.

Параметры: cl\_base \*parent, string name.

Алгоритм конструктора представлен в таблице 7.

Таблица 7 – Алгоритм конструктора класса cl\_3

№	Предикат	Действия	№ перехода
1		Вызов конструктора класса cl_base с передачей параметров cl_base *parent, string name	∅

### 3.6 Алгоритм метода plus класса cl\_3

Функционал: сложение двух чисел.

Параметры: int num\_1, int num\_2.

Возвращаемое значение: int.

Алгоритм метода представлен в таблице 8.

Таблица 8 – Алгоритм метода plus класса cl\_3

№	Предикат	Действия	№ перехода
1		Возврат num_1+num_2	∅

### 3.7 Алгоритм метода minus класса cl\_3

Функционал: разность двух чисел.

Параметры: int num\_1, int num\_2.

Возвращаемое значение: int.

Алгоритм метода представлен в таблице 9.

Таблица 9 – Алгоритм метода minus класса cl\_3

№	Предикат	Действия	№ перехода
1		Возврат num_1-num_2	Ø

### 3.8 Алгоритм метода umn класса cl\_3

Функционал: Произведение чисел.

Параметры: int num\_1, int num\_2.

Возвращаемое значение: int.

Алгоритм метода представлен в таблице 10.

Таблица 10 – Алгоритм метода umn класса cl\_3

№	Предикат	Действия	№ перехода
1		Возврат num_1*num_2	Ø

### 3.9 Алгоритм метода del класса cl\_3

Функционал: Целочисленное деление.

Параметры: int num\_1, int num\_2.

Возвращаемое значение: int.

Алгоритм метода представлен в таблице 11.

Таблица 11 – Алгоритм метода del класса cl\_3

№	Предикат	Действия	№ перехода
1		Возврат num_1/num_2	Ø

### 3.10 Алгоритм метода ost класса cl\_3

Функционал: Остаток от деления.

Параметры: int num\_1, int num\_2.

Возвращаемое значение: int.

Алгоритм метода представлен в таблице 12.

Таблица 12 – Алгоритм метода ost класса cl\_3

№	Предикат	Действия	№ перехода
1		Возврат num_1%num_2	∅

### 3.11 Алгоритм метода handler\_operation класса cl\_3

Функционал: Обработчик для выполнения операций.

Параметры: string msg.

Возвращаемое значение: Отсутствует.

Алгоритм метода представлен в таблице 13.

Таблица 13 – Алгоритм метода handler\_operation класса cl\_3

№	Предикат	Действия	№ перехода
1		Инициализация логической переменной tem = false	2
2	Значение msg не равно "Off" и значение msg не равно "C" и (значение msg равно "+" или значение msg равно "-" или значение msg равно "*" или значение msg равно "/" или значение msg равно "%") и значение msg не равно "SHOWTREE"	Присвоение operation=msg	3
			7
3		Присвоение result+=" "+operation	4
4		Вызов метода emit_signal с передачей параметров SIGNAL_D(cl_3::signal_res_to_4), result	5
5		Вызов метода emit_signal с передачей параметров SIGNAL_D(cl_3::signal_oper_to_4), operation	6
6		Присвоение flag_C=false	∅
7	Значение msg не равно "Off"		8

№	Предикат	Действия	№ перехода
	и значение msg не равно "C" и значение msg не равно "SHOWTREE"		
			∅
8	Значение operation не равно "" и значение operation не равно "<<" и значение operation не равно ">>"	Присвоение result+=" "+msg;	9
			55
9		Инициализация целочисленной переменной num_2=stoi(msg)	10
10	Значение operation равно "+"	Присвоение num_1 результата вызова метода plus с передачей параметров num_1, num_2	11
			11
11	Значение operation равно "+"	Присвоение num_1 результата вызова метода minus с передачей параметров num_1, num_2	12
			12
12	Значение operation равно "*"	Присвоение num_1 результата вызова метода umn с передачей параметров num_1, num_2	13
			13
13	Значение operation равно "/"		14
			25
14	Значение msg не равно "0"	Присвоение num_1 результата вызова метода del с передачей параметров num_1, num_2	25
			15
15		Присвоение result+=" Division by zero"	16
16		Вызов метода emit_signal с передачей параметров SIGNAL_D(cl_3::signal_print), result	17

№	Предикат	Действия	№ перехода
1 7		Присвоение num_1=0	18
1 8		Присвоение result="0"	19
1 9		Инициализация строковой переменной numb=to_string(num_1)	20
2 0		Вызов метода emit_signal с передачей параметров SIGNAL_D(cl_3::signal_num_to_4), numb	21
2 1		Присвоение tem=true	22
2 2		Вызов метода emit_signal с передачей параметров SIGNAL_D(cl_3::signal_res_to_4), result	23
2 3		Присвоение operation=""	24
2 4		Вызов метода emit_signal с передачей параметров SIGNAL_D(cl_3::signal_oper_to_4), operation	25
2 5	Значение operation равно "%"		26
			37
2 6	Значение msg не равно "0"	Присвоение num_1 результата вызова метода ost с передачей параметров num_1, num_2	37
			27
2 7		Присвоение result+=" Division by zero"	28
2 8		Вызов метода emit_signal с передачей параметров SIGNAL_D(cl_3::signal_print), result	29
2 9		Присвоение num_1=0	30
3 0		Присвоение result="0"	31

№	Предикат	Действия	№ перехода
3 1		Инициализация строковой переменной numb=to_string(num_1)	32
3 2		Вызов метода emit_signal с передачей параметров SIGNAL_D(cl_3::signal_num_to_4), numb	33
3 3		Присвоение tem=true	34
3 4		Вызов метода emit_signal с передачей параметров SIGNAL_D(cl_3::signal_res_to_4), result	35
3 5		Присвоение operation=""	36
3 6		Вызов метода emit_signal с передачей параметров SIGNAL_D(cl_3::signal_oper_to_4), operation	37
3 7	Значение tem равно false		38
			∅
3 8	Значение num_1 меньше или равно 32767 и значение num_1 больше или равно -32768	Присвоение operation=""	39
			45
3 9		Инициализация строковой переменной nu=to_string(num_1)	40
4 0		Вызов метода emit_signal с передачей параметров SIGNAL_D(cl_3::signal_num_to_4), nu	41
4 1		Инициализация строковой переменной prin=result+" "+to_string(num_1)	42
4 2		Вызов метода emit_signal с передачей параметров SIGNAL_D(cl_3::signal_print), prin	43
4 3		Вызов метода emit_signal с передачей параметров SIGNAL_D(cl_3::signal_res_to_4), result	44

№	Предикат	Действия	№ перехода
4		Вызов метода emit_signal с передачей параметров SIGNAL_D(cl_3::signal_oper_to_4), operation)	∅
4		Присвоение result+=" Overflow"	46
5			
4		Вызов метода emit_signal с передачей параметров SIGNAL_D(cl_3::signal_print), result	47
6			
4		Присвоение num_1=0	48
7			
4		Присвоение result="0"	49
8			
4		Инициализация строковой переменной numb=to_string(num_1)	50
9			
5		Вызов метода emit_signal с передачей параметров SIGNAL_D(cl_3::signal_num_to_4), numb	51
0			
5		Присвоение tem=true	52
1			
5		Вызов метода emit_signal с передачей параметров SIGNAL_D(cl_3::signal_res_to_4), result	53
2			
5		Присвоение operation=""	54
3			
5		Вызов метода emit_signal с передачей параметров SIGNAL_D(cl_3::signal_oper_to_4), operation	∅
4			
5	Значение msg не равно "<<" и значение msg не равно ">>" и значение operation не равно "<<" и значение operation не равно ">>" и значение msg не равно "SHOWTREE"	Присвоение num_1=stoi(msg)	56

№	Предикат	Действия	№ перехода
			∅
5 6	Значение result не равно "" и значение operation не равно ""	Присвоение result+=" "+to_string(num_1)	60
			57
5 7	Значение flag_C не равно false	Присвоение result+=msg	60
			58
5 8		Присвоение result=msg	59
5 9		Присвоение flag_C=false	60
6 0		Инициализация строковой переменной nu=to_string(num_1)	61
6 1		Вызов метода emit_signal с передачей параметров SIGNAL_D(cl_3::signal_num_to_4), nu	62
6 2		Вызов метода emit_signal с передачей параметров SIGNAL_D(cl_3::signal_res_to_4), result	63
6 3		Вызов метода emit_signal с передачей параметров SIGNAL_D(cl_3::signal_oper_to_4), operation	∅

### 3.12 Алгоритм метода signal\_print класса cl\_3

Функционал: Сигнал для вывода.

Параметры: string &msg.

Возвращаемое значение: Отсутствует.

Алгоритм метода представлен в таблице 14.



Таблица 14 – Алгоритм метода `signal_print` класса `cl_3`

№	Предикат	Действия	№ перехода
1			∅

### 3.13 Алгоритм метода `signal_num_to_4` класса `cl_3`

Функционал: Сигнал для передачи числа в класс `cl_4`.

Параметры: `string &msg`.

Возвращаемое значение: Отсутствует.

Алгоритм метода представлен в таблице 15.

Таблица 15 – Алгоритм метода `signal_num_to_4` класса `cl_3`

№	Предикат	Действия	№ перехода
1			∅

### 3.14 Алгоритм метода `signal_oper_to_4` класса `cl_3`

Функционал: Сигнал для передачи операции в класс `cl_4`.

Параметры: `string &msg`.

Возвращаемое значение: Отсутствует.

Алгоритм метода представлен в таблице 16.

Таблица 16 – Алгоритм метода `signal_oper_to_4` класса `cl_3`

№	Предикат	Действия	№ перехода
1			∅

### 3.15 Алгоритм метода `signal_res_to_4` класса `cl_3`

Функционал: Сигнал для передачи результата в класс `cl_4`.

Параметры: `string &msg`.

Возвращаемое значение: Отсутствует.

Алгоритм метода представлен в таблице 17.

Таблица 17 – Алгоритм метода signal\_res\_to\_4 класса cl\_3

№	Предикат	Действия	№ перехода
1			∅

### 3.16 Алгоритм метода handler\_num\_to\_3 класса cl\_3

Функционал: Обработчик для переданного числа из класса cl\_4.

Параметры: string msg.

Возвращаемое значение: Отсутствует.

Алгоритм метода представлен в таблице 18.

Таблица 18 – Алгоритм метода handler\_num\_to\_3 класса cl\_3

№	Предикат	Действия	№ перехода
1		Присвоение num_1=stoi(msg)	∅

### 3.17 Алгоритм метода handler\_oper\_to\_3 класса cl\_3

Функционал: Обработчик для переданной операции из класса cl\_4.

Параметры: string msg.

Возвращаемое значение: Отсутствует.

Алгоритм метода представлен в таблице 19.

Таблица 19 – Алгоритм метода handler\_oper\_to\_3 класса cl\_3

№	Предикат	Действия	№ перехода
1		Присвоение operation=msg	∅

### 3.18 Алгоритм метода handler\_res\_to\_3 класса cl\_3

Функционал: Обработчик для переданного результата из класса cl\_4.

Параметры: string msg.

Возвращаемое значение: Отсутствует.

Алгоритм метода представлен в таблице 20.

Таблица 20 – Алгоритм метода handler\_res\_to\_3 класса cl\_3

№	Предикат	Действия	№ перехода
1		Присвоение result=msg	∅

### 3.19 Алгоритм конструктора класса cl\_4

Функционал: Вызов конструктора класса cl\_base.

Параметры: cl\_base \*parent, string name.

Алгоритм конструктора представлен в таблице 21.

Таблица 21 – Алгоритм конструктора класса cl\_4

№	Предикат	Действия	№ перехода
1		Вызов конструктора класса cl_base с передачей параметров cl_base *parent, string name	∅

### 3.20 Алгоритм метода sdv\_lev класса cl\_4

Функционал: Побитовый сдвиг влево.

Параметры: int num\_1, int num\_2.

Возвращаемое значение: int.

Алгоритм метода представлен в таблице 22.

Таблица 22 – Алгоритм метода sdv\_lev класса cl\_4

№	Предикат	Действия	№ перехода
1		Инициализация целочисленной переменной num_num=num_1	2
2	Значение num_2 больше 0	Присвоение num_num*=2	3
		Возврат num_num	∅
3		Значение num_2--	2

### 3.21 Алгоритм метода sdv\_prav класса cl\_4

Функционал: Побитовый сдвиг вправо.

Параметры: int num\_1, int num\_2.

Возвращаемое значение: int.

Алгоритм метода представлен в таблице 23.

Таблица 23 – Алгоритм метода `sdv_prav` класса `cl_4`

№	Предикат	Действия	№ перехода
1		Инициализация целочисленной переменной <code>num_num=num_1</code>	2
2	Значение <code>num_2</code> больше 0	Присвоение <code>num_num/=2</code>	3
		Возврат <code>num_num</code>	∅
3		Значение <code>num_2--</code>	2

### 3.22 Алгоритм метода `handler_operation` класса `cl_4`

Функционал: Обработчик для выполнения операций.

Параметры: `string msg`.

Возвращаемое значение: Отсутствует.

Алгоритм метода представлен в таблице 24.

Таблица 24 – Алгоритм метода `handler_operation` класса `cl_4`

№	Предикат	Действия	№ перехода
1	Значение <code>msg</code> не равно "Off" и значение <code>msg</code> не равно "C" и (значение <code>msg</code> равно "<<" или значение <code>msg</code> равно ">>")и значение <code>msg</code> не равно "SHOWTREE"	Присвоение <code>operation=msg</code>	2
			5
2		Присвоение <code>result+=" "+operation</code>	3
3		Вызов метода <code>emit_signal</code> с передачей параметров <code>SIGNAL_D(cl_4::signal_res_to_3), result</code>	4
4		Вызов метода <code>emit_signal</code> с передачей параметров <code>SIGNAL_D(cl_4::signal_oper_to_3), operation</code>	∅

№	Предикат	Действия	№ перехода
5	Значение msg не равно "Off" и значение msg не равно "C" и (значение msg не равно "+" и значение msg не равно "-" и значение msg не равно "*" и значение msg не равно "/" и значение msg не равно "%"		6
			∅
6	Значение operation не равно ""	Присвоение result+=" "+msg	7
			∅
7	Значение operation равно "<<"	Присвоение num_1 результата вызова метода sdv_lev с передачей параметров num_1, stoi(msg)	8
			8
8	Значение operation равно ">>"	Присвоение num_1 результата вызова метода sdv_prav с передачей параметров num_1, stoi(msg)	9
			9
9	Значение num_1 меньше или равно 32767 и значение num_1 больше или равно - 32768	Присвоение operation=""	10
			15
10		Инициализация строковой переменной nu=to_string(num_1)	11
11		Вызов метода emit_signal с передачей параметров SIGNAL_D(cl_4::signal_num_to_3), nu	12
12		Вызов метода emit_signal с передачей параметров SIGNAL_D(cl_4::signal_res_to_3), result	13

№	Предикат	Действия	№ перехода
1 3		Вызов метода emit_signal передачей параметров SIGNAL_D(cl_4::signal_oper_to_3), operation	14
1 4		Инициализация строковой переменной prin=result +" "+ to_string(num_1)	∅
1 5		Присвоение result+=" Overflow"	16
1 6		Вызов метод emit_signal с передачей параметров SIGNAL_D(cl_4::signal_print), result	17
1 7		Присвоение num_1=0	18
1 8		Присвоение result="0"	19
1 9		Инициализация строковой переменной numb=to_string(num_1)	20
2 0		Вызов метода emit_signal с передачей параметров SIGNAL_D(cl_4::signal_num_to_3), numb	21
2 1		Вызов метода emit_signal с передачей параметров SIGNAL_D(cl_4::signal_res_to_3), result	22
2 2		Присвоение operation=""	23
2 3		Вызов метода emit_signal с передачей параметров SIGNAL_D(cl_4::signal_oper_to_3), operation	∅

### 3.23 Алгоритм метода signal\_print класса cl\_4

Функционал: Сигнал для вывода.

Параметры: string &msg.

Возвращаемое значение: Отсутствует.

Алгоритм метода представлен в таблице 25.

Таблица 25 – Алгоритм метода `signal_print` класса `cl_4`

№	Предикат	Действия	№ перехода
1			∅

### 3.24 Алгоритм метода `signal_num_to_3` класса `cl_4`

Функционал: Сигнал для передачи числа в класс `cl_3`.

Параметры: `string &msg`.

Возвращаемое значение: Отсутствует.

Алгоритм метода представлен в таблице 26.

Таблица 26 – Алгоритм метода `signal_num_to_3` класса `cl_4`

№	Предикат	Действия	№ перехода
1			∅

### 3.25 Алгоритм метода `signal_oper_to_3` класса `cl_4`

Функционал: Сигнал для передачи операции в класс `cl_3`.

Параметры: `string &msg`.

Возвращаемое значение: Отсутствует.

Алгоритм метода представлен в таблице 27.

Таблица 27 – Алгоритм метода `signal_oper_to_3` класса `cl_4`

№	Предикат	Действия	№ перехода
1			∅

### 3.26 Алгоритм метода `signal_res_to_3` класса `cl_4`

Функционал: Сигнал для передачи результата в класс `cl_3`.

Параметры: `string &msg`.

Возвращаемое значение: Отсутствует.

Алгоритм метода представлен в таблице 28.

Таблица 28 – Алгоритм метода signal\_res\_to\_3 класса cl\_4

№	Предикат	Действия	№ перехода
1			∅

### 3.27 Алгоритм метода handler\_num\_to\_4 класса cl\_4

Функционал: Обработчик для переданного числа из класса cl\_3.

Параметры: string msg.

Возвращаемое значение: Отсутствует.

Алгоритм метода представлен в таблице 29.

Таблица 29 – Алгоритм метода handler\_num\_to\_4 класса cl\_4

№	Предикат	Действия	№ перехода
1		Присвоение num_1=stoi(msg)	∅

### 3.28 Алгоритм метода handler\_oper\_to\_4 класса cl\_4

Функционал: Обработчик для переданной операции из класса cl\_3.

Параметры: string msg.

Возвращаемое значение: Отсутствует.

Алгоритм метода представлен в таблице 30.

Таблица 30 – Алгоритм метода handler\_oper\_to\_4 класса cl\_4

№	Предикат	Действия	№ перехода
1		Присвоение operation=msg	∅

### 3.29 Алгоритм метода handler\_res\_to\_4 класса cl\_4

Функционал: Обработчик для переданного результата из класса cl\_3.

Параметры: string msg.

Возвращаемое значение: Отсутствует.

Алгоритм метода представлен в таблице 31.



Таблица 31 – Алгоритм метода handler\_res\_to\_4 класса cl\_4

№	Предикат	Действия	№ перехода
1		Присвоение result=msg	∅

### 3.30 Алгоритм конструктора класса cl\_5

Функционал: Вызов конструктора класса cl\_base.

Параметры: cl\_base \*parent, string name.

Алгоритм конструктора представлен в таблице 32.

Таблица 32 – Алгоритм конструктора класса cl\_5

№	Предикат	Действия	№ перехода
1		Вызов конструктора класса cl_base с передачей параметров cl_base *parent, string name	∅

### 3.31 Алгоритм метода handler\_operation класса cl\_5

Функционал: Обработчик для выполнения операций.

Параметры: string msg.

Возвращаемое значение: Отсутствует.

Алгоритм метода представлен в таблице 33.

Таблица 33 – Алгоритм метода handler\_operation класса cl\_5

№	Предикат	Действия	№ перехода
1	Значение msg равно "C" и значение msg не равно "SHOWTREE"	Присвоение result=""	2
			∅
2		Присвоение num_1="0"	3
3		Присвоение operation = ""	4
4		Вызов метода emit_signal с передачей параметров SIGNAL_D(cl_5::signal_oper_5), operation	5

№	Предикат	Действия	№ перехода
5		Вызов метода emit_signal с передачей параметров SIGNAL_D(cl_5::signal_num_5), num_1	6
6		Вызов метода emit_signal с передачей параметров SIGNAL_D(cl_5::signal_res_5), result	7
7		Присвоение flag_C=true	∅

### 3.32 Алгоритм метода signal\_res\_5 класса cl\_5

Функционал: Сигнал для передачи результата из класса cl\_5.

Параметры: string &msg.

Возвращаемое значение: Отсутствует.

Алгоритм метода представлен в таблице 34.

Таблица 34 – Алгоритм метода signal\_res\_5 класса cl\_5

№	Предикат	Действия	№ перехода
1			∅

### 3.33 Алгоритм метода signal\_num\_5 класса cl\_5

Функционал: Сигнал для передачи числа из класса cl\_5.

Параметры: string &msg.

Возвращаемое значение: Отсутствует.

Алгоритм метода представлен в таблице 35.

Таблица 35 – Алгоритм метода signal\_num\_5 класса cl\_5

№	Предикат	Действия	№ перехода
1			∅

### 3.34 Алгоритм метода `signal_oper_5` класса `cl_5`

Функционал: Сигнал для передачи операции из класса `cl_5`.

Параметры: `string &msg`.

Возвращаемое значение: Отсутствует.

Алгоритм метода представлен в таблице 36.

Таблица 36 – Алгоритм метода `signal_oper_5` класса `cl_5`

№	Предикат	Действия	№ перехода
1			Ø

### 3.35 Алгоритм конструктора класса `cl_6`

Функционал: Вызов конструктора класса `cl_base`.

Параметры: `cl_base *parent`, `string name`.

Алгоритм конструктора представлен в таблице 37.

Таблица 37 – Алгоритм конструктора класса `cl_6`

№	Предикат	Действия	№ перехода
1		Вызов конструктора класса <code>cl_base</code> с передачей параметров <code>cl_base *parent</code> , <code>string name</code>	Ø

### 3.36 Алгоритм метода `handler_print` класса `cl_6`

Функционал: Обработчик для вывода.

Параметры: `string msg`.

Возвращаемое значение: Отсутствует.

Алгоритм метода представлен в таблице 38.

Таблица 38 – Алгоритм метода `handler_print` класса `cl_6`

№	Предикат	Действия	№ перехода
1	Значение <code>msg.size()</code> больше 8		2
			25

№	Предикат	Действия	№ перехода
2	Значение msg.substr(msg.size()-8) равно "Overflow"		3
			11
3	Значение flagg равно true	Вывод endl	4
			4
4		Вывод msg	5
5		Присвоение flagg=true	6
6		Инициализация строковой переменной result=""	7
7		Присвоение num_1=0	8
8		Инициализация строковой переменной nu=to_string(num_1)	9
9		Вызов метода emit_signal с передачей параметров SIGNAL_D(cl_6::signal_res_6), result	10
10		Вызов метода emit_signal с передачей параметров SIGNAL_D(cl_6::signal_num_1_6), nu	∅
11	Значение msg.substr(msg.size()-4) равно "zero"		12
			20
12	Значение flagg равно true	Вывод endl	13
			13
13		Вывод msg	14
14		Присвоение flagg=true	15
15		Инициализация строковой переменной result=""	16
16		Присвоение num_1=0	17

№	Предикат	Действия	№ перехода
6			
1 7		Инициализация строковой переменной nu=to_string(num_1)	18
1 8		Вызов метода emit_signal с передачей параметров SIGNAL_D(cl_6::signal_res_6), result	19
1 9		Вызов метода emit_signal с передачей параметров SIGNAL_D(cl_6::signal_num_1_6), nu	∅
2 0		Инициализация целочисленной переменной ind=msg.rfind(" ")	21
2 1		Инициализация целочисленной переменной num_print=stoi(msg.substr(ind+1))	22
2	Значение flagg равно true	Вывод endl	23
2			23
2 3		Вывод msg.substr(0, ind) и " " и "HEX " и результата вызова метода HEX_16 с передачей параметра num_print и " " и "DEC " и num_print и " " и "BIN " и результат вызова метода BIN_2 с передачей параметра num_print	24
2 4		Присвоение flagg=true	∅
2 5		Инициализация целочисленной переменной ind=msg.rfind(" ")	26
2 6		Инициализация целочисленной переменной num_print=stoi(msg.substr(ind+1))	27
2	Значение flagg равно true	Вывод endl	28
7			28
2 8		Вывод msg.substr(0, ind) и " " и "HEX " и результата вызова метода HEX_16 с передачей параметра num_print и " " и "DEC " и num_print и "	29

№	Предикат	Действия	№ перехода
		" и "BIN " и результат вызова метода BIN_2 с передачей параметра num_print	
2 9		Присвоение flagg=true	∅

### 3.37 Алгоритм метода HEX\_16 класса cl\_6

Функционал: Перевод для шестнадцатиричную систему счисления.

Параметры: int number.

Возвращаемое значение: string.

Алгоритм метода представлен в таблице 39.

Таблица 39 – Алгоритм метода HEX\_16 класса cl\_6

№	Предикат	Действия	№ перехода
1		Инициализация логической переменной tem=true	2
2	Значение number меньше 0	tem=false	3
			3
3	Значение tem равно true	Инициализация строковой переменнй hex_num1=""	4
			15
4		Объявление строковой переменной hex_number	5
5		Инициализация целочисленной переменной num=abs(number)	6
6	Значение num больше 0	Инициализация целочисленной переменной ost=num%16	7
			9
7	Значение ost меньше или равно 9	Присвоение hex_num1+=(ost+'0')	8
		Присвоение hex_num1+='A'+(ost-10)	8
8		Присвоение num/=16	6
9	Значение hex_num1.size()	Привоеение hex_num1+="0"	9

№	Предикат	Действия	№ перехода
	меньше 4		
			10
1 0	Значение счетчика меньше (hex_num1.size())/2	Объявление переменной типа char temp	11
			14
1 1		Присвоение temp = hex_num1[i]	12
1 2		Присвоение hex_num1[i]=hex_num1[hex_num1.size()-i-1]	13
1 3		Присвоение hex_num1[hex_num1.size()-i-1]=temp	10
1 4		Возврат hex_num1	∅
1 5		Инициализация целочисленной переменной num_h=0	16
1 6		Инициализация строковой переменной hex=""	17
1 7		Инициализация строковой переменной numb=BIN_2(number)	18
1 8		Вызов метода строк erase(14, 1) для numb	19
1 9		Вызов метода строк erase(9, 1) для numb	20
2 0		Вызов метода строк erase(4, 1) для numb	21
2 1	Счетчик цикла меньше numb.size()-1		22
			23
2	Значение numb[i] равно '1'	Присвоение num_h+=pow(2, numb.size()-i-1)	21

№	Предикат	Действия	№ перехода
2			21
2 3	Значение num_h больше 0	Инициализация целочисленной переменной ost=num_h%16	24
			26
2 4	Значение ost больше или равно 9	Присвоение hex+=(ost+'0')	25
		Присвоение hex+='A'+(ost-10)	25
2 5		Присвоение num_h/=16	23
2 6	Значение hex.size() меньше 4	Присвоение hex+="0"	26
			27
2 7	Значение счетчика меньше (hex.size())/2	Объявление переменной типа char temp	28
			31
2 8		Присвоение temp = hex[i]	29
2 9		Присвоение hex[i]=hex[hex.size()-i-1]	30
3 0		Присвоение hex[hex.size()-i-1]=temp	27
3 1		Возврат hex	∅

### 3.38 Алгоритм метода BIN\_2 класса cl\_6

Функционал: Перевод в двоичную систему счисления.

Параметры: int number.

Возвращаемое значение: string.

Алгоритм метода представлен в таблице 40.



Таблица 40 – Алгоритм метода BIN\_2 класса cl\_6

№	Предикат	Действия	№ перехода
1		Инициализация логической переменной temp=true	2
2	Значение number меньше 0	Присвоение temp=false	3
			3
3		Инициализация целочисленных переменных bin_number=0, k=1, num=abs(number)	4
4		Инициализация строковой переменной num_BIN	5
5	Значение num больше 0	Присвоение num_BIN+=to_string(num%2)	6
			7
6		Присвоение num/=2	5
7	Значение num_BIN.size() меньше 16	Присвоение num_BIN+="0"	7
			8
8		Вызов метода insert(4, 1, ' ') для строки num_BIN	9
9		Вызов метода insert(9, 1, ' ') для строки num_BIN	10
10		Вызов метода insert(14, 1, ' ') для строки num_BIN	11
11	Счетчик цикла меньше (num_BIN.size())/2	Объявление переменной типа char temp	12
			15
12		Присвоение temp =num_BIN[i]	13
13		Присвоение num_BIN[i]=num_BIN[num_BIN.size()- i-1]	14
14		Присвоение num_BIN[num_BIN.size()-i-1]=temp	11
15	Значение temp равно true	Возврат num_BIN	∅
5		Инициализация целочисленной переменной ind_1=num_BIN.rfind("1")	16

№	Предикат	Действия	№ перехода
1	Значение ind_1 не равно -1		17
6		Возврат num_BIN	∅
1	Счетчик цикла меньше ind_1		18
7		Возврат num_BIN	∅
1	Значение num_BIN[i] равно	Присвоение num_BIN[i]='0'	17
8	'1'		
		Присвоение num_BIN[i]='1'	17

### 3.39 Алгоритм метода signal\_res\_6 класса cl\_6

Функционал: Сигнал для передачи результата из класса cl\_6.

Параметры: string &msg.

Возвращаемое значение: Отсутствует.

Алгоритм метода представлен в таблице 41.

Таблица 41 – Алгоритм метода signal\_res\_6 класса cl\_6

№	Предикат	Действия	№ перехода
1			∅

### 3.40 Алгоритм метода signal\_num\_1\_6 класса cl\_6

Функционал: Сигнал для передачи числа из класса cl\_6.

Параметры: string &msg.

Возвращаемое значение: Отсутствует.

Алгоритм метода представлен в таблице 42.

Таблица 42 – Алгоритм метода signal\_num\_1\_6 класса cl\_6

№	Предикат	Действия	№ перехода
1			∅

### 3.41 Алгоритм конструктора класса cl\_application

Функционал: Вызов конструктора базового класса.

Параметры: cl\_base\*parent.

Алгоритм конструктора представлен в таблице 43.

Таблица 43 – Алгоритм конструктора класса cl\_application

№	Предикат	Действия	№ перехода
1		Вызов конструктора класса cl_base с передачей параметра cl_base*parent	∅

### 3.42 Алгоритм метода build\_tree\_objects класса cl\_application

Функционал: Построение дерева иерархии.

Параметры: .

Возвращаемое значение: bool.

Алгоритм метода представлен в таблице 44.

Таблица 44 – Алгоритм метода build\_tree\_objects класса cl\_application

№	Предикат	Действия	№ перехода
1		Объявление указателей на объекты класса cl_base *parent_ptr, *child_ptr_1, *child_ptr_2, *child_ptr_3, *child_ptr_4, *child_ptr_5	2
2		Объявление строковых переменных parent_name, child_name, name_cin, name_1	3
3		Присвоение parent_name="SYSTEM"	4
4		Вызов метода set_name с передачей параметра parent_name	5
5		Присвоение parent_ptr=this	6
6		Вызов метода set_state с передачей параметра 1 для parent_ptr	7
7		child_ptr_1 присваивается конструктор класса cl_2 с передачей параметров parent_ptr, "READER"	8
8		Вызов метода set_state с передачей параметра 1 для child_ptr_1	9
9		child_ptr_2 присваивается конструктор класса cl_3 с передачей параметров parent_ptr, "OPERATION"	10
10		Вызов метода set_state с передачей параметра 1 для child_ptr_2	11

№	Предикат	Действия	№ перехода
1 1		child_ptr_3 присваивается конструктор класса cl_4 с передачей параметров parent_ptr, "BIN_SDV"	12
1 2		Вызов метода set_state с передачей параметра 1 для child_ptr_3	13
1 3		child_ptr_4 присваивается конструктор класса cl_5 с передачей параметров parent_ptr, "COM_C"	14
1 4		Вызов метода set_state с передачей параметра 1 для child_ptr_4	15
1 5		child_ptr_5 присваивается конструктор класса cl_6 с передачей параметров parent_ptr, "PRINT"	16
1 6		Вызов метода set_state с передачей параметра 1 для child_ptr_5	17
1 7		Вызов метода set_connection с передачей параметров SIGNAL_D(cl_application::signal_reader), child_ptr_1, HANDLER_D(cl_2::handler_reader)	18
1 8		Вызов метода set_connection с передачей параметров SIGNAL_D(cl_2::signal_operation), child_ptr_2, HANDLER_D(cl_3::handler_operation) для child_ptr_1	19
1 9		Вызов метода set_connection с передачей параметров SIGNAL_D(cl_2::signal_operation), child_ptr_3, HANDLER_D(cl_4::handler_operation) для child_ptr_1	20
2 0		Вызов метода set_connection с передачей параметров SIGNAL_D(cl_2::signal_operation), child_ptr_4, HANDLER_D(cl_5::handler_operation) для child_ptr_1	21
2 1		Вызов метода set_connection с передачей параметров SIGNAL_D(cl_2::signal_operation), this, HANDLER_D(cl_application::handler_operation) для child_ptr_1	22
2 2		Вызов метода set_connection с передачей параметров SIGNAL_D(cl_3::signal_num_to_4), child_ptr_3,	23

№	Предикат	Действия	№ перехода
		HANDLER_D(cl_4::handler_num_to_4) для child_ptr_2	
2 3		Вызов метода set_connection с передачей параметров SIGNAL_D(cl_4::signal_num_to_3), child_ptr_2, HANDLER_D(cl_3::handler_num_to_3) для child_ptr_3	24
2 4		Вызов метода set_connection с передачей параметров SIGNAL_D(cl_3::signal_oper_to_4), child_ptr_3, HANDLER_D(cl_4::handler_oper_to_4) для child_ptr_2	25
2 5		Вызов метода set_connection с передачей параметров SIGNAL_D(cl_4::signal_oper_to_3), child_ptr_2, HANDLER_D(cl_3::handler_oper_to_3) для child_ptr_3	26
2 6		Вызов метода set_connection с передачей параметров SIGNAL_D(cl_3::signal_res_to_4), child_ptr_3, HANDLER_D(cl_4::handler_res_to_4) для child_ptr_2	27
2 7		Вызов метода set_connection с передачей параметров SIGNAL_D(cl_4::signal_res_to_3), child_ptr_2, HANDLER_D(cl_3::handler_res_to_3) для child_ptr_3	28
2 8		Вызов метода set_connection с передачей параметров SIGNAL_D(cl_3::signal_print), child_ptr_5, HANDLER_D(cl_6::handler_print) для child_ptr_2	29
2 9		Вызов метода set_connection с передачей параметров SIGNAL_D(cl_4::signal_print), child_ptr_5, HANDLER_D(cl_6::handler_print) для child_ptr_3	30
3 0		Вызов метода set_connection с передачей параметров SIGNAL_D(cl_5::signal_res_5), child_ptr_2, HANDLER_D(cl_3::handler_res_to_3) для child_ptr_4	31
3 1		Вызов метода set_connection с передачей параметров SIGNAL_D(cl_5::signal_res_5), child_ptr_3, HANDLER_D(cl_4::handler_res_to_4) для child_ptr_4	32
3		Вызов метода set_connection с передачей параметров	33

№	Предикат	Действия	№ перехода
2		SIGNAL_D(cl_5::signal_num_5), child_ptr_2, HANDLER_D(cl_3::handler_num_to_3) для child_ptr_4	
3		Вызов метода set_connection я с передачей параметров	34
3		SIGNAL_D(cl_5::signal_num_5), child_ptr_3, HANDLER_D(cl_4::handler_num_to_4) для child_ptr_4	
3		Вызов метода set_connection с передачей параметров	35
4		SIGNAL_D(cl_5::signal_oper_5), child_ptr_2, HANDLER_D(cl_3::handler_oper_to_3) для child_ptr_4	
3		Вызов метода set_connection с передачей параметров	36
5		SIGNAL_D(cl_5::signal_oper_5), child_ptr_3, HANDLER_D(cl_4::handler_oper_to_4) для child_ptr_4	
3		Вызов метода set_connection	37
6		с передачей параметров SIGNAL_D(cl_6::signal_res_6), child_ptr_2, HANDLER_D(cl_3::handler_res_to_3) для child_ptr_5	
3		Вызов метода set_connection с передачей параметров	38
7		SIGNAL_D(cl_6::signal_res_6), child_ptr_3, HANDLER_D(cl_4::handler_res_to_4) для child_ptr_5	
3		Вызов метода set_connection с передачей параметров	39
8		SIGNAL_D(cl_6::signal_num_1_6), child_ptr_2, HANDLER_D(cl_3::handler_num_to_3) для child_ptr_5	
3		Вызов метода set_connection с передачей параметров	40
9		SIGNAL_D(cl_6::signal_num_1_6), child_ptr_3, HANDLER_D(cl_4::handler_num_to_4) для child_ptr_5	
4		Возврат true	∅
0			

### 3.43 Алгоритм метода exes\_app класса cl\_application

Функционал: Начало работы калькулятора.

Параметры: bool b.

Возвращаемое значение: int.

Алгоритм метода представлен в таблице 45.

Таблица 45 – Алгоритм метода `exes_app` класса `cl_application`

№	Предикат	Действия	№ перехода
1		Инициализация строковой переменной <code>ms=""</code>	2
2		Присвоение <code>flag_input=true</code>	3
3	Значение <code>flag_input</code> истинно	Вызов метода <code>emit_signal</code> с передачей параметров <code>SIGNAL_D(cl_application::signal_reader), ms</code>	3
		Возврат 0	∅

### 3.44 Алгоритм метода `signal_reader` класса `cl_application`

Функционал: Сигнал для считывания данных.

Параметры: `string &msg`.

Возвращаемое значение: Отсутствует.

Алгоритм метода представлен в таблице 46.

Таблица 46 – Алгоритм метода `signal_reader` класса `cl_application`

№	Предикат	Действия	№ перехода
1			∅

### 3.45 Алгоритм метода `handler_operation` класса `cl_application`

Функционал: Обработчик сигнала для вывода иерархии.

Параметры: `string msg`.

Возвращаемое значение: Отсутствует.

Алгоритм метода представлен в таблице 47.

Таблица 47 – Алгоритм метода `handler_operation` класса `cl_application`

№	Предикат	Действия	№ перехода
1	Значение <code>msg</code> равно "Overflow"	Присвоение <code>flag_input=false</code>	∅
			2

№	Предикат	Действия	№ перехода
2	Значение msg равно "SHOWTREE"	Вывод "Object tree"	3
			7
3		Присвоение flag_input=false	4
4		Вывод endl и "SYSTEM"	5
5	Значение get_ptr_by_name("SYSTEM") ->get_state() равно 1	Вывод " is ready"	6
		Вывод " is not ready"	6
6		Вызов метода print_state с передачей параметра 1	∅
7	Значение msg равно "Off"	Присвоение flag_input=false	∅
			∅

### 3.46 Алгоритм конструктора класса cl\_base

Функционал: Добавление объекта в вектор.

Параметры: cl\_base \*parent, string name="".

Алгоритм конструктора представлен в таблице 48.

Таблица 48 – Алгоритм конструктора класса cl\_base

№	Предикат	Действия	№ перехода
1		Текущий name=name	2
2		Текущий parent = parent	3
3	Значение parent не равно nullptr	Вызв метода children.push_back с передачей параметра this для parent	∅
			∅

### 3.47 Алгоритм деструктора класса cl\_base

Функционал: Удаление вектора.

Параметры: .

Алгоритм деструктора представлен в таблице 49.



Таблица 49 – Алгоритм деструктора класса cl\_base

№	Предикат	Действия	№ перехода
1	Счетчик цикла меньше children.size()	Удаление children[i]	1
			2
2	Счетчик цикла меньше connections.size()	Удаление connections[i]	2
			Ø

### 3.48 Алгоритм метода get\_name класса cl\_base

Функционал: Возврат имени.

Параметры: .

Возвращаемое значение: string.

Алгоритм метода представлен в таблице 50.

Таблица 50 – Алгоритм метода get\_name класса cl\_base

№	Предикат	Действия	№ перехода
1		Вызврат name	Ø

### 3.49 Алгоритм метода set\_name класса cl\_base

Функционал: Заполнение поля name.

Параметры: string name.

Возвращаемое значение: Отсутствует.

Алгоритм метода представлен в таблице 51.

Таблица 51 – Алгоритм метода set\_name класса cl\_base

№	Предикат	Действия	№ перехода
1		Присвоение name=name1	Ø

### 3.50 Алгоритм метода get\_parent класса cl\_base

Функционал: Возврат указателя на родителя.

Параметры: .

Возвращаемое значение: cl\_base\*.

Алгоритм метода представлен в таблице 52.

Таблица 52 – Алгоритм метода get\_parent класса cl\_base

№	Предикат	Действия	№ перехода
1		Возврат parent	∅

### 3.51 Алгоритм метода set\_parent класса cl\_base

Функционал: Установка родительского объекта.

Параметры: cl\_base \*new\_parent.

Возвращаемое значение: Отсутствует.

Алгоритм метода представлен в таблице 53.

Таблица 53 – Алгоритм метода set\_parent класса cl\_base

№	Предикат	Действия	№ перехода
1	Значение parent не равно nullptr		2
			5
2	Значение счетчика цикла меньше размера вектра children для parent		3
			5
3	Значение parent->children[i] равно this	Вызов children.erase(parent->children.begin()+i) для parent	4
			2
4		break	2
5		Присвоение parent = new_parent	6
6	Значение parent не равно nullptr	Вызов children.push_back(this) для parent	∅

№	Предикат	Действия	№ перехода
	nullptr		
			∅

### 3.52 Алгоритм метода get\_ptr\_by\_name класса cl\_base

Функционал: Нахождение указателя по имени объекта.

Параметры: string name.

Возвращаемое значение: cl\_base\*.

Алгоритм метода представлен в таблице 54.

Таблица 54 – Алгоритм метода get\_ptr\_by\_name класса cl\_base

№	Предикат	Действия	№ перехода
1	Значение get_name() для this равно name	Возврат this	∅
			2
2	Счетчик цикла меньше размера вектора children для this	Инициализация указателя на объект класса cl_base buff = children[i]->get_ptr_by_name(name)	3
		Возврат nullptr	∅
3	Значение buff не равно nullptr	Возврат buff	2
			2

### 3.53 Алгоритм метода get\_state класса cl\_base

Функционал: Возврат готовности.

Параметры: .

Возвращаемое значение: int.

Алгоритм метода представлен в таблице 55.

Таблица 55 – Алгоритм метода get\_state класса cl\_base

№	Предикат	Действия	№ перехода
1		Возврат state	∅

### 3.54 Алгоритм метода set\_state класса cl\_base

Функционал: Установка готовности.

Параметры: int state1.

Возвращаемое значение: Отсутствует.

Алгоритм метода представлен в таблице 56.

Таблица 56 – Алгоритм метода set\_state класса cl\_base

№	Предикат	Действия	№ перехода
1	Значение parent не равно nullptr		2
			4
2	Значение get_state() для parent не равно 0 и значение state1 не равно 0	Присвоение state=state1	∅
		Присвоение state=0	3
3	Счетчик цикла меньше размера вектора children для this	Вызов метода set_state с передачей параметра 0 для children[i]	∅
			∅
4		Присвоение state=state1	5
5	Значение state равно 0		6
			∅
6	Счетчик цикла меньше размера вектор children для this	Вызов метода set_state с передачей параметра 0 для children[i]	∅
			∅

### 3.55 Алгоритм метода print\_state класса cl\_base

Функционал: Вывод дерева иерархии.

Параметры: int num\_rec2.

Возвращаемое значение: Отсутствует.

Алгоритм метода представлен в таблице 57.

Таблица 57 – Алгоритм метода print\_state класса cl\_base

№	Предикат	Действия	№ перехода
1		Присвоение num_rec=num_rec2	2
2	Значение размера вектора children не равно 0		3
			∅
3	Счетчик цикла меньше размера вектора children	Инициализация строковой переменной len=children[i]->get_name()	4
			∅
4	Значение get_state() для children[i] равно 0	Присвоение len+=" is not ready"	5
		Присвоение len+=" is ready"	5
5		Вывод endl и setw(len.size()+4*num_rec) и len	6
6		Значение num_rec++	7
7		Вызов метода print_state с передачей параметра num_rec для children[i]	8
8		Значение num_rec--	∅

### 3.56 Алгоритм метода set\_connection класса cl\_base

Функционал: Установка связи.

Параметры: TYPE\_SIGNAL signal, cl\_base\* target\_obj, TYPE\_HANDLER handler.

Возвращаемое значение: Отсутствует.

Алгоритм метода представлен в таблице 58.

Таблица 58 – Алгоритм метода set\_connection класса cl\_base

№	Предикат	Действия	№ перехода
1		Инициализация указателя на структуру connection=new o_sh(signal, target_obj, handler)	2
2		Добавление в вектор connections элемента connection	∅

### 3.57 Алгоритм метода delete\_connection класса cl\_base

Функционал: Удаление связи.

Параметры: TYPE\_SIGNAL signal, cl\_base\* target\_obj, TYPE\_HANDLER handler.

Возвращаемое значение: Отсутствует.

Алгоритм метода представлен в таблице 59.

Таблица 59 – Алгоритм метода delete\_connection класса cl\_base

№	Предикат	Действия	№ перехода
1	Значение счетчика меньше размера вектора connections		2
			∅
2	Значение (connections[i]->signal) равно signal и (connections[i])->target_obj равно target_obj	Удаление элемента с индексом счетчика	1
			1

### 3.58 Алгоритм метода emit\_signal класса cl\_base

Функционал: выдать сигнал от заданного по координате объекта.

Параметры: TYPE\_SIGNAL signal, string& msg.

Возвращаемое значение: Отсутствует.

Алгоритм метода представлен в таблице 60.

Таблица 60 – Алгоритм метода emit\_signal класса cl\_base

№	Предикат	Действия	№ перехода
1		Объявление указателя на объект класса cl_base target_obj_ptr	2
2		Объявление переменной типа TYPE_HANDLER handler_obj	3
3	Значение метода get_state() для this не равно 0	Вызов (this->*signal)(msg)	4
			∅
4	Счетчик цикла меньше размера вектора connections		5
			∅
5	Значение connections[i]- >signal равно signal	Присвоение target_obj_ptr=connections[i]-> target_obj	6
			∅
6		Присвоение handler_obj= connections[i]-> handler	7
7	Значение target_obj_ptr- >get_state() истинно	Вызов (target_obj_ptr->*handler_obj)(msg)	4
			4

## 4 БЛОК-СХЕМЫ АЛГОРИТМОВ

Представим описание алгоритмов в графическом виде на рисунках 1-52.

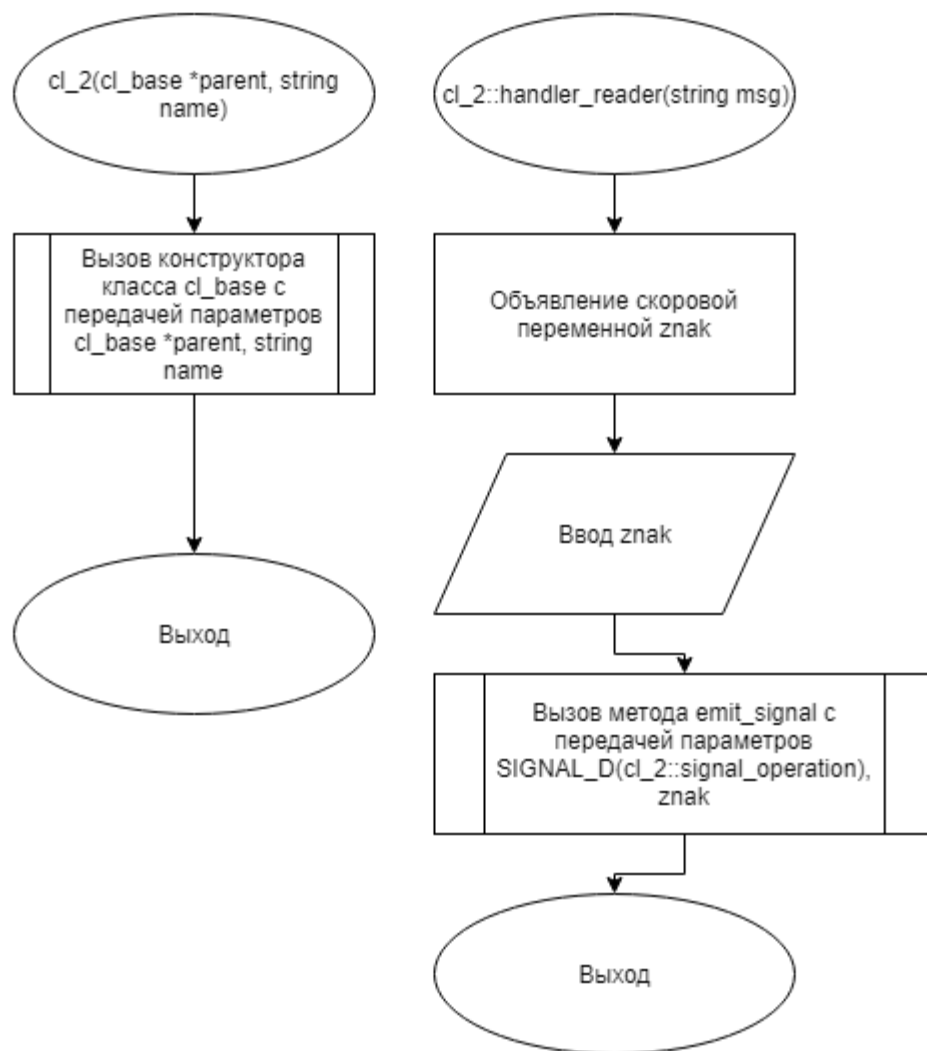
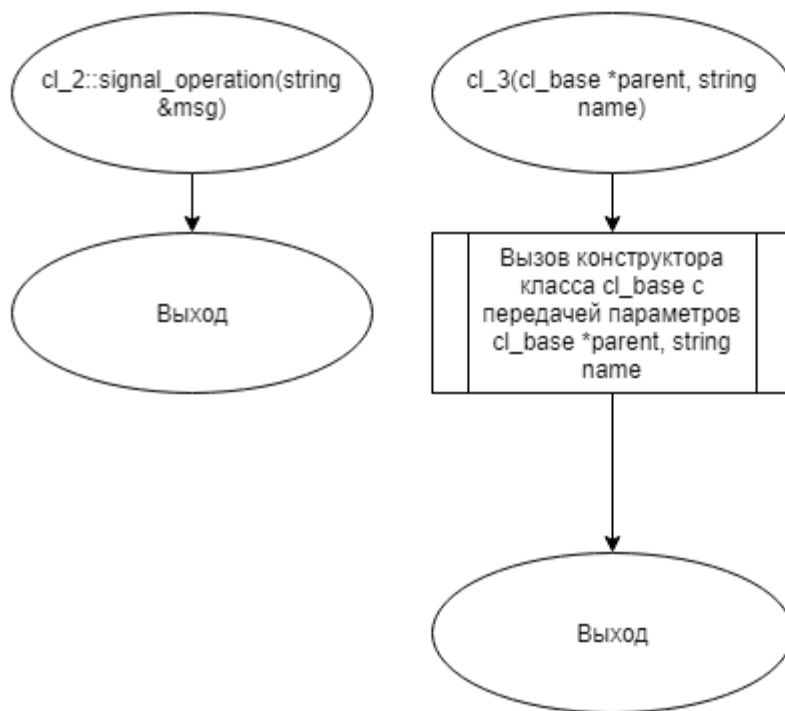
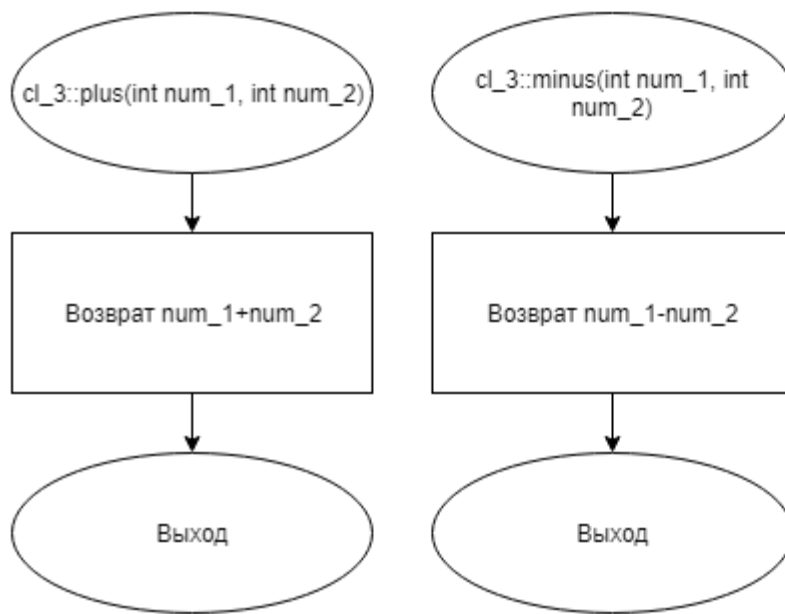


Рисунок 1 – Блок-схема алгоритма

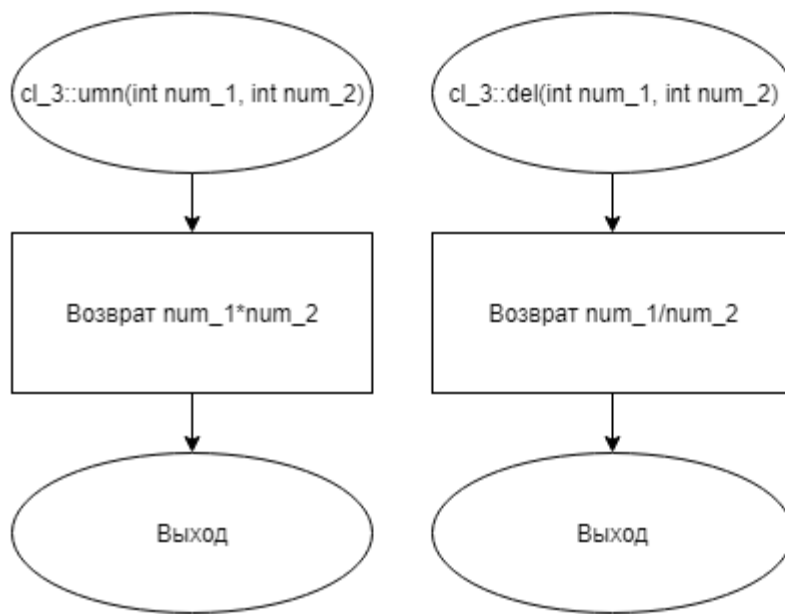




**Рисунок 2 – Блок-схема алгоритма**



**Рисунок 3 – Блок-схема алгоритма**



**Рисунок 4 – Блок-схема алгоритма**

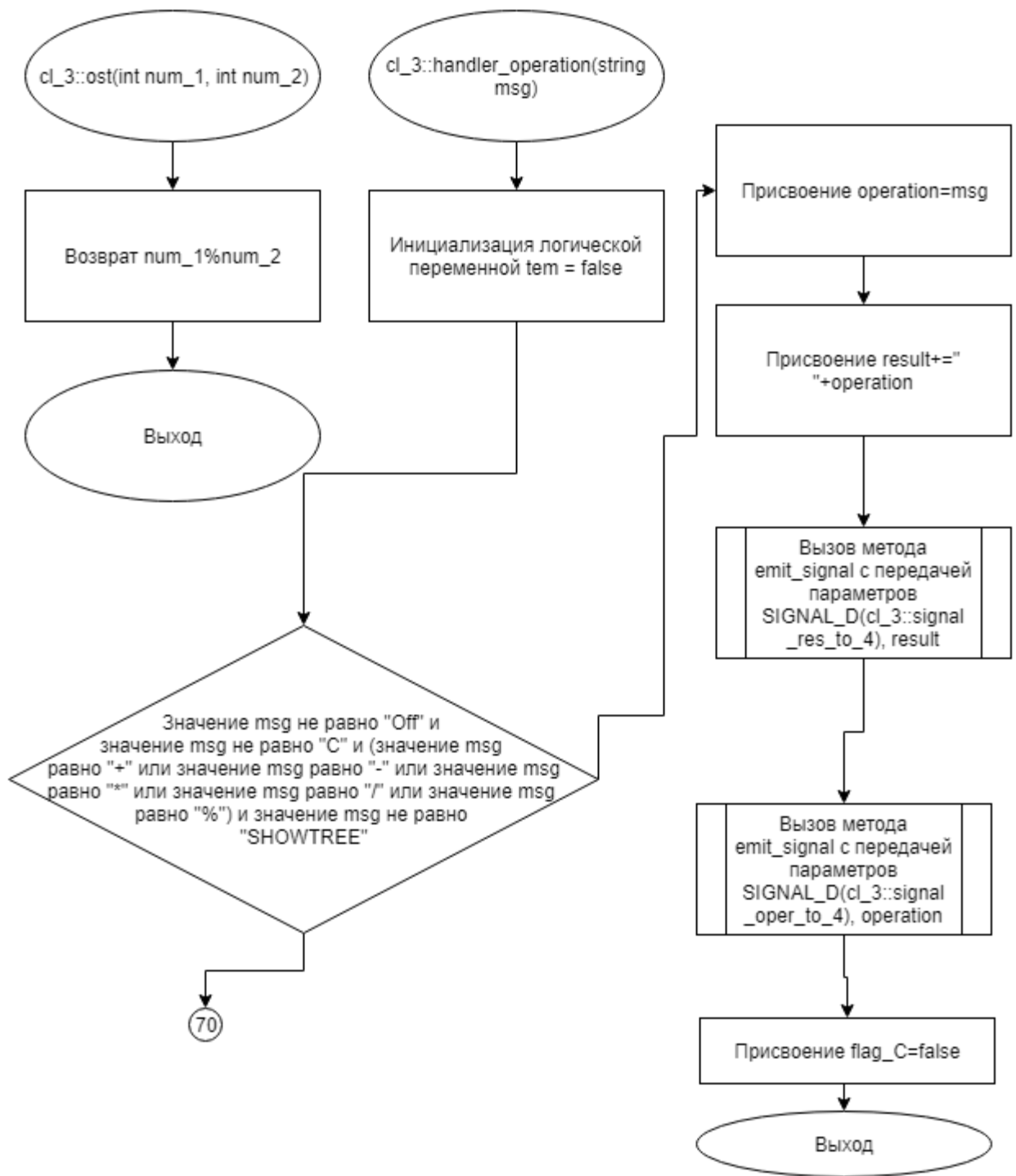
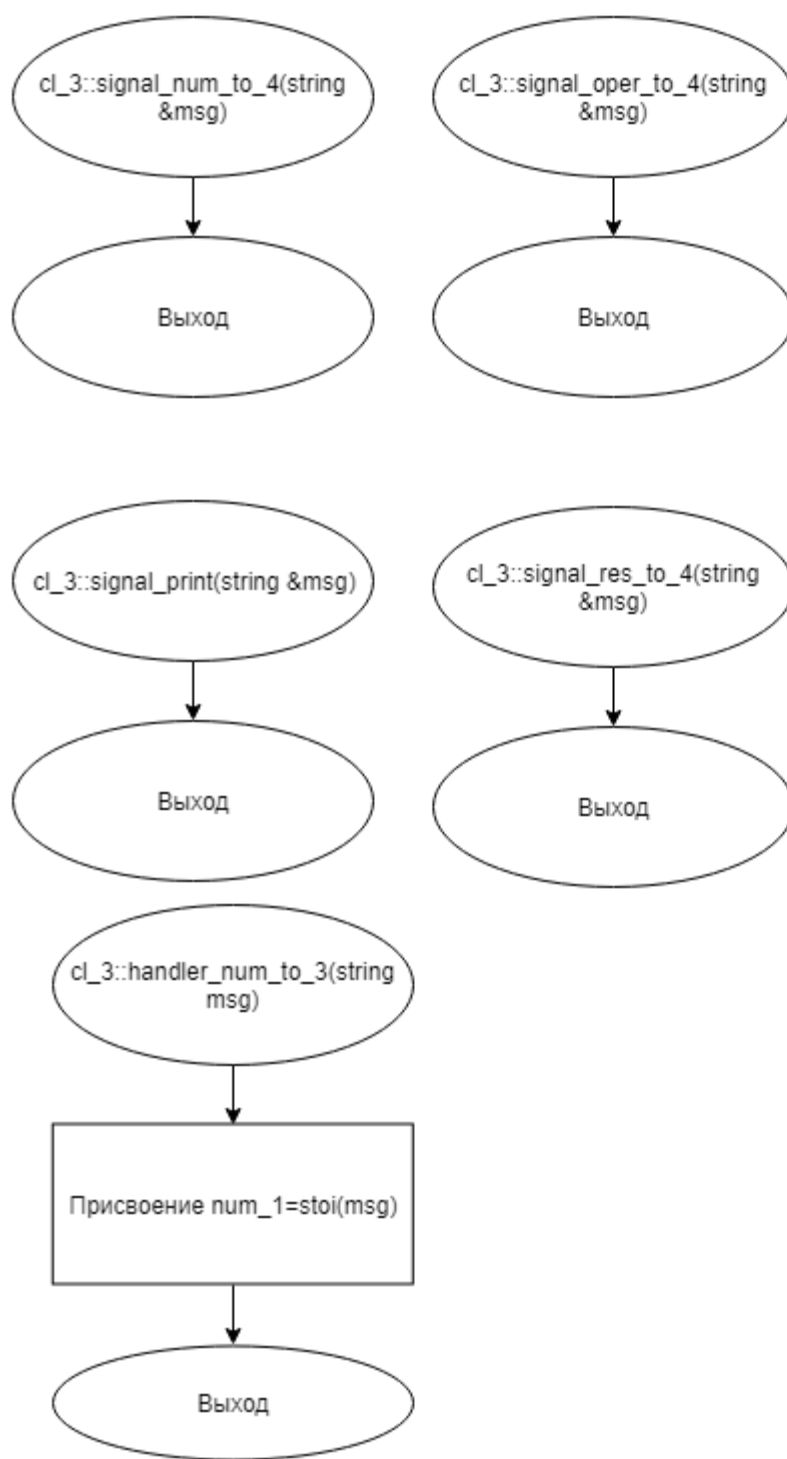
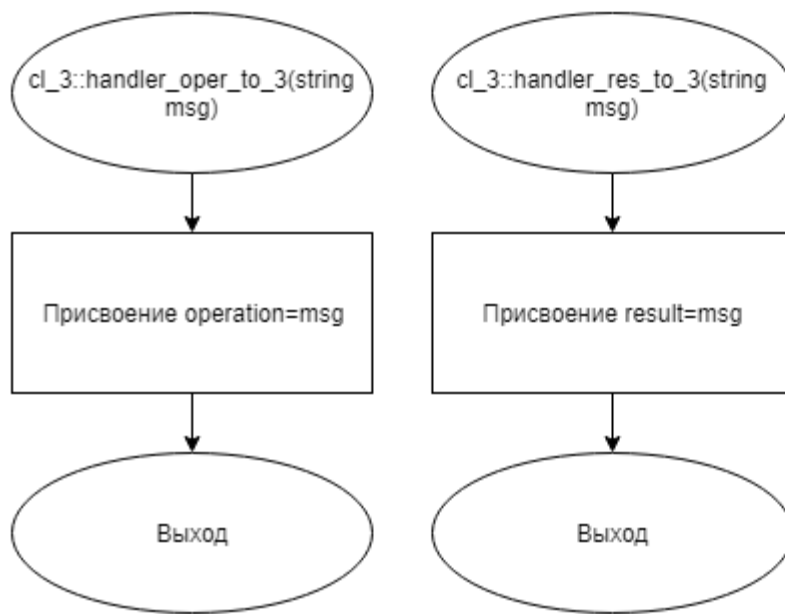


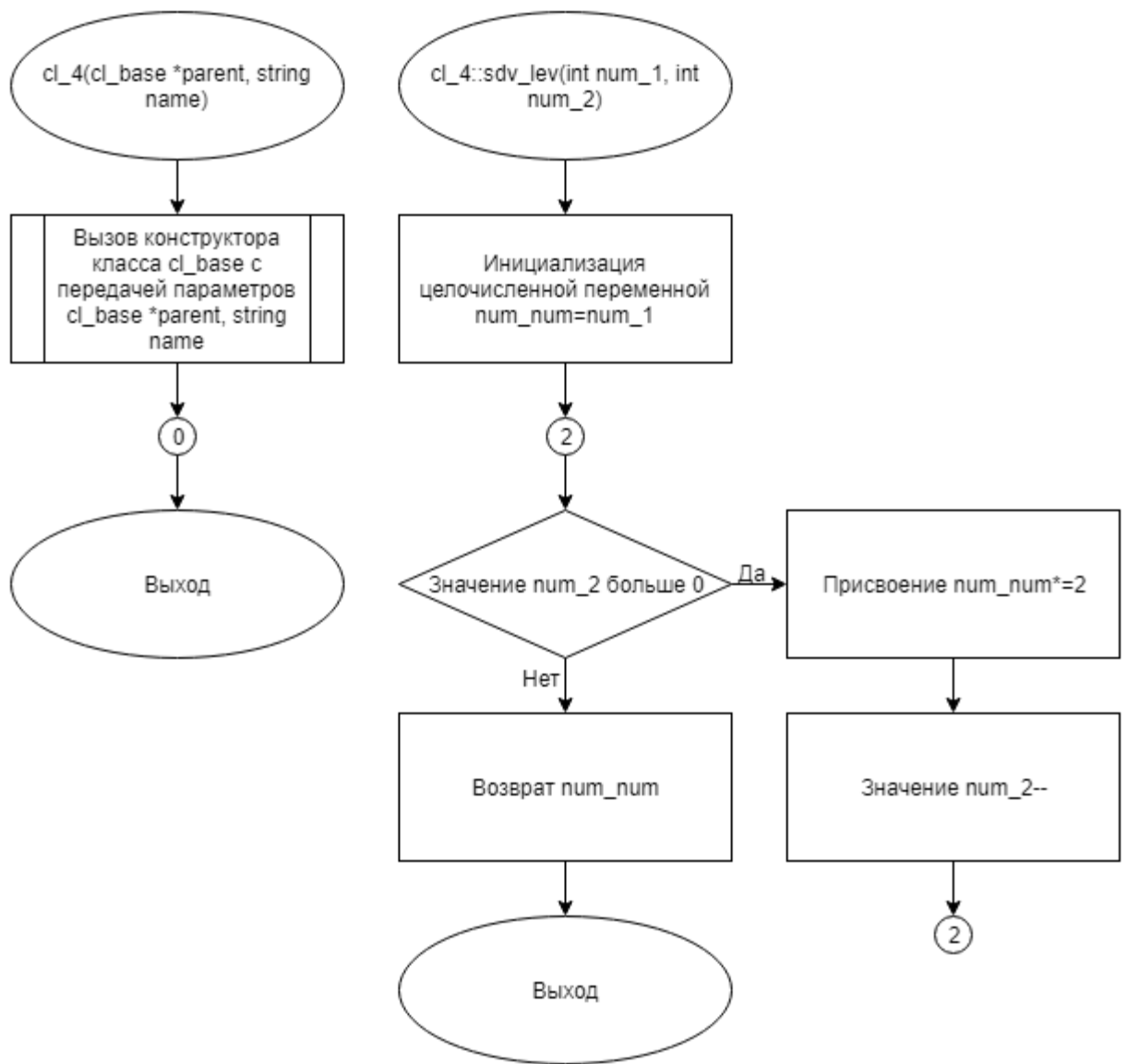
Рисунок 5 – Блок-схема алгоритма



**Рисунок 6 – Блок-схема алгоритма**



**Рисунок 7 – Блок-схема алгоритма**



**Рисунок 8 – Блок-схема алгоритма**

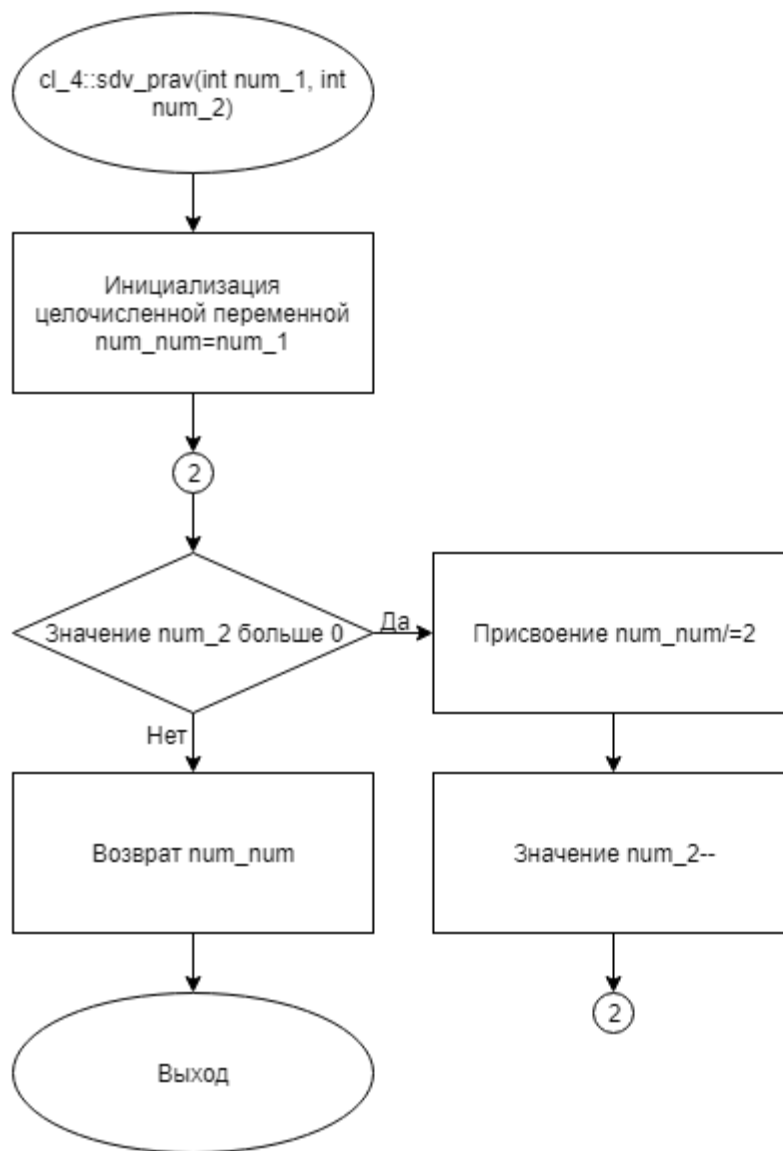


Рисунок 9 – Блок-схема алгоритма



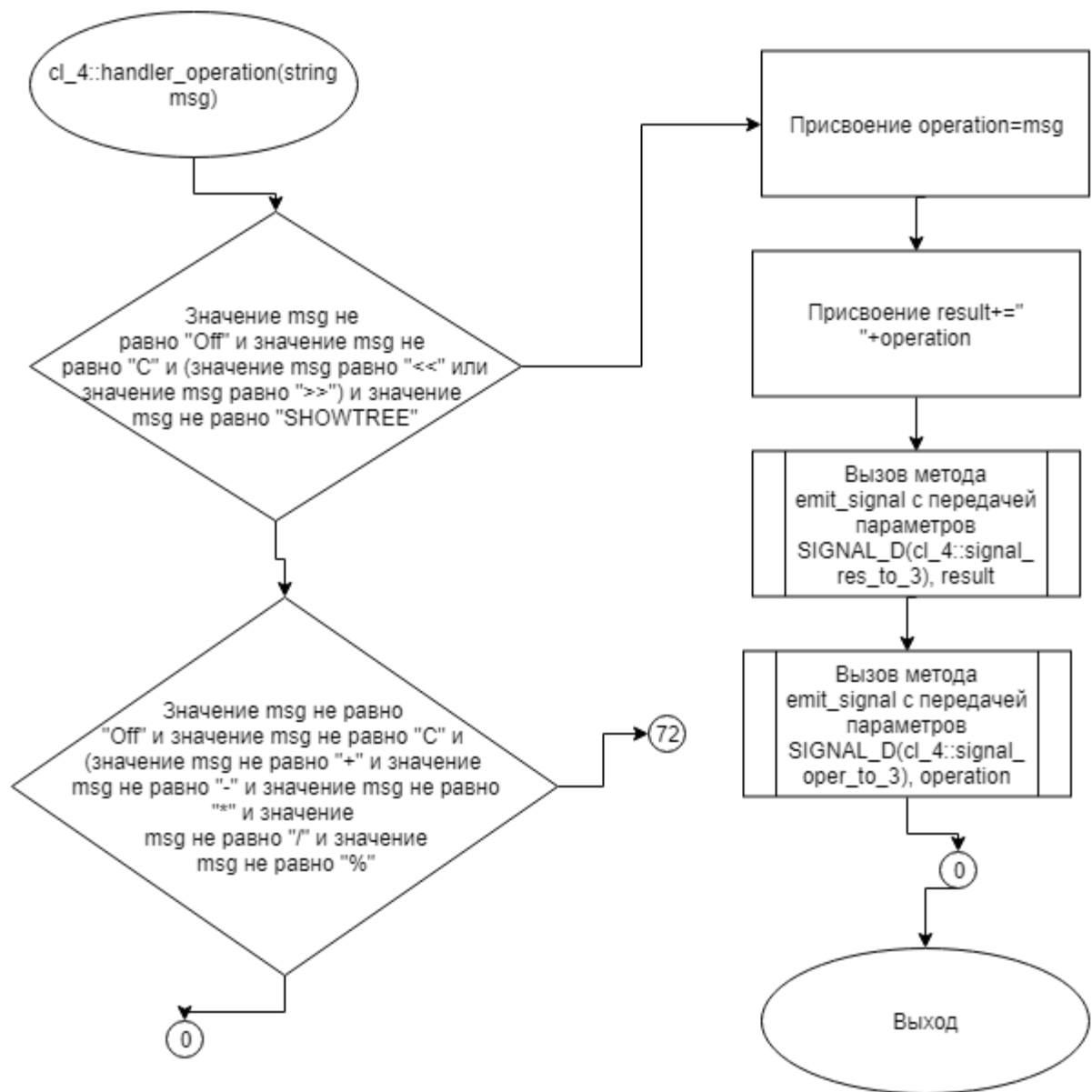
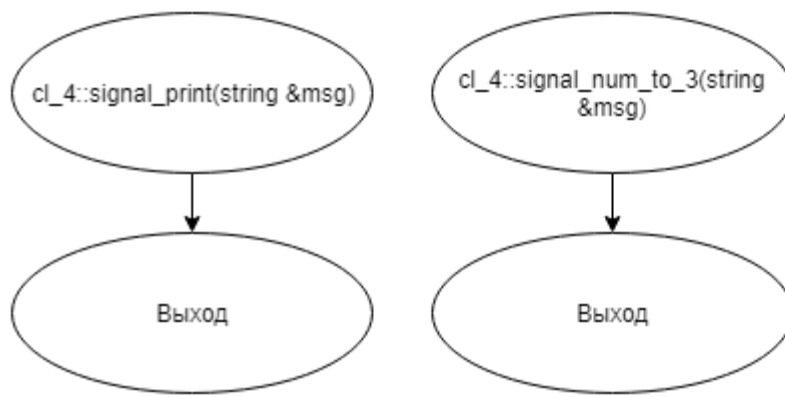


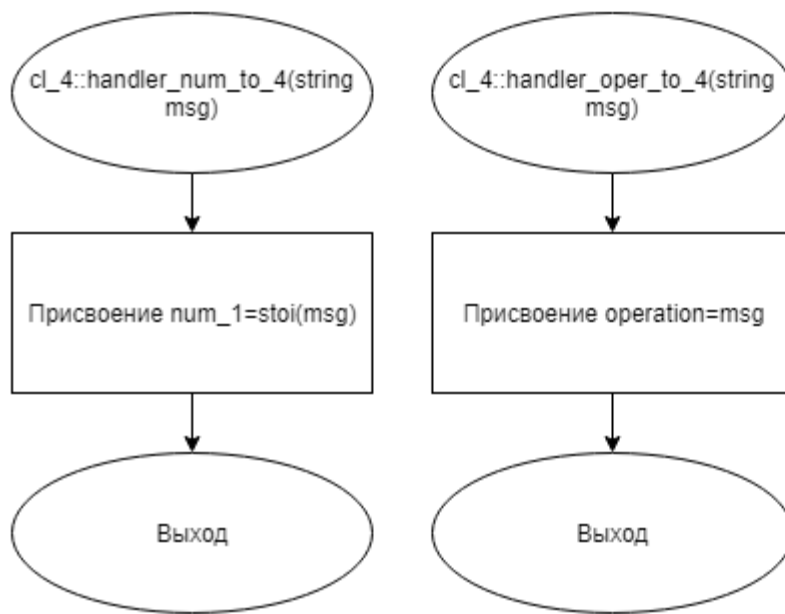
Рисунок 10 – Блок-схема алгоритма



**Рисунок 11 – Блок-схема алгоритма**



**Рисунок 12 – Блок-схема алгоритма**



**Рисунок 13 – Блок-схема алгоритма**

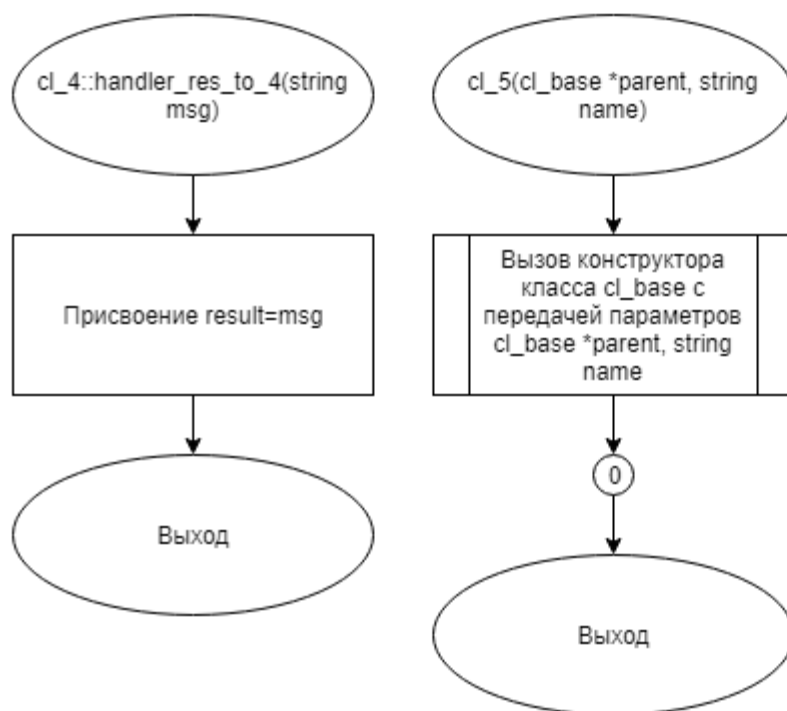


Рисунок 14 – Блок-схема алгоритма

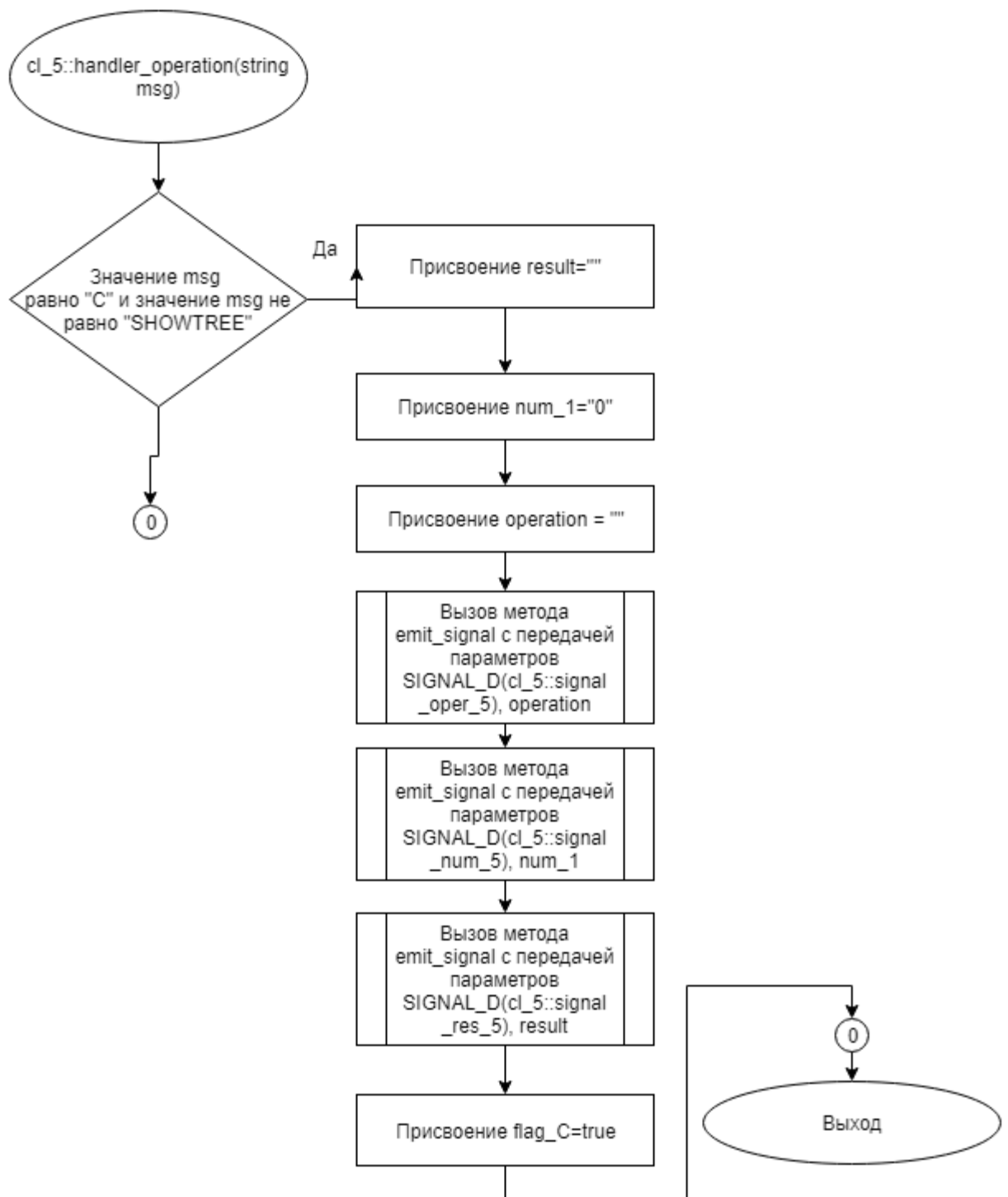
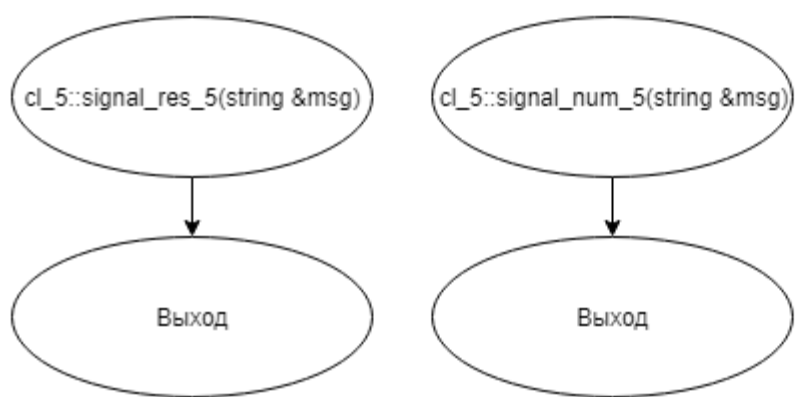
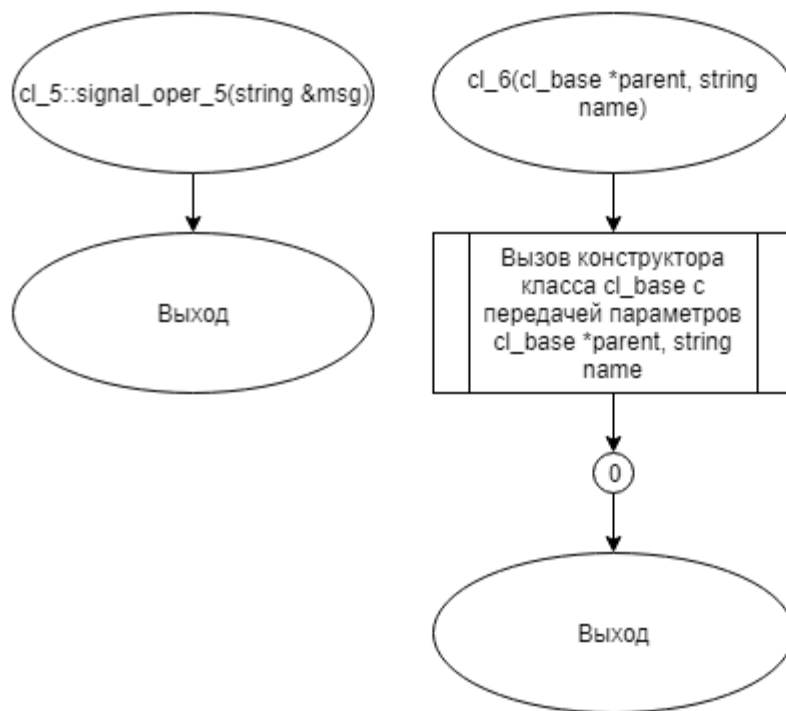


Рисунок 15 – Блок-схема алгоритма



**Рисунок 16 – Блок-схема алгоритма**



**Рисунок 17 – Блок-схема алгоритма**



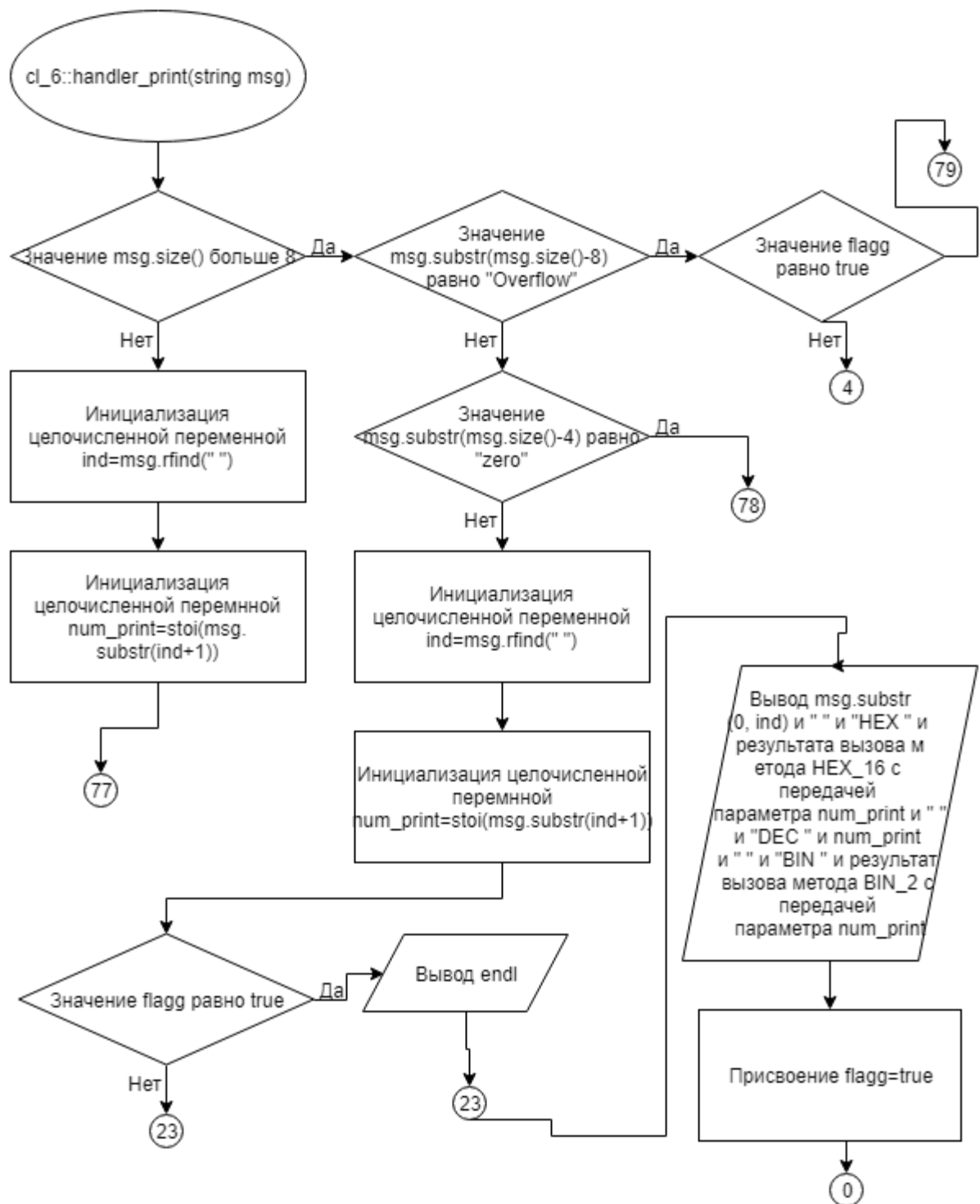


Рисунок 18 – Блок-схема алгоритма

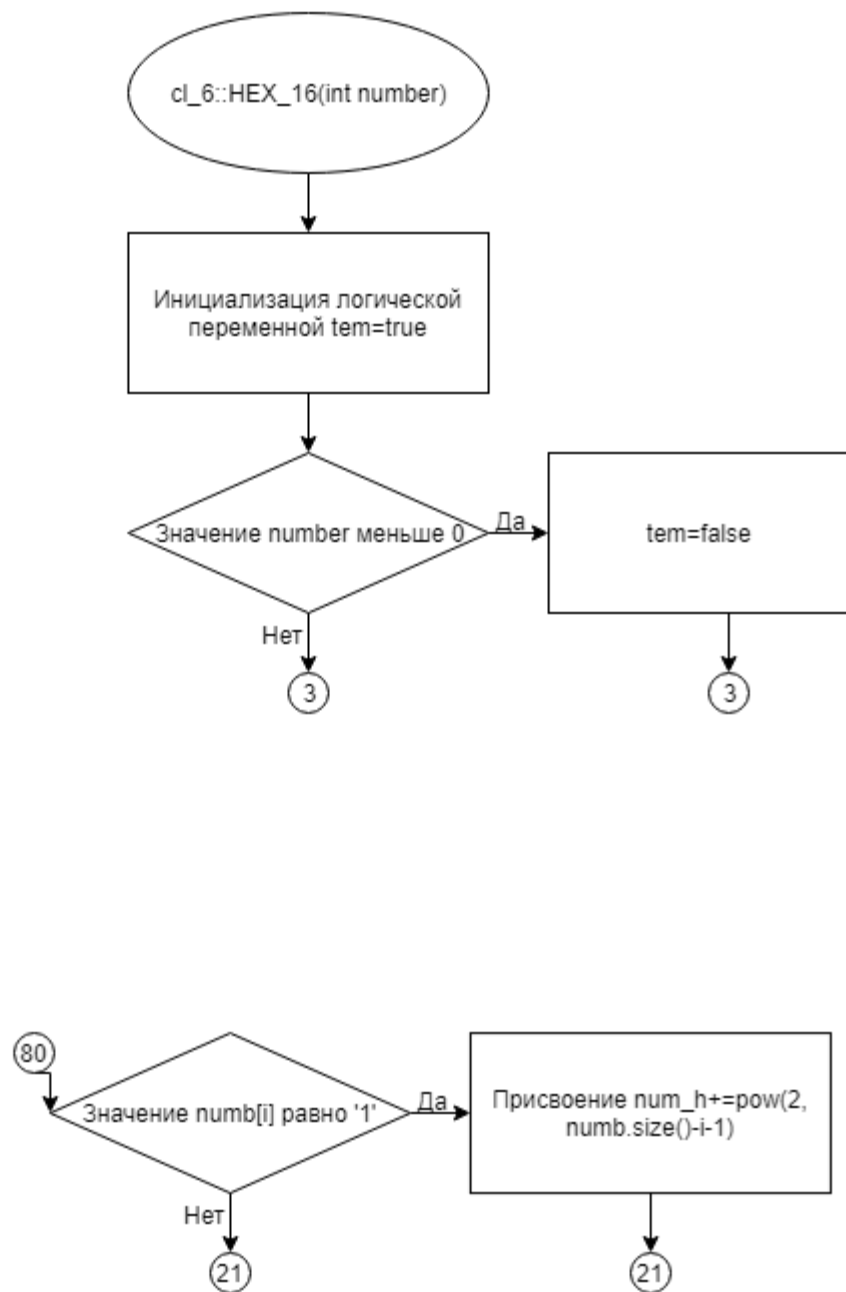


Рисунок 19 – Блок-схема алгоритма

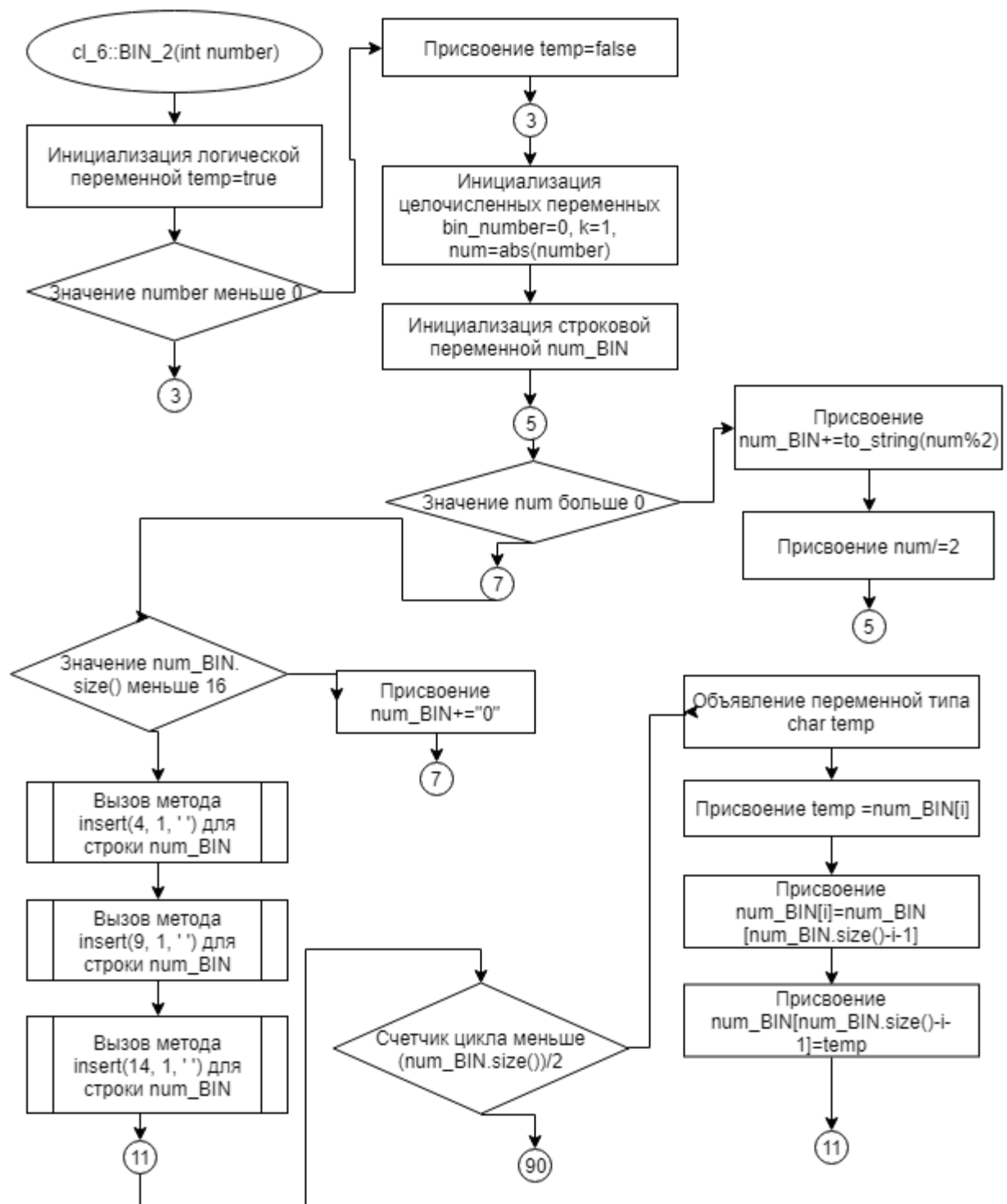
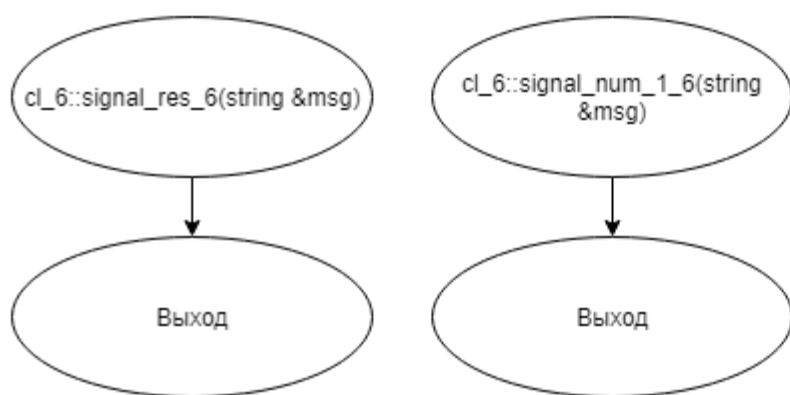


Рисунок 20 – Блок-схема алгоритма



**Рисунок 21 – Блок-схема алгоритма**

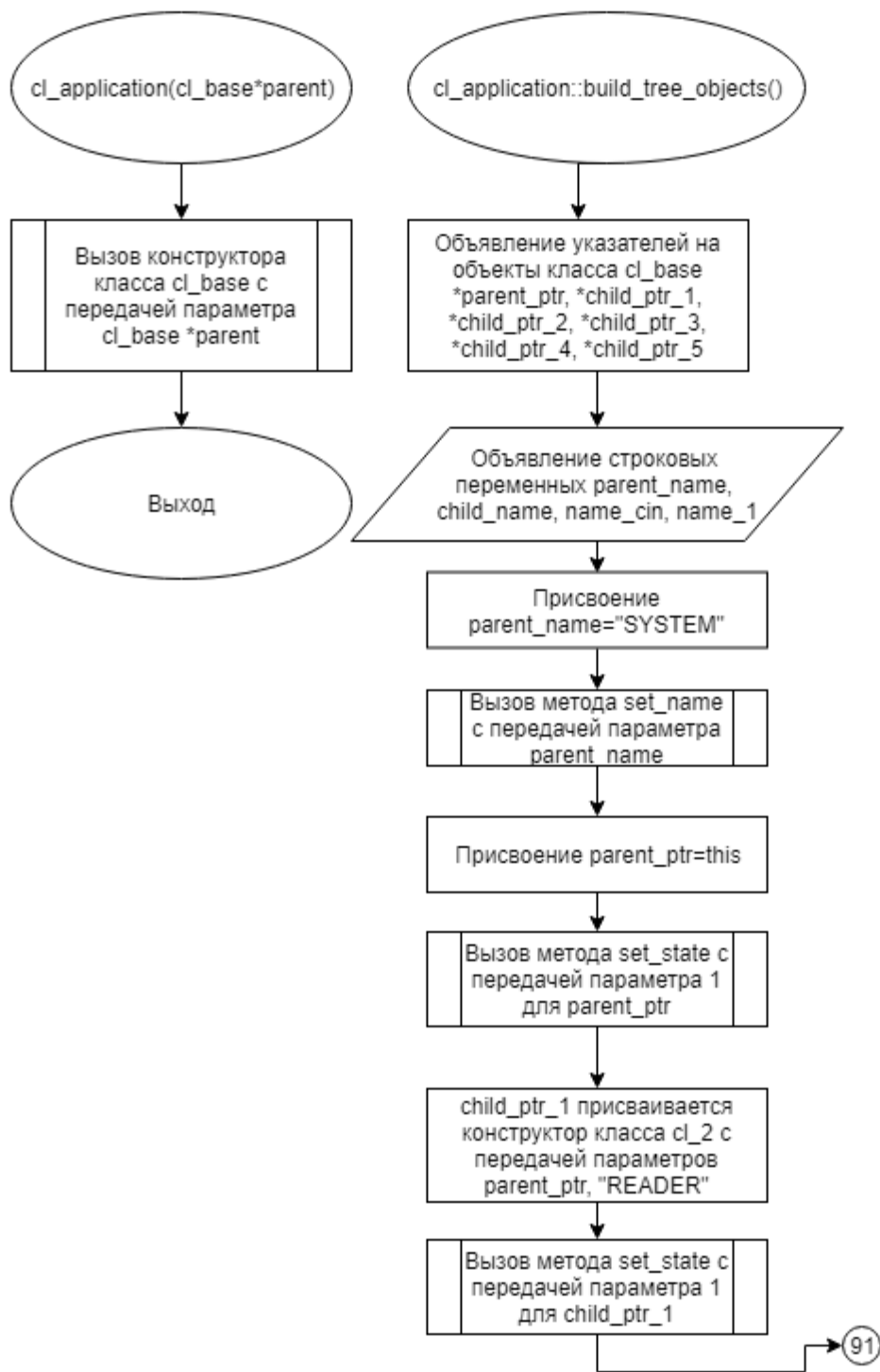


Рисунок 22 – Блок-схема алгоритма

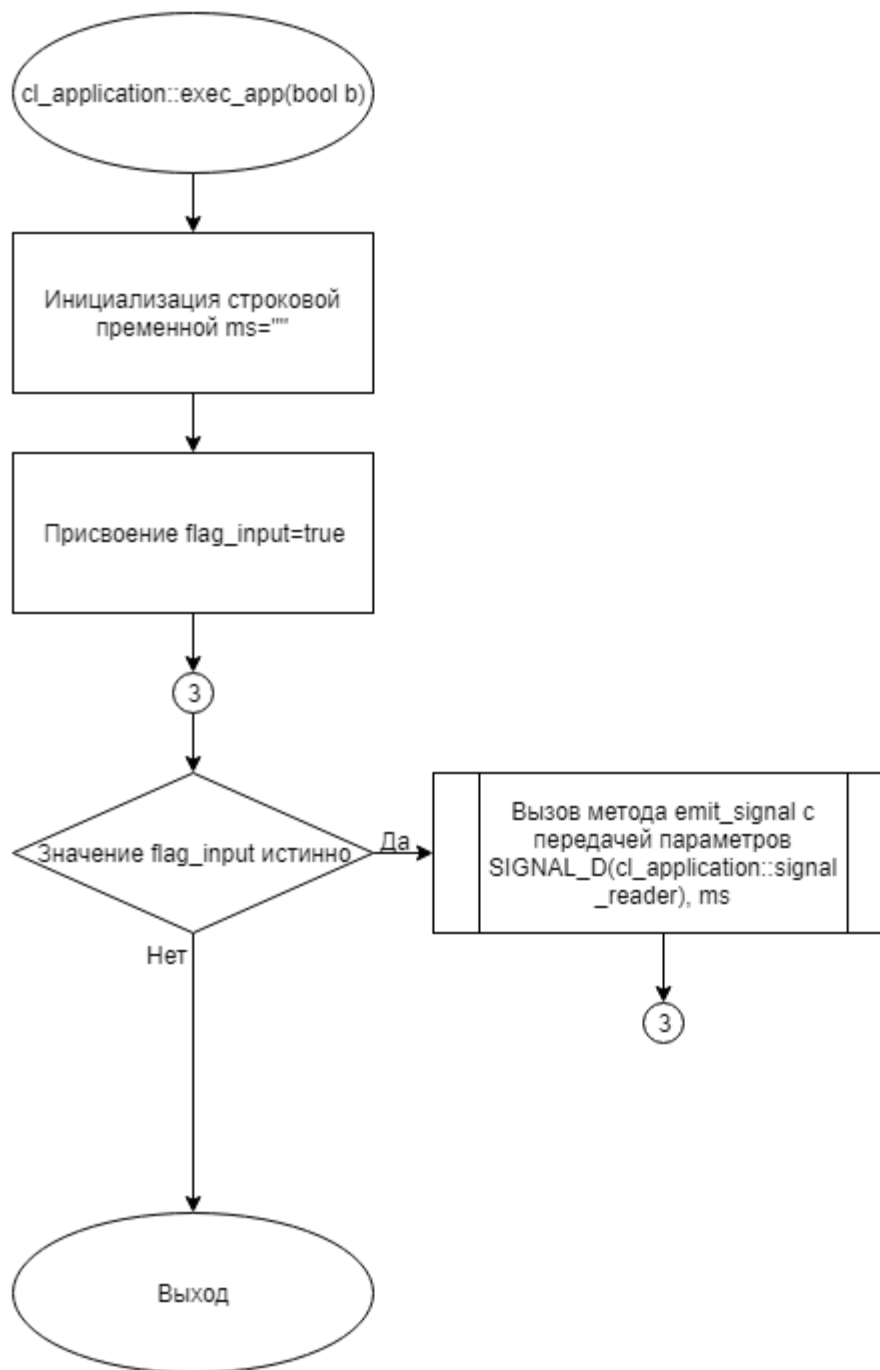


Рисунок 23 – Блок-схема алгоритма

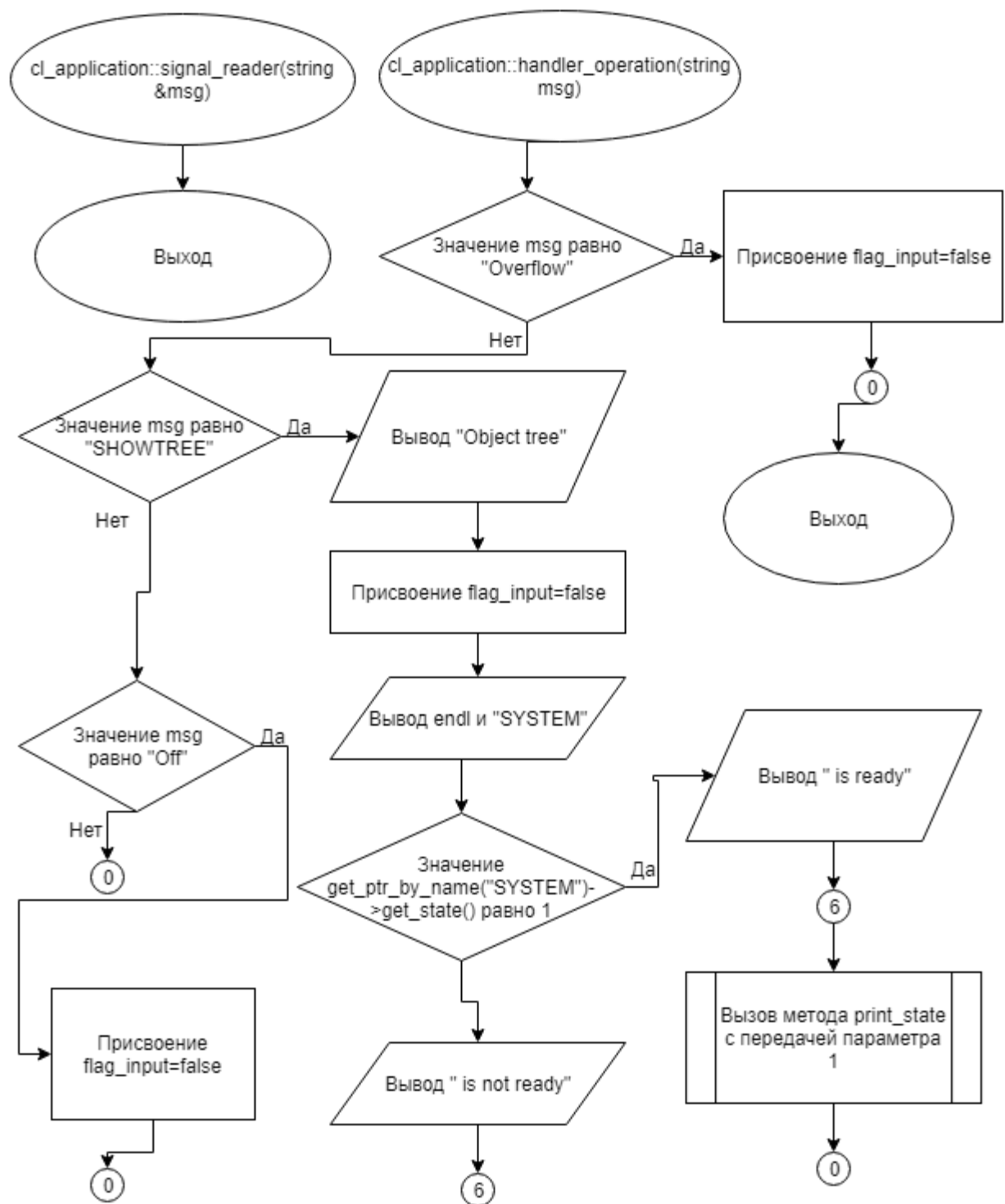
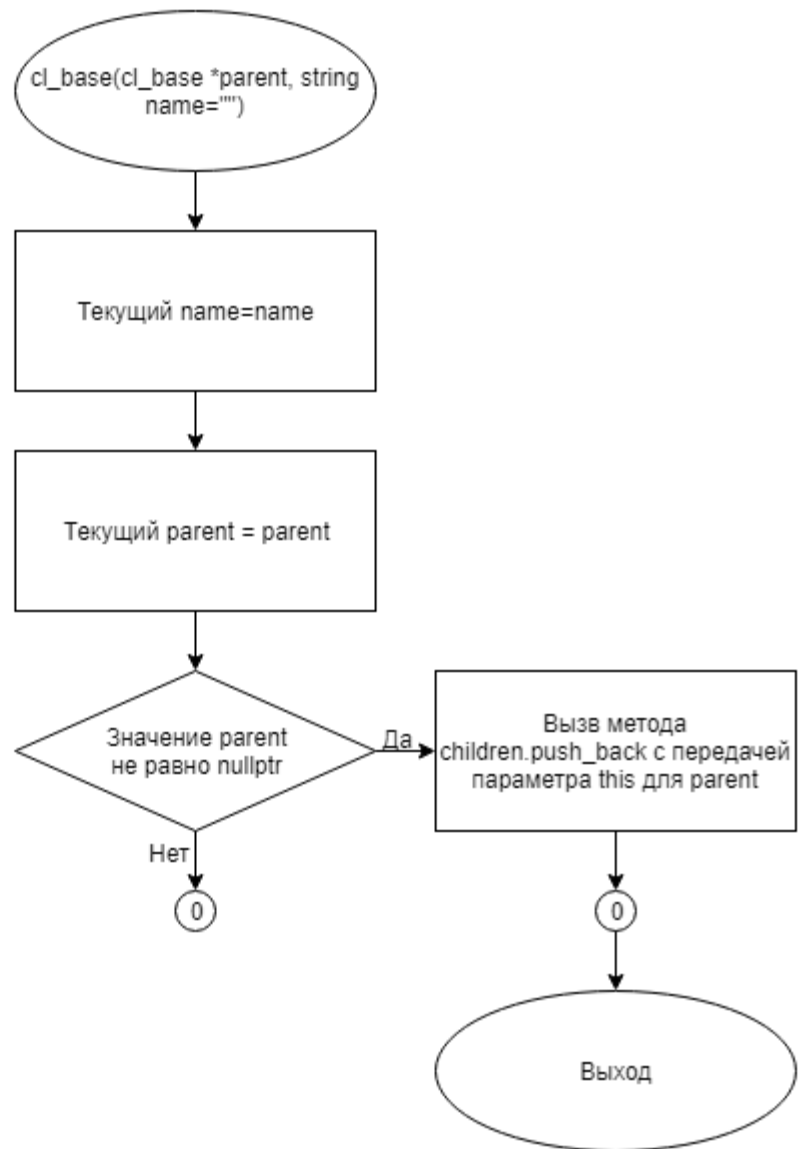


Рисунок 24 – Блок-схема алгоритма



**Рисунок 25 – Блок-схема алгоритма**



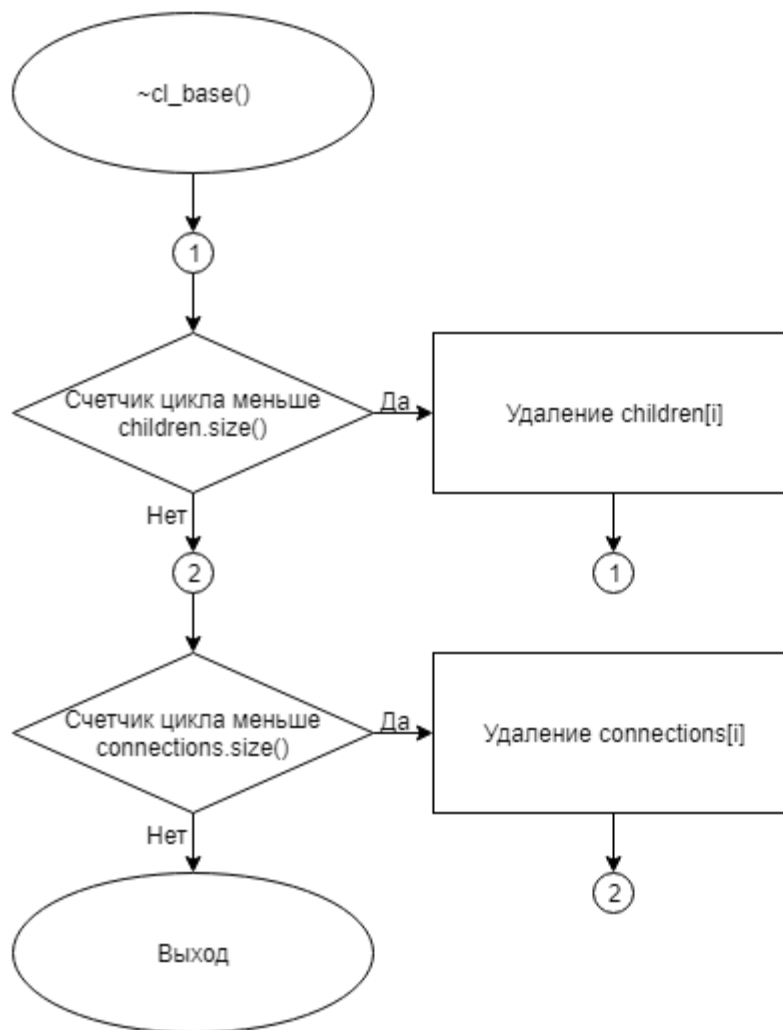
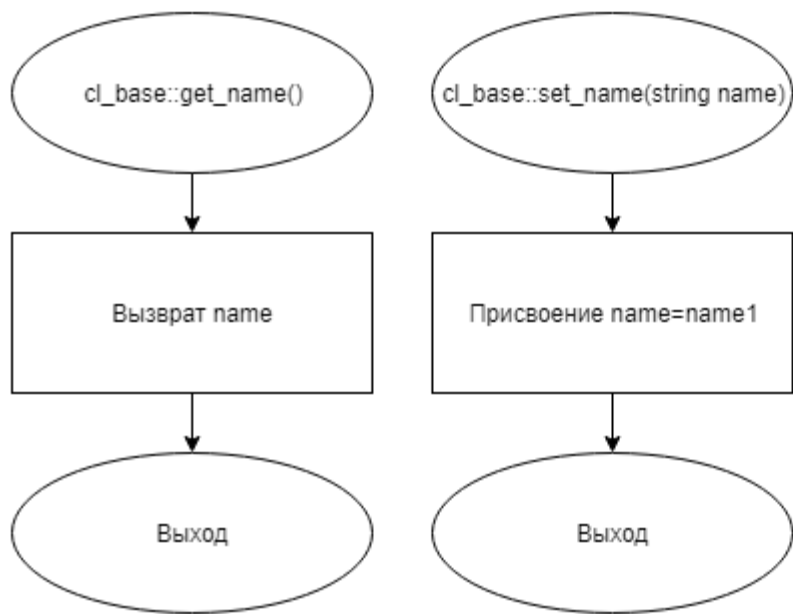


Рисунок 26 – Блок-схема алгоритма



**Рисунок 27 – Блок-схема алгоритма**

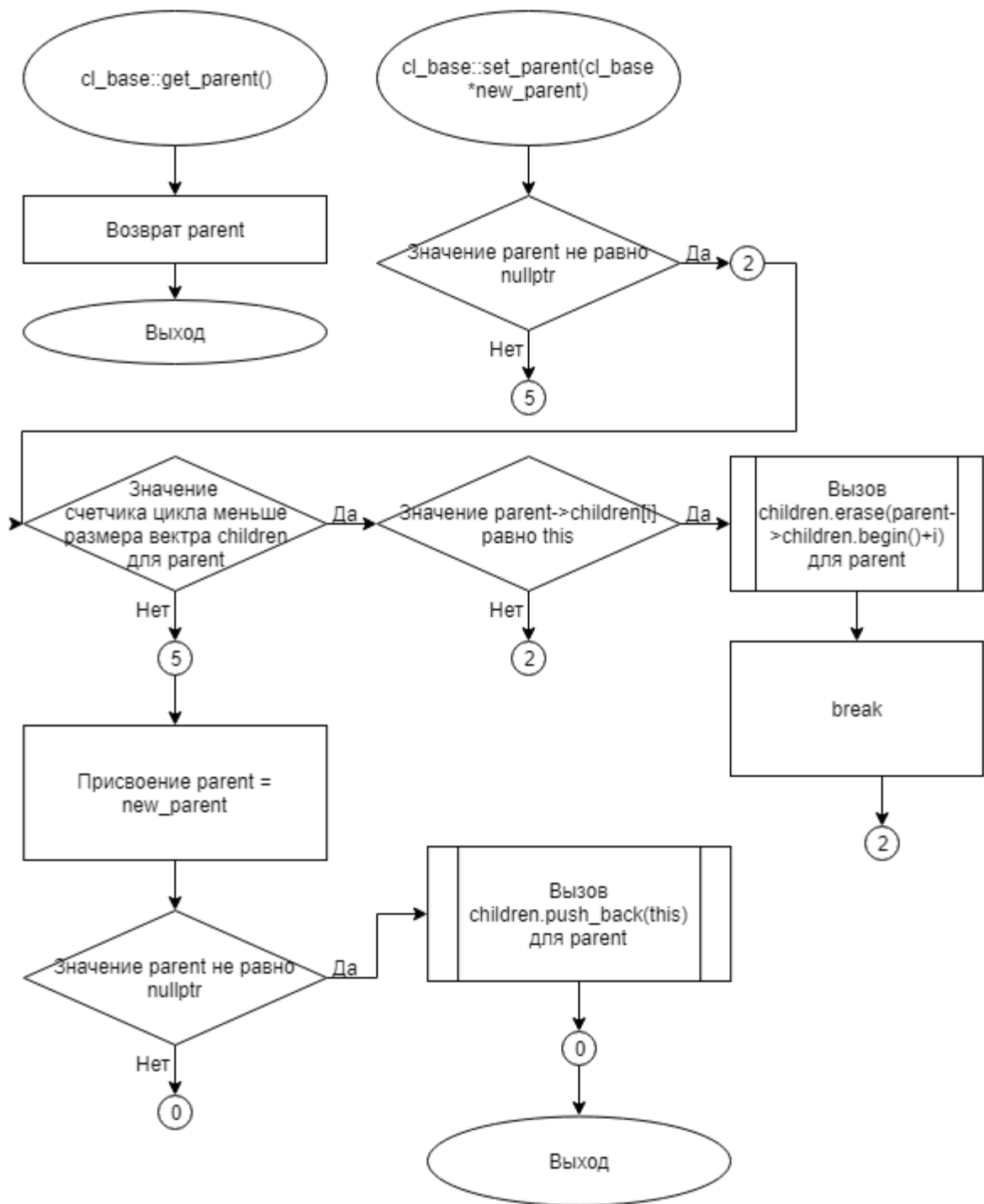


Рисунок 28 – Блок-схема алгоритма

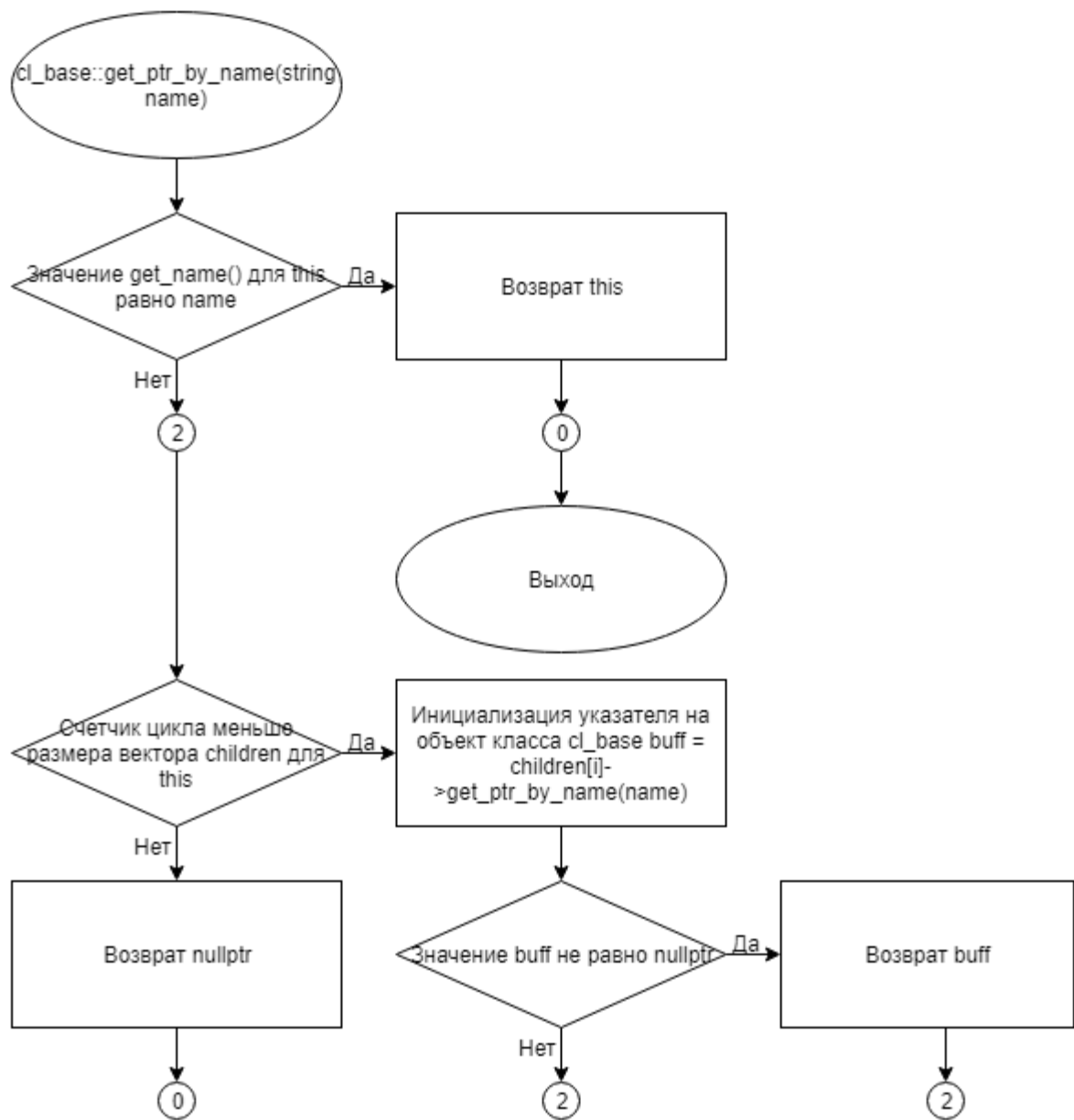


Рисунок 29 – Блок-схема алгоритма

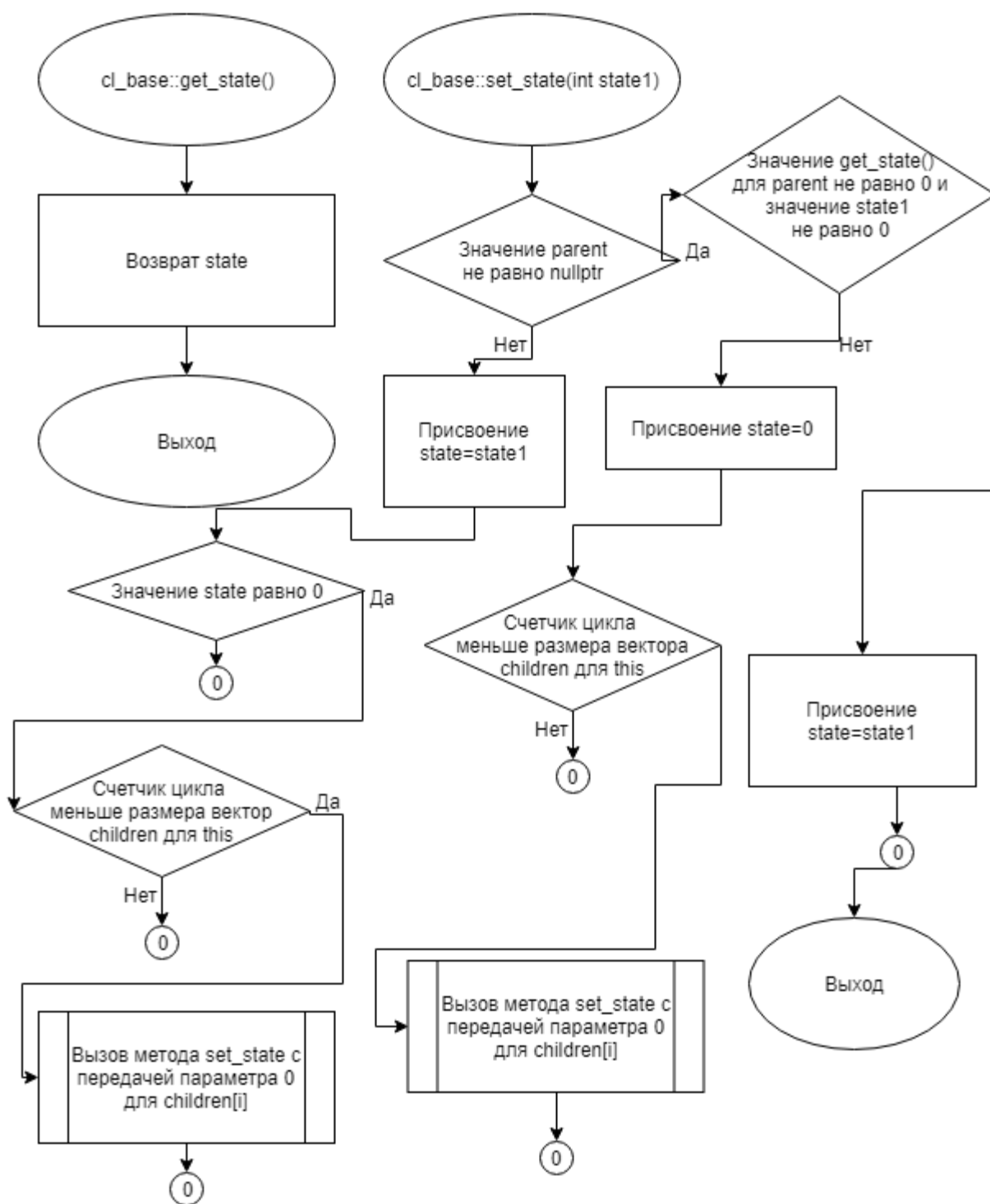


Рисунок 30 – Блок-схема алгоритма

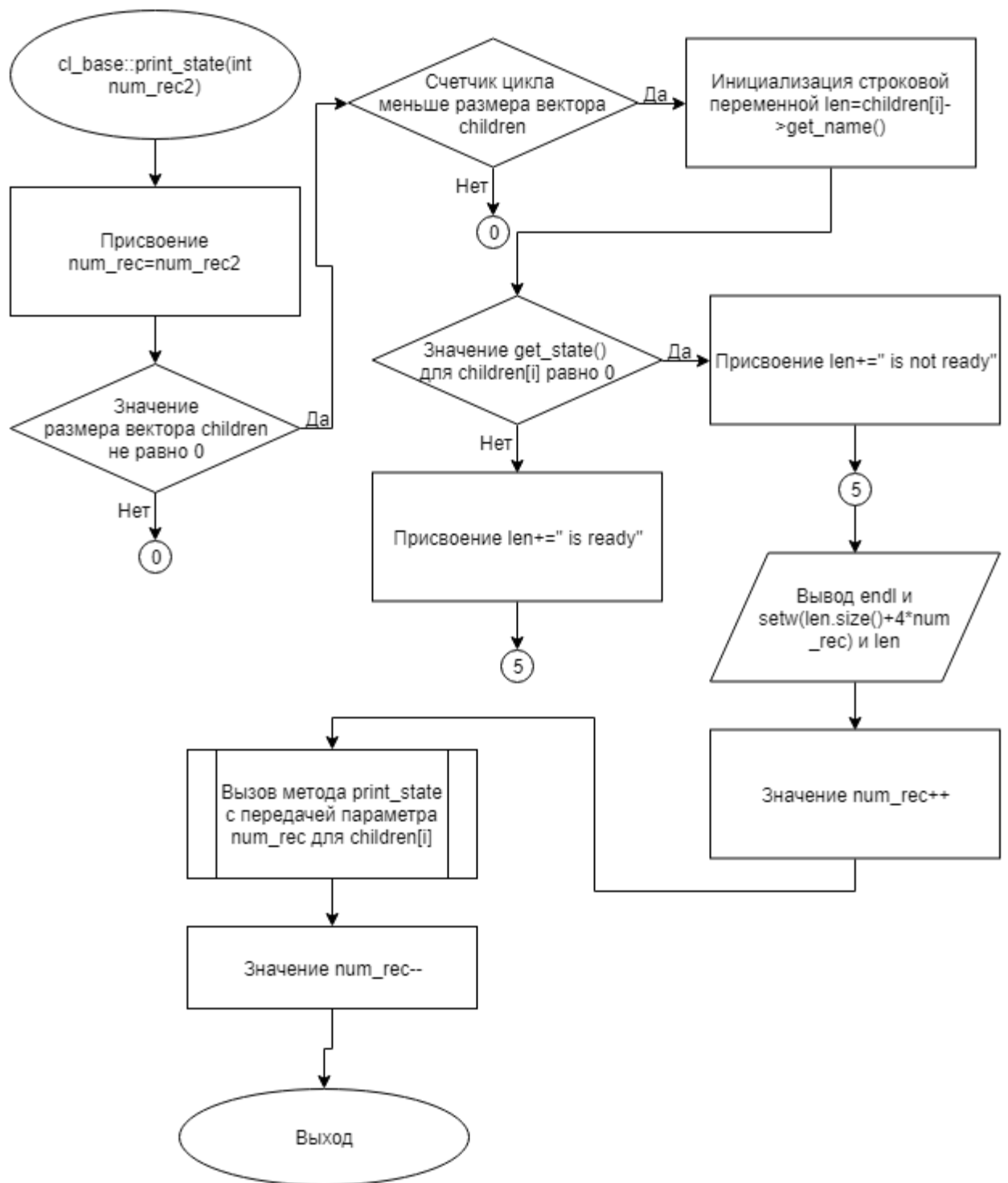


Рисунок 31 – Блок-схема алгоритма

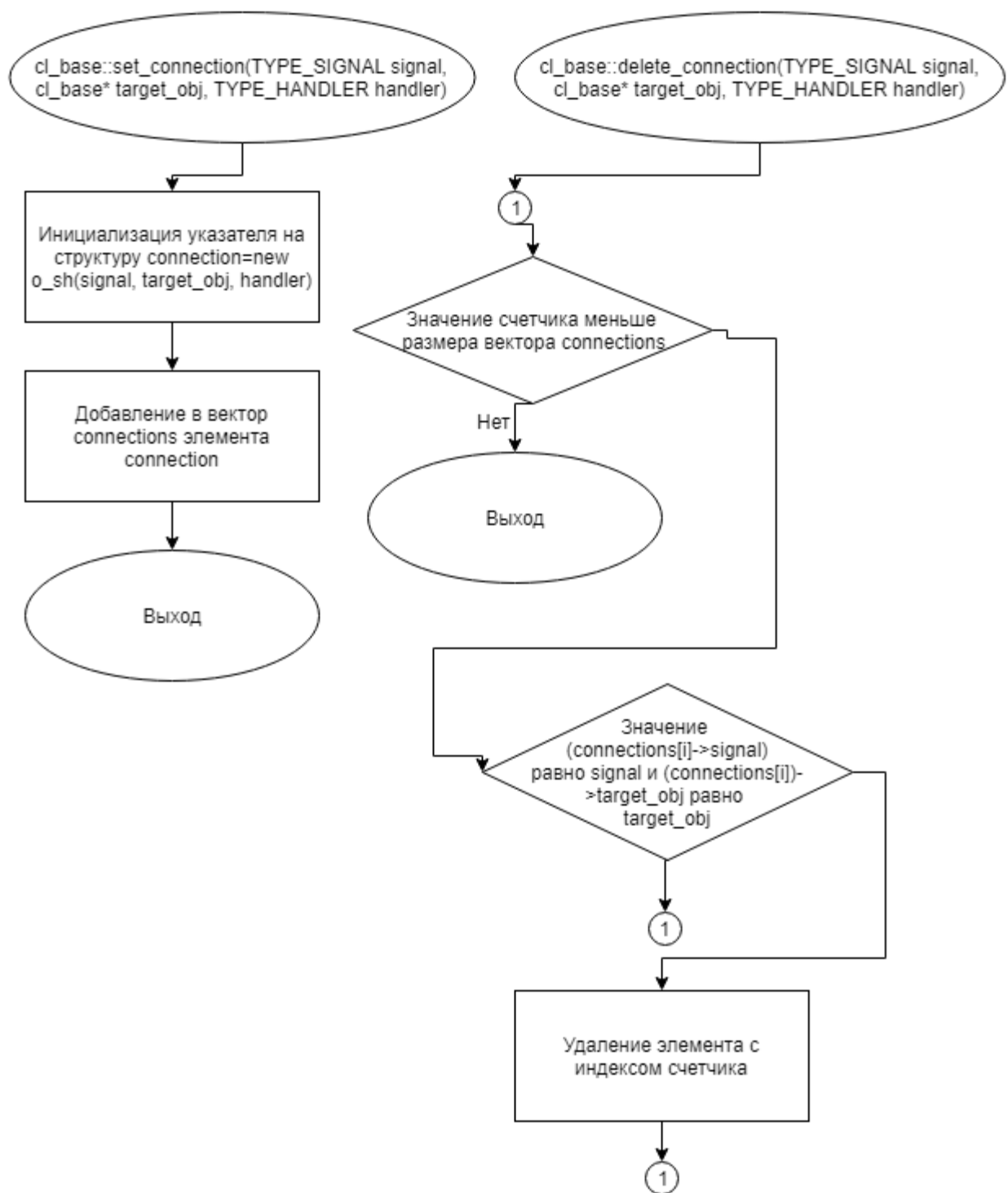


Рисунок 32 – Блок-схема алгоритма

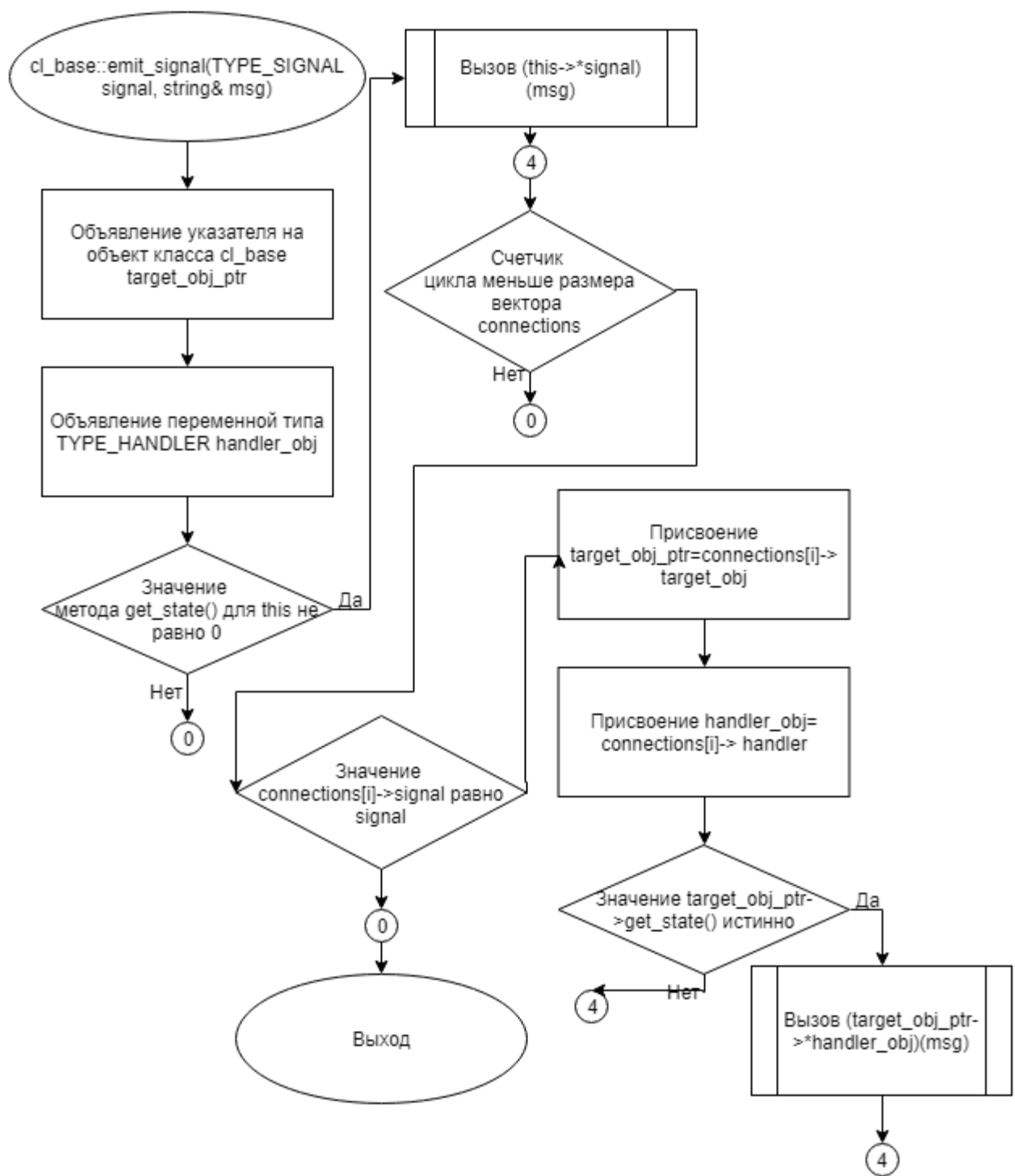


Рисунок 33 – Блок-схема алгоритма



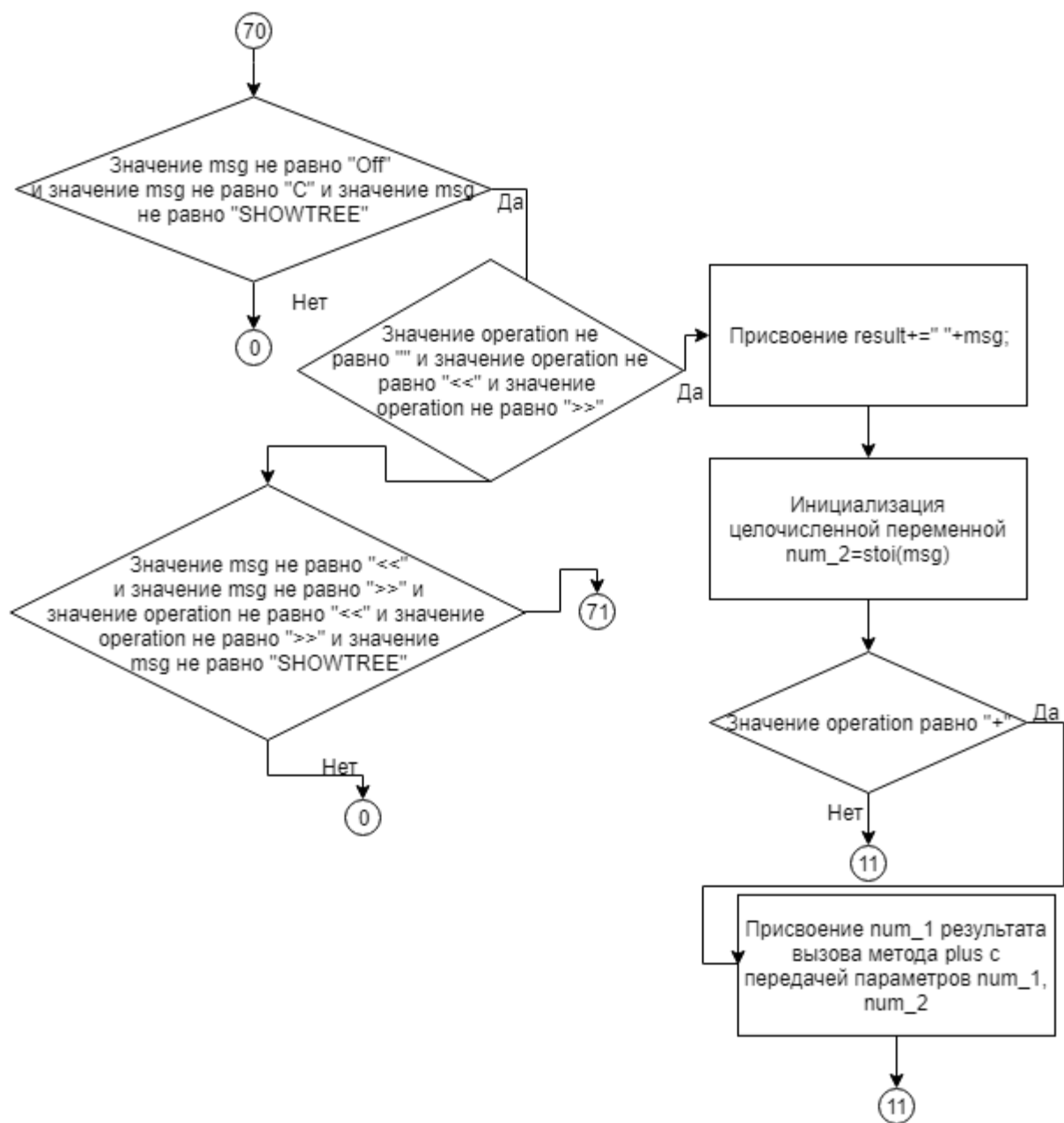


Рисунок 34 – Блок-схема алгоритма

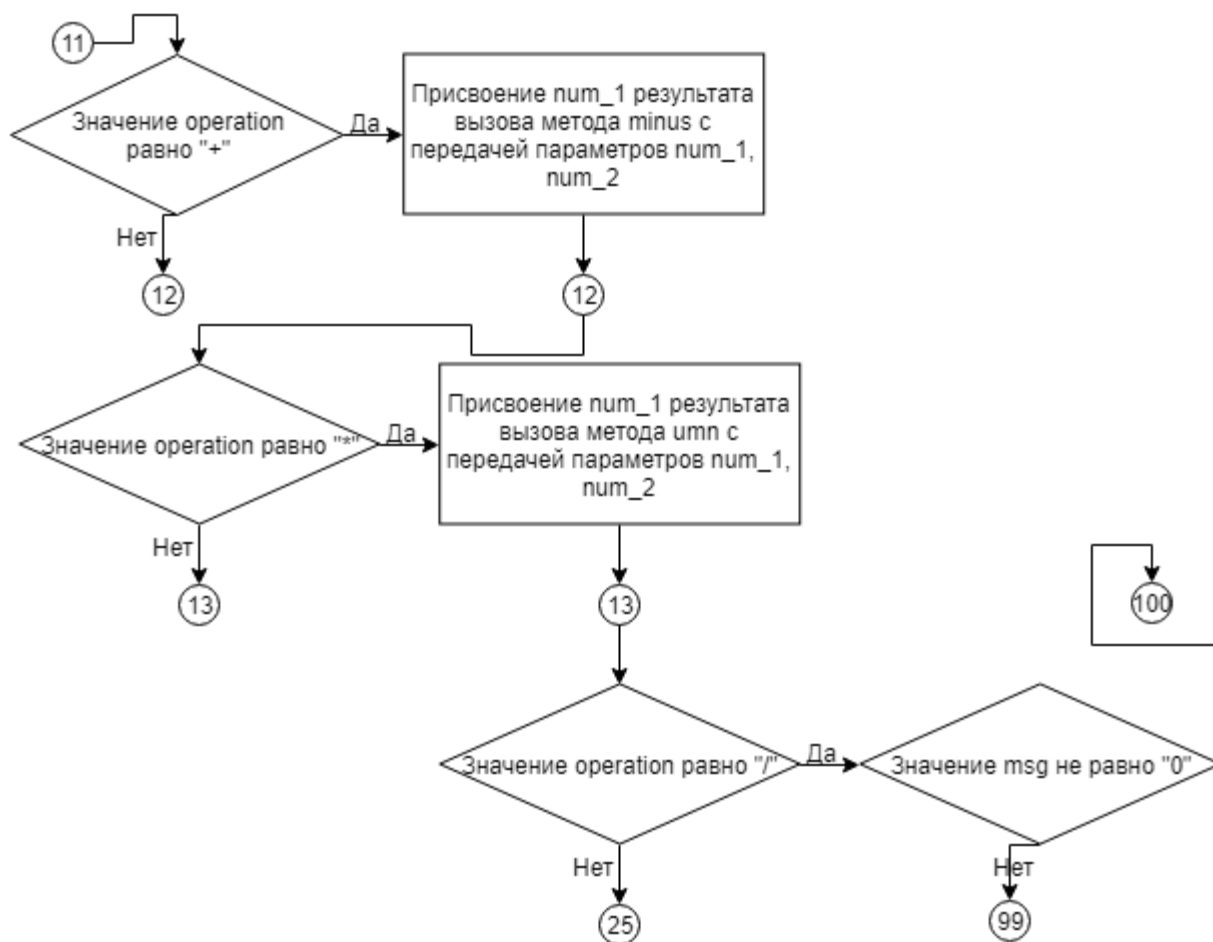


Рисунок 35 – Блок-схема алгоритма

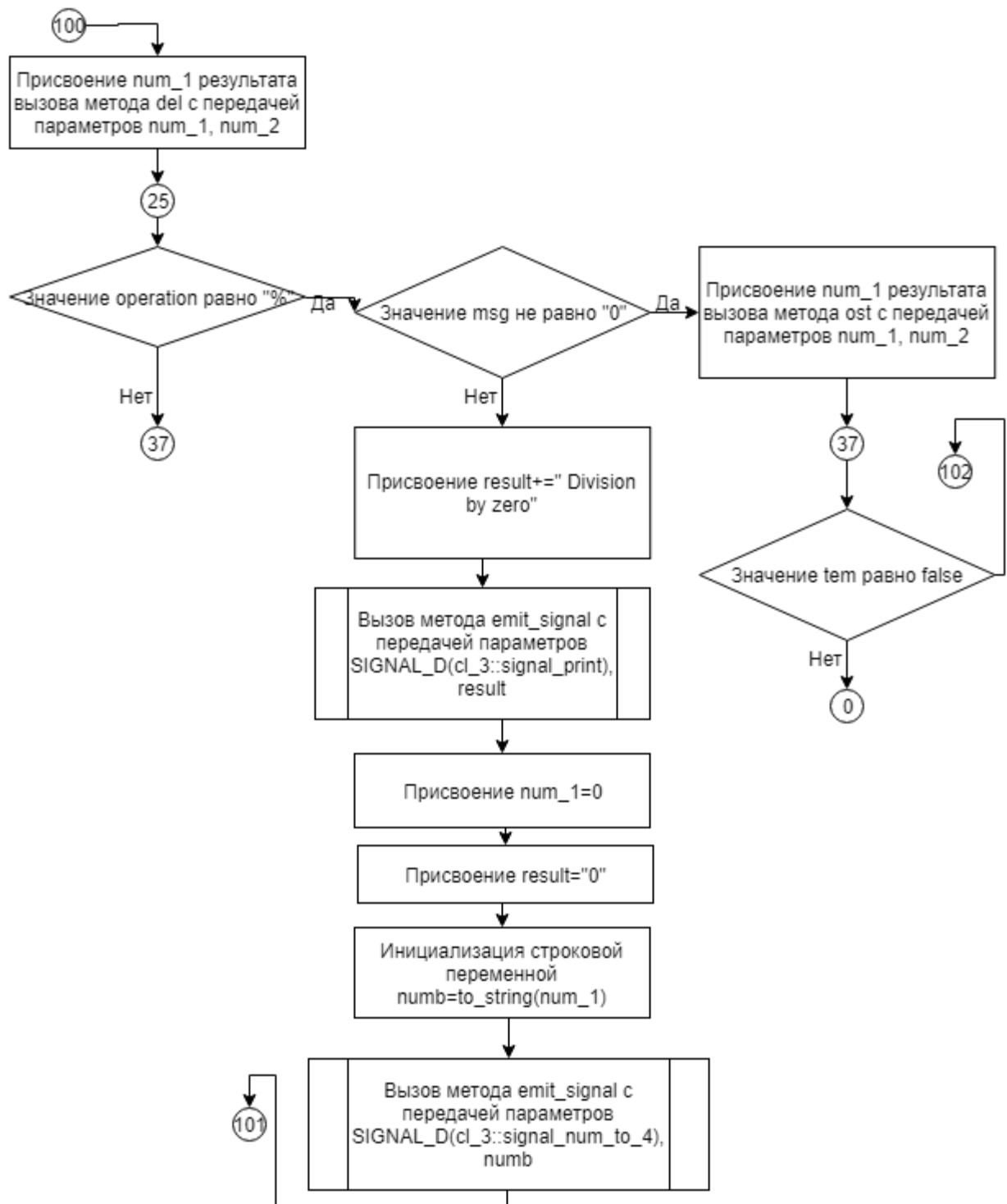


Рисунок 36 – Блок-схема алгоритма

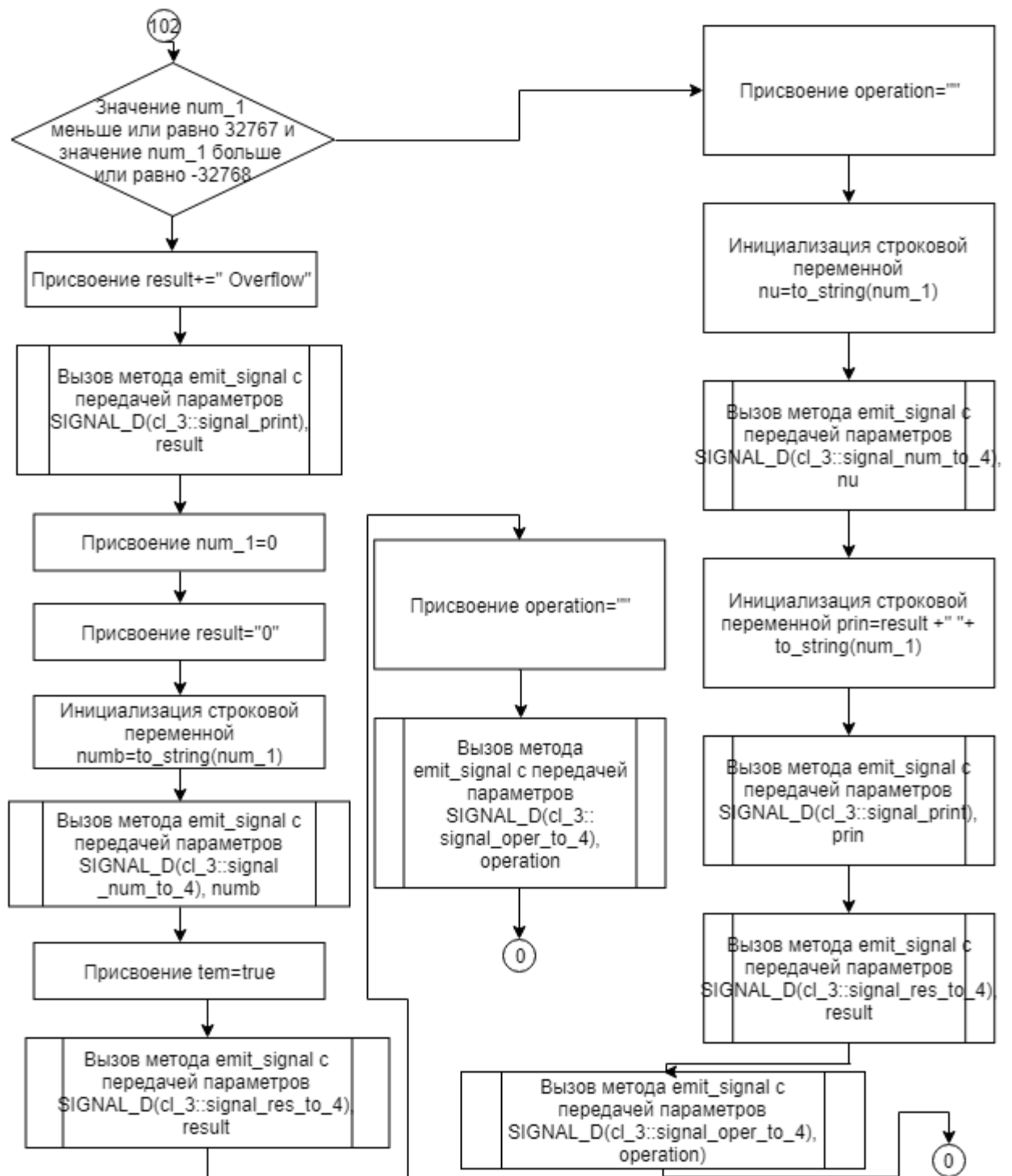


Рисунок 37 – Блок-схема алгоритма

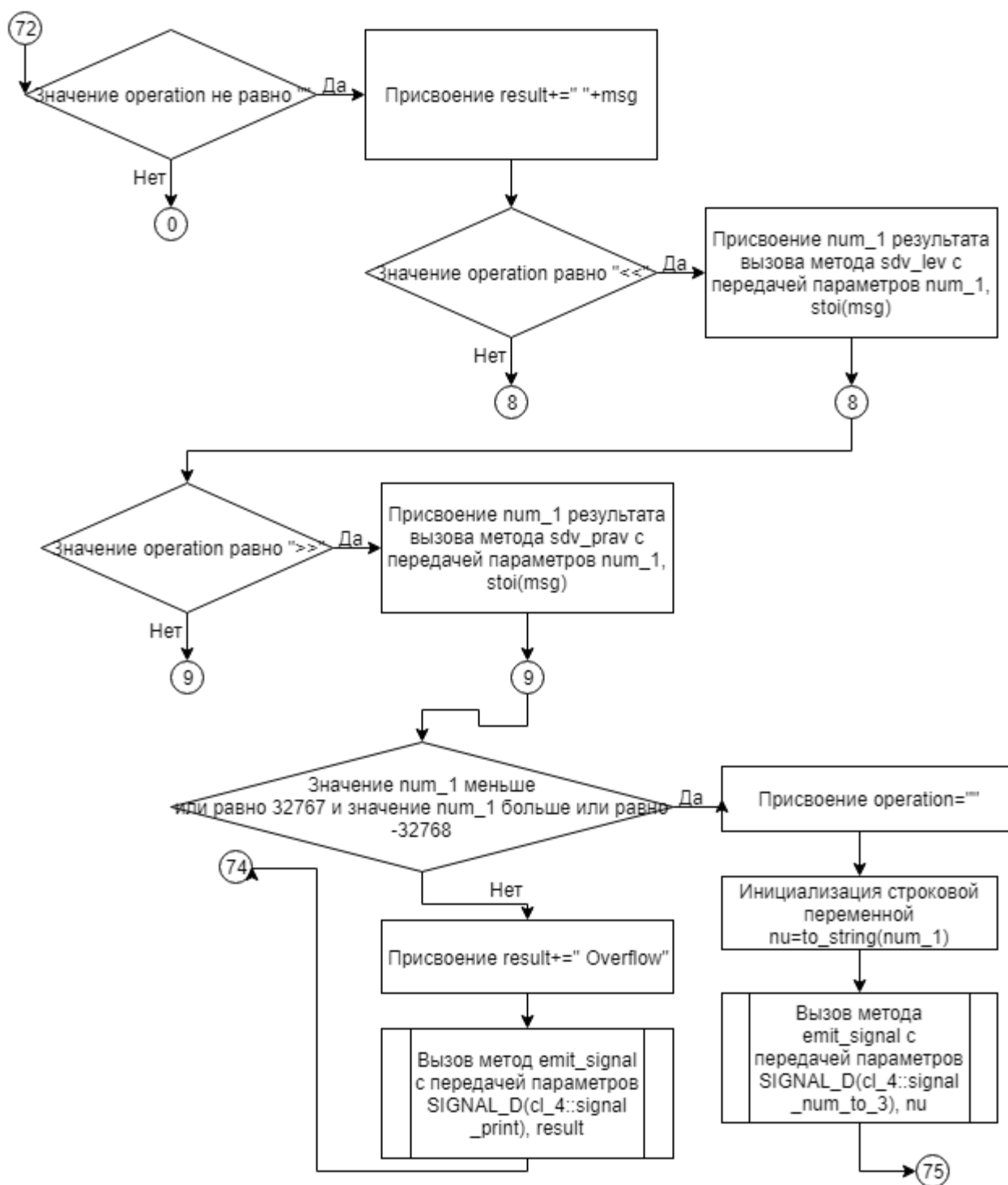


Рисунок 38 – Блок-схема алгоритма

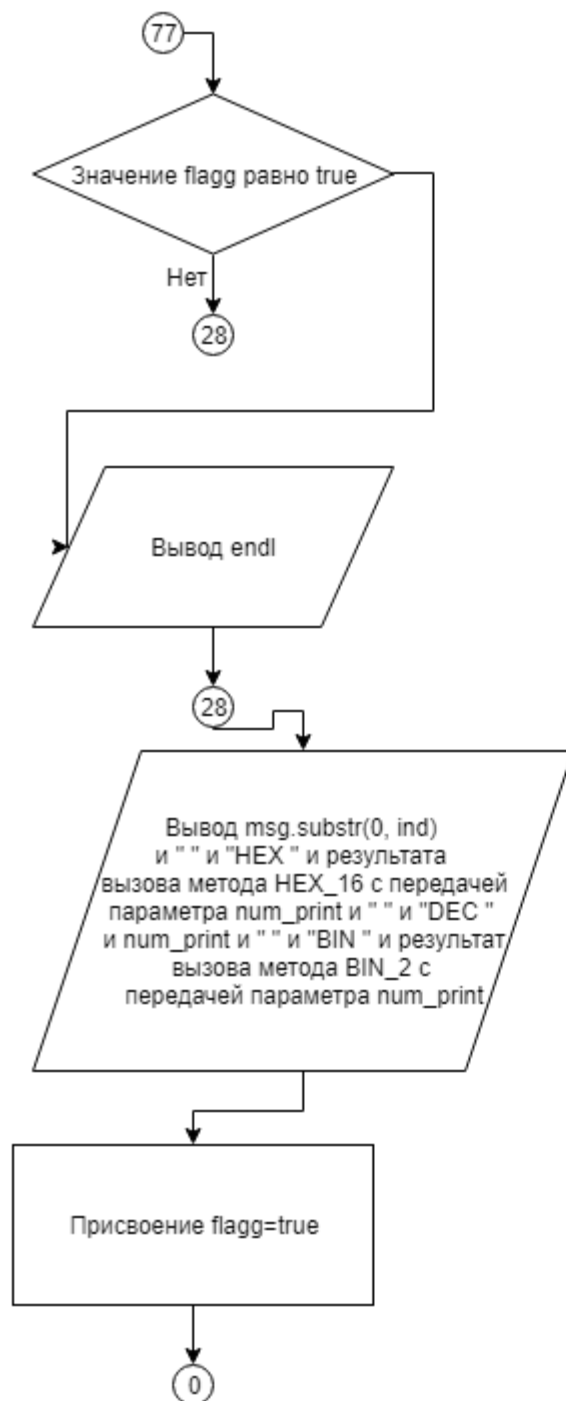


Рисунок 39 – Блок-схема алгоритма

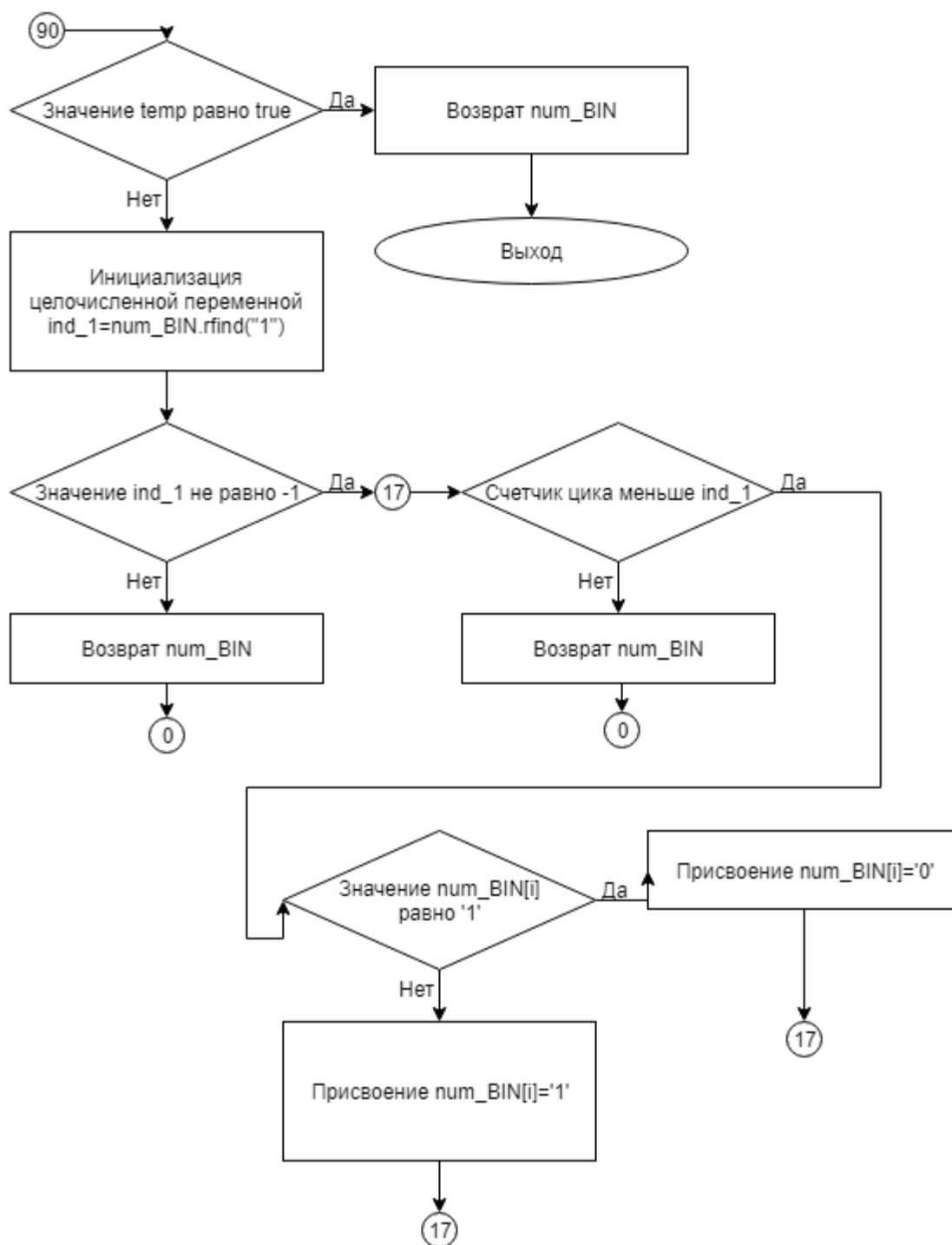


Рисунок 40 – Блок-схема алгоритма

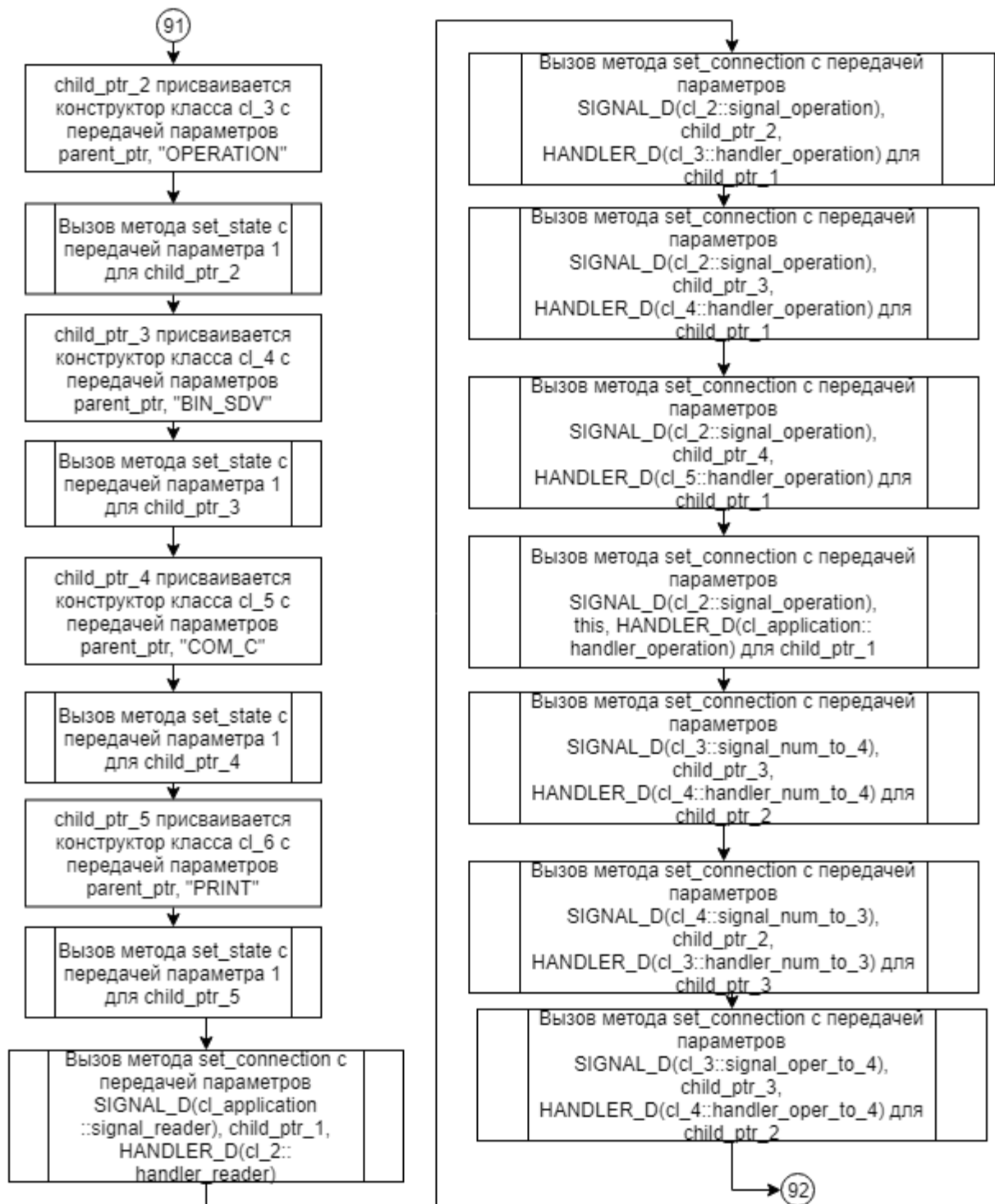


Рисунок 41 – Блок-схема алгоритма



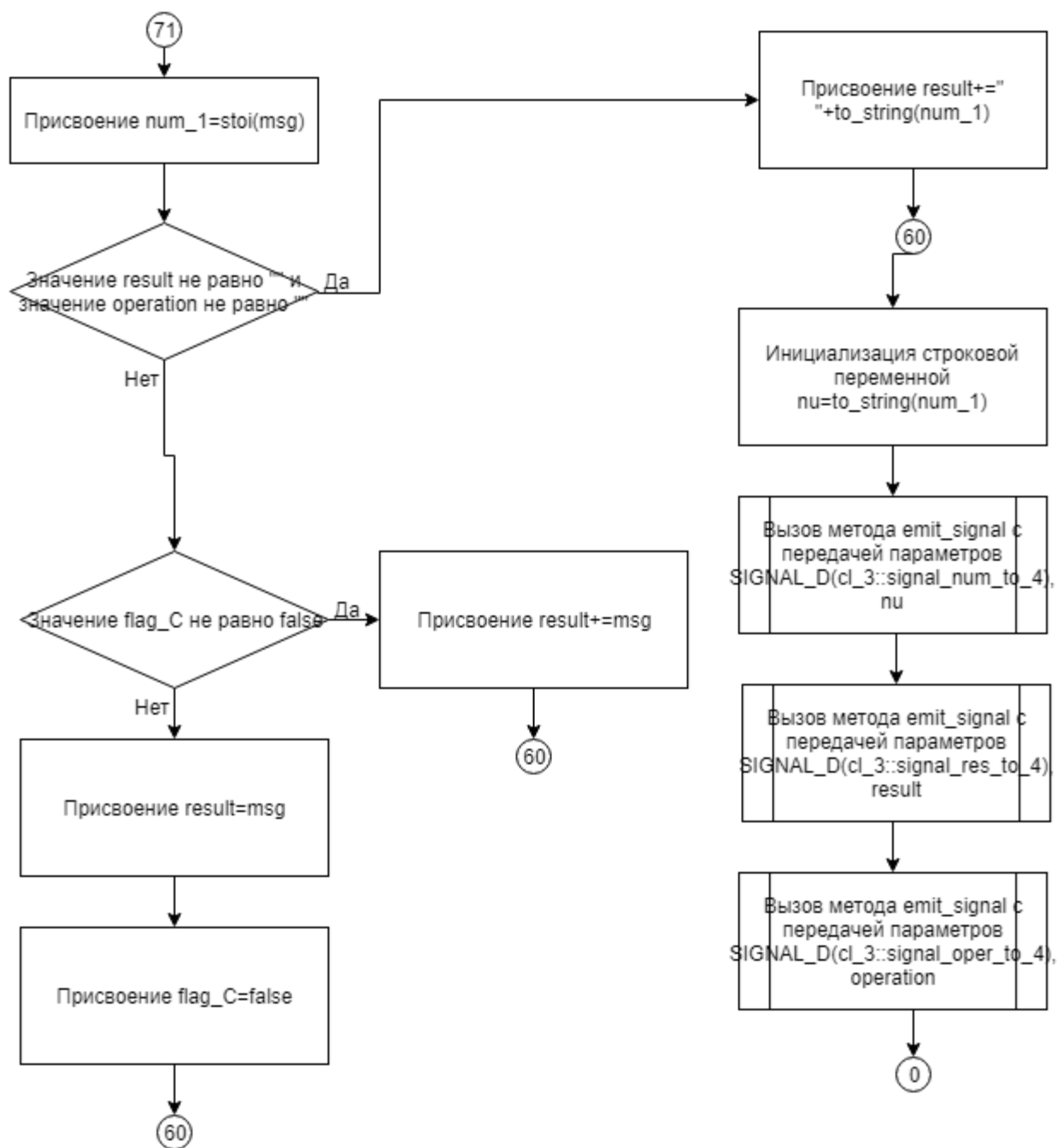


Рисунок 42 – Блок-схема алгоритма

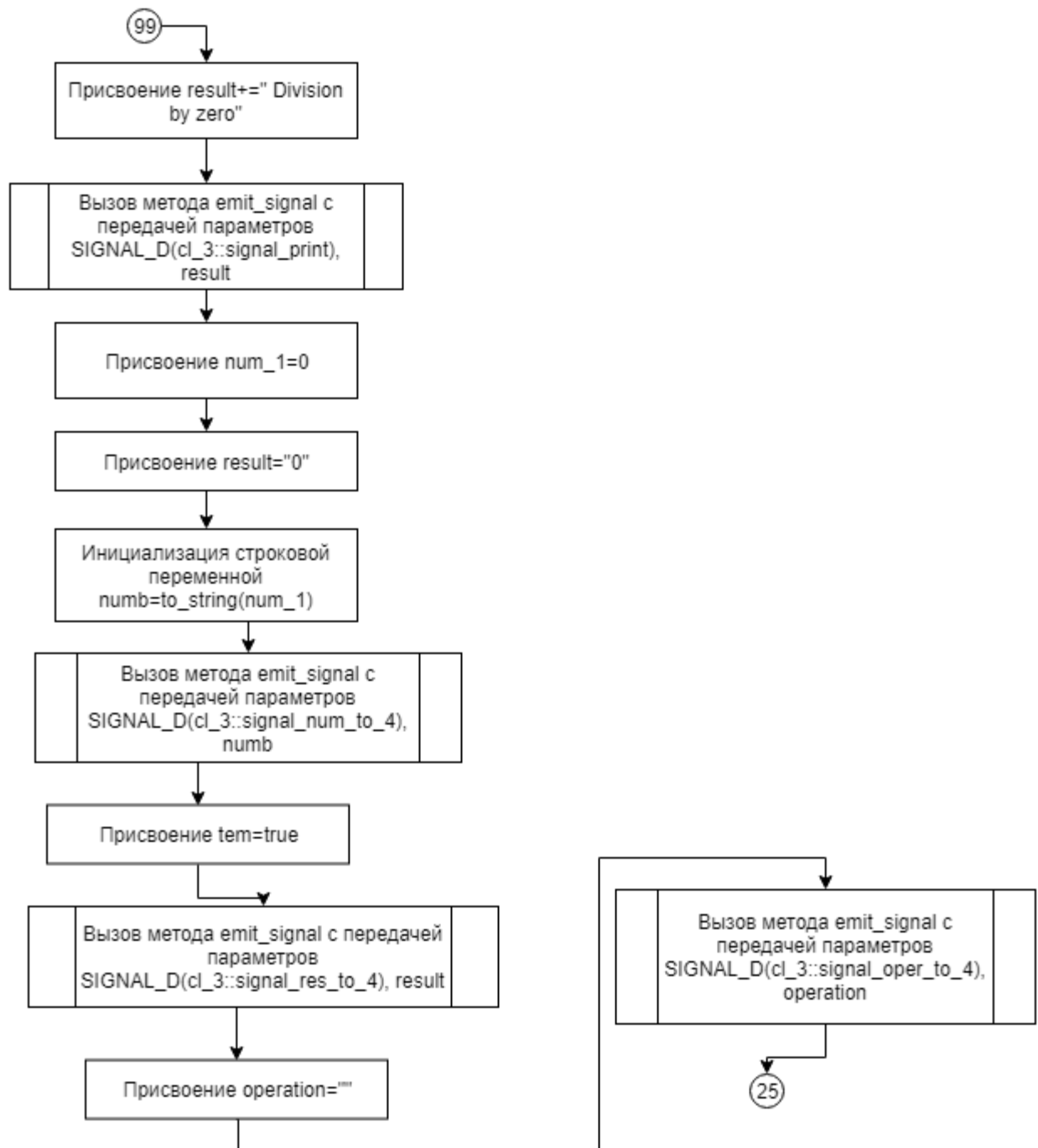
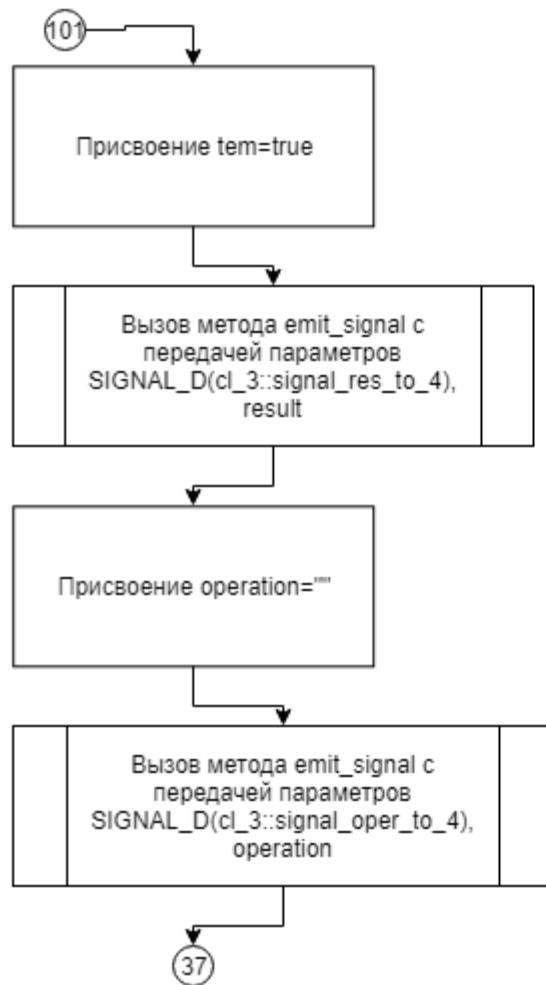


Рисунок 43 – Блок-схема алгоритма



**Рисунок 44 – Блок-схема алгоритма**

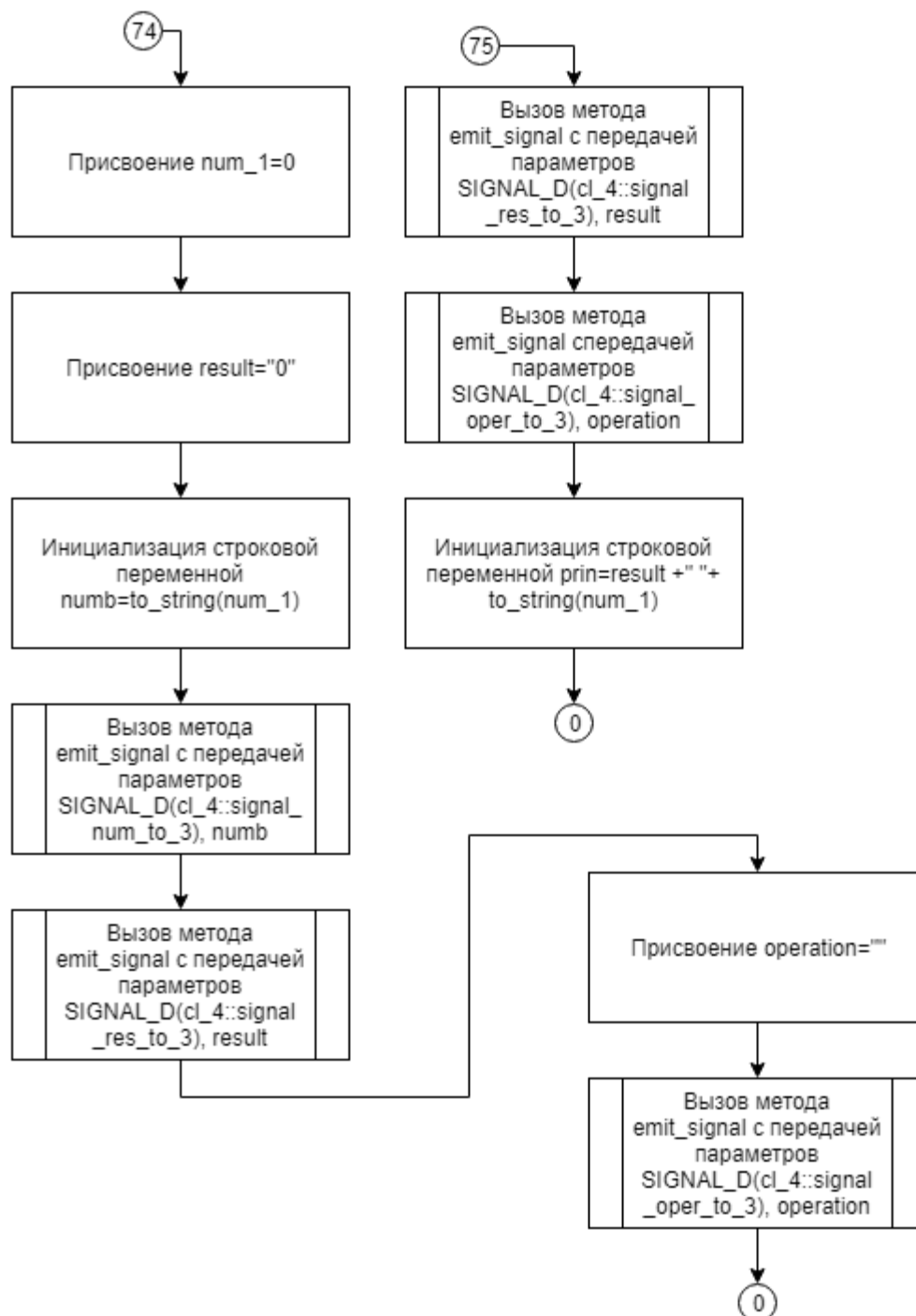


Рисунок 45 – Блок-схема алгоритма

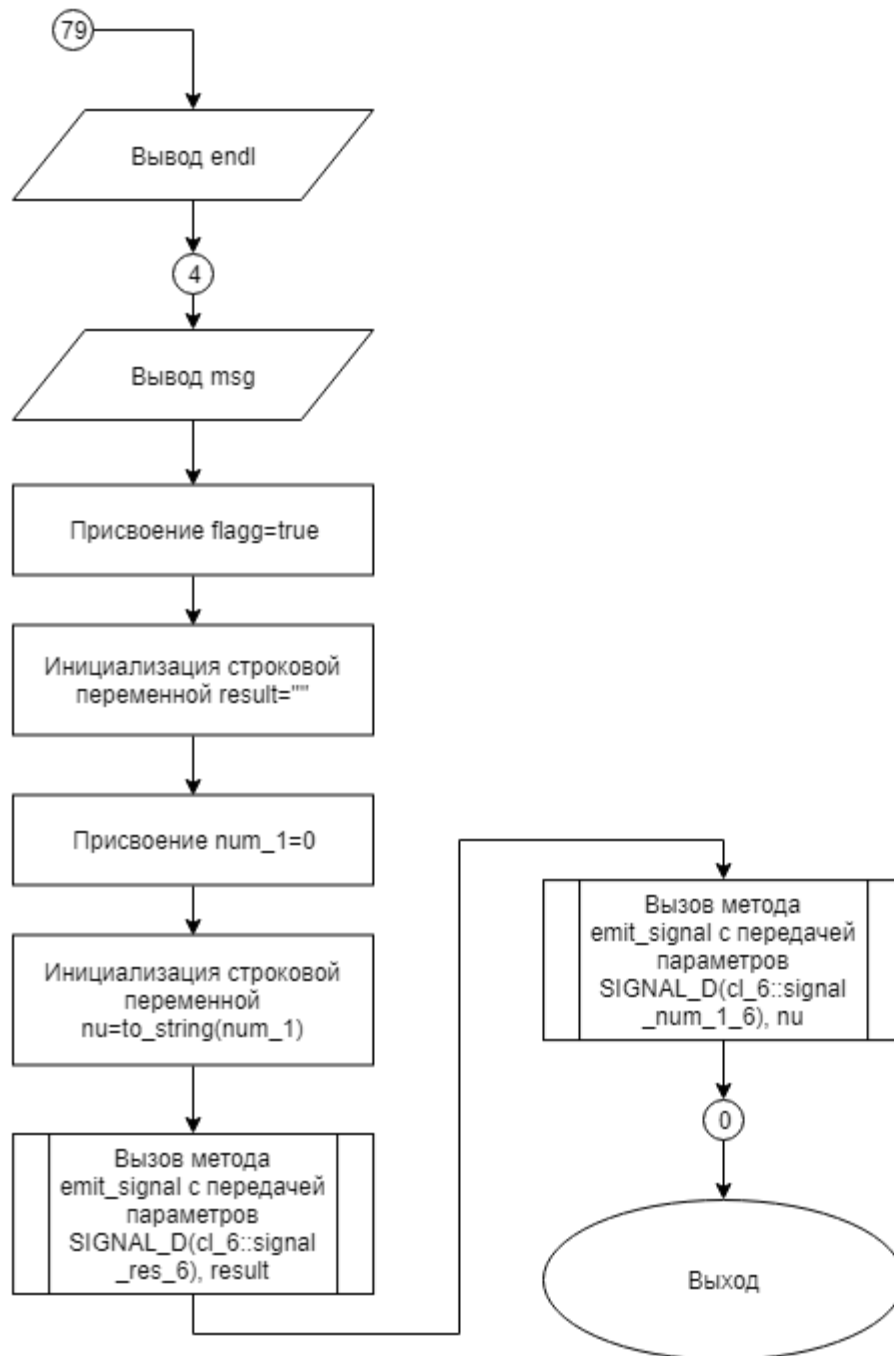


Рисунок 46 – Блок-схема алгоритма

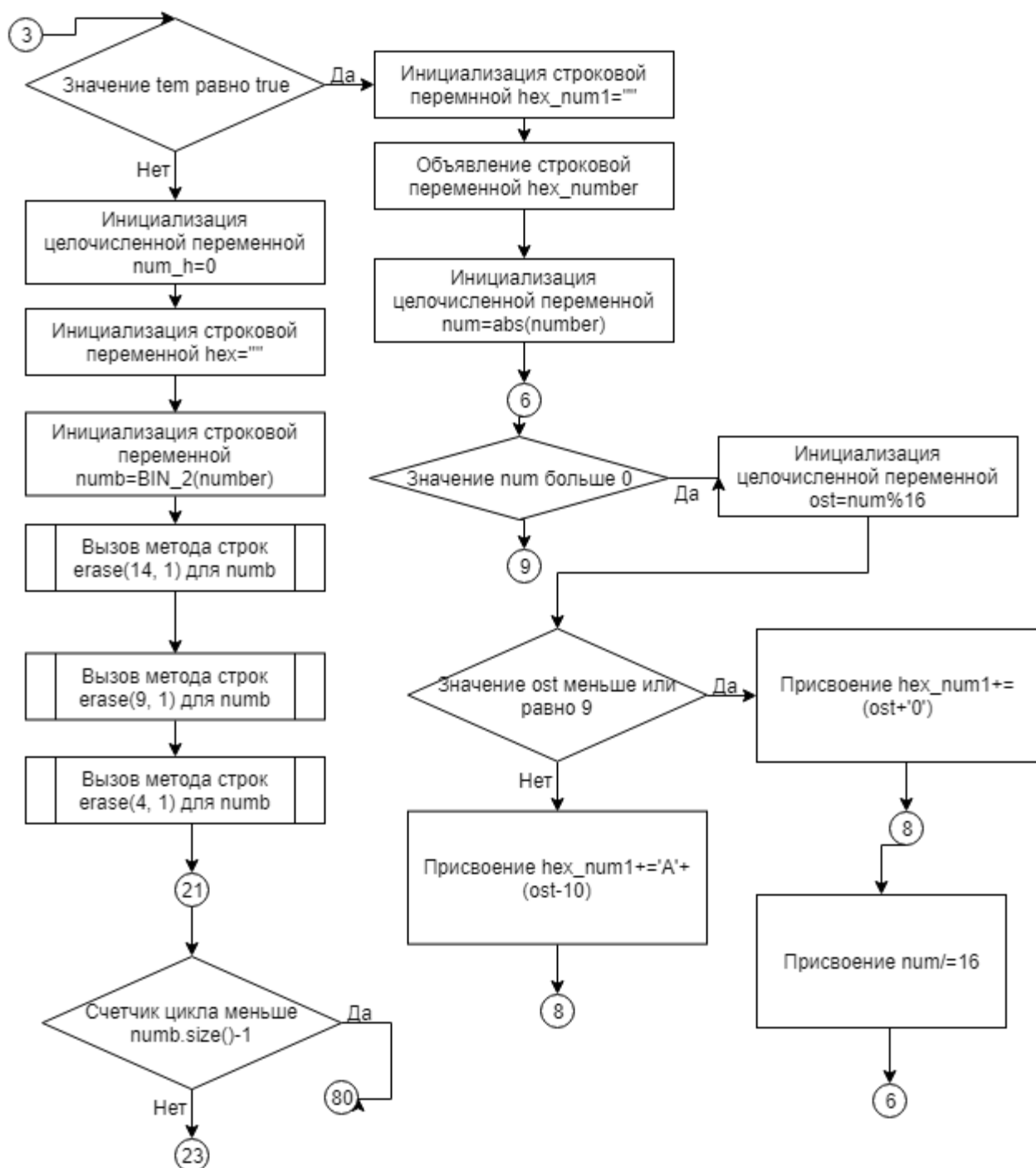


Рисунок 47 – Блок-схема алгоритма

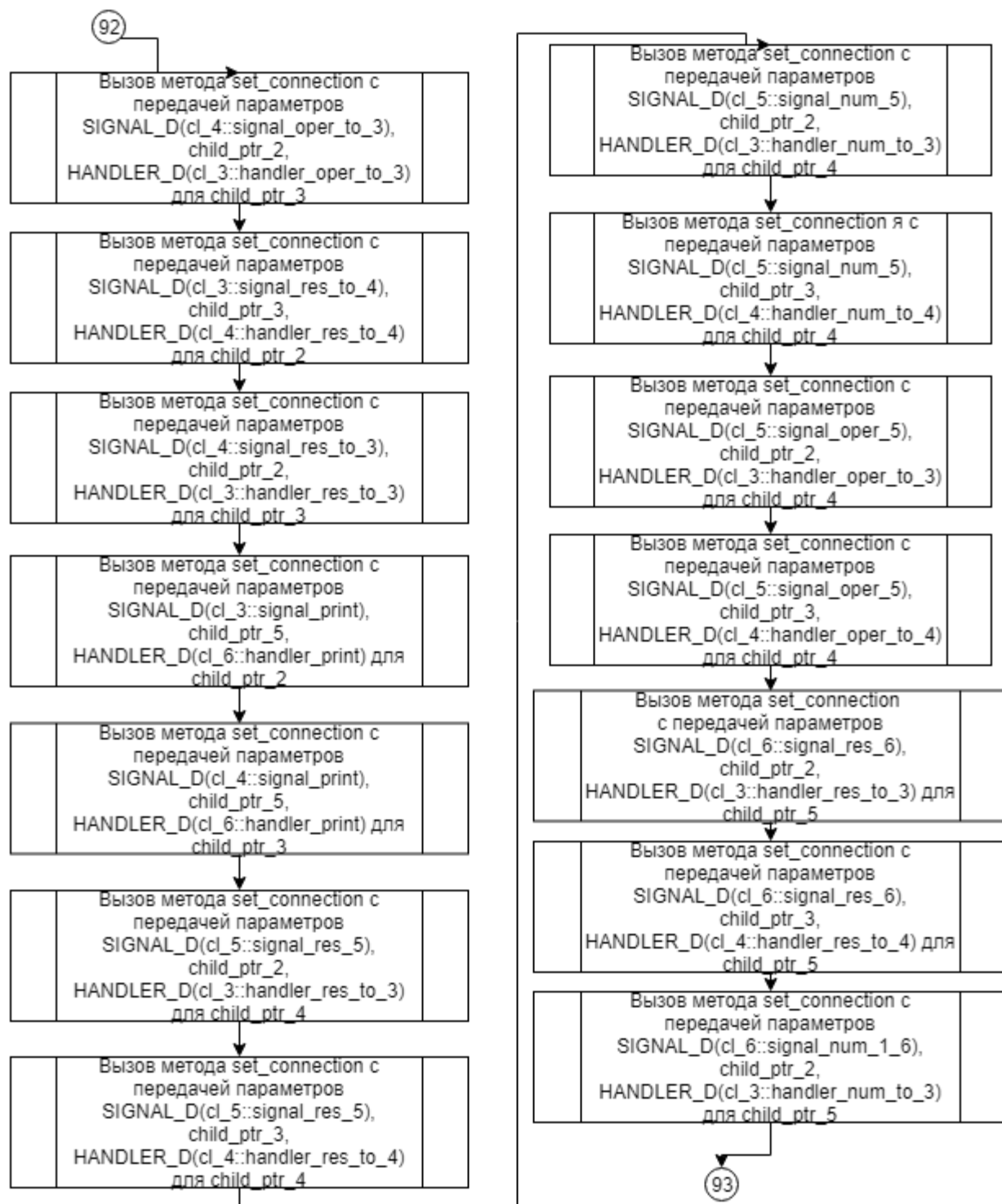


Рисунок 48 – Блок-схема алгоритма

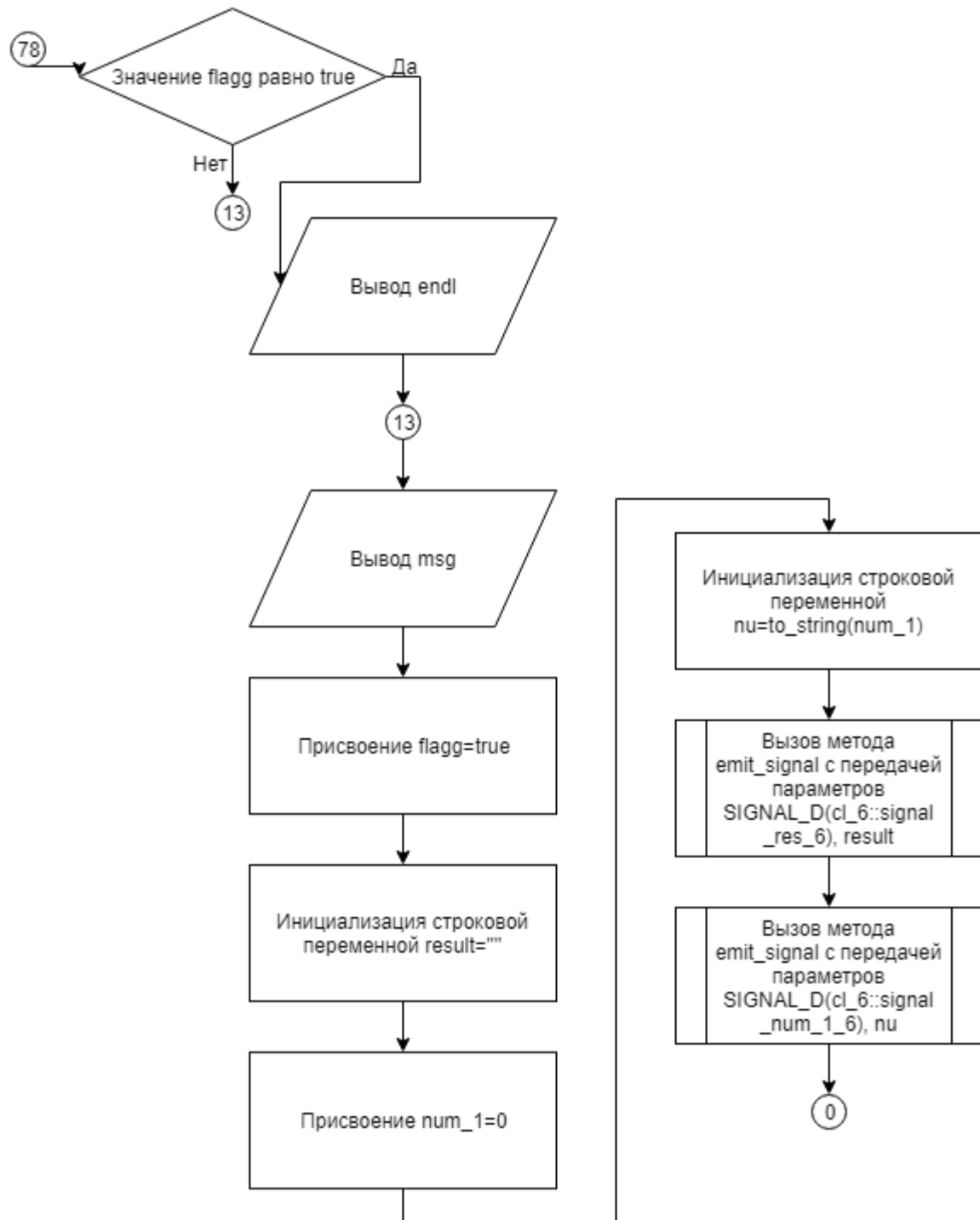


Рисунок 49 – Блок-схема алгоритма



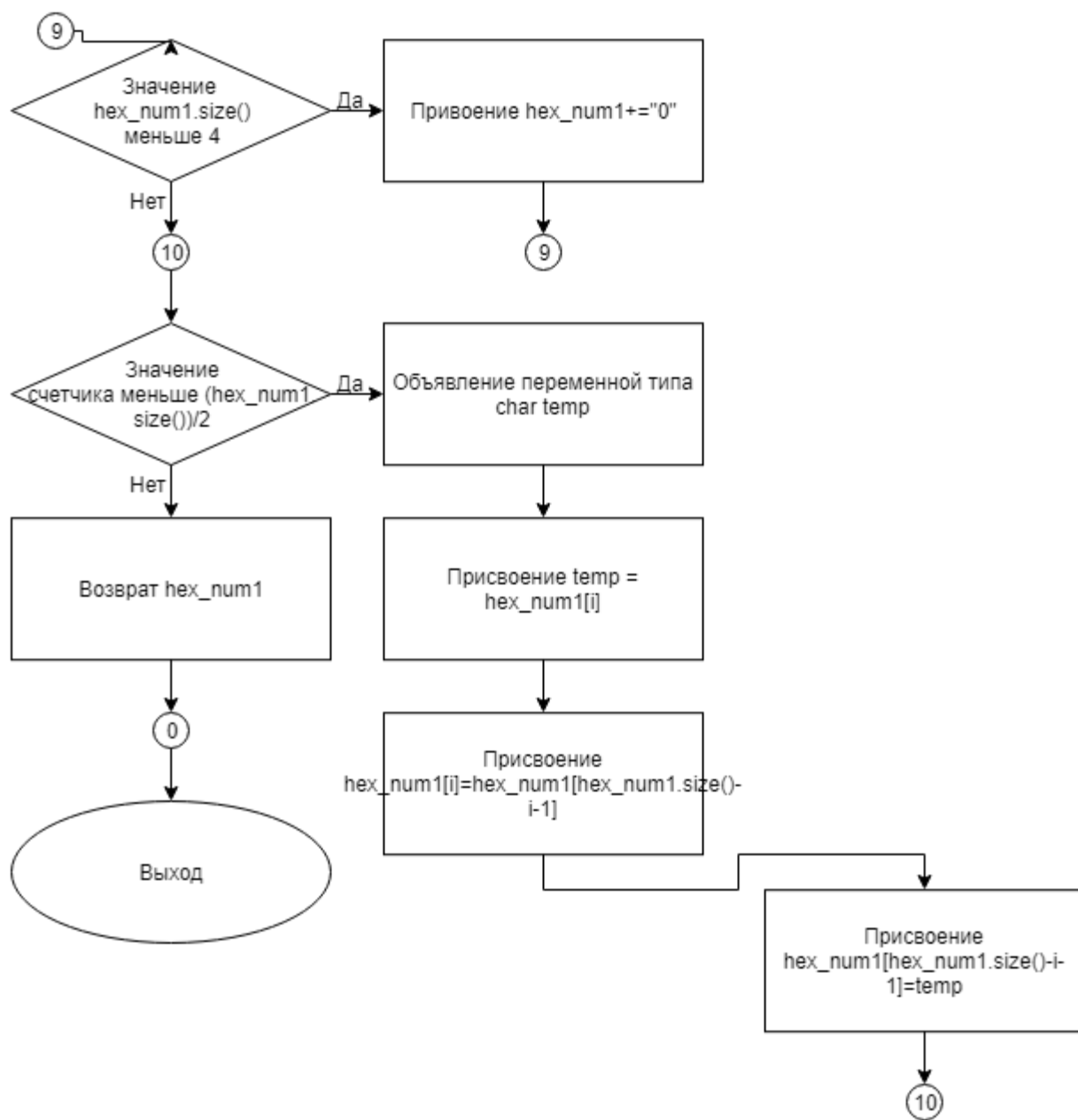
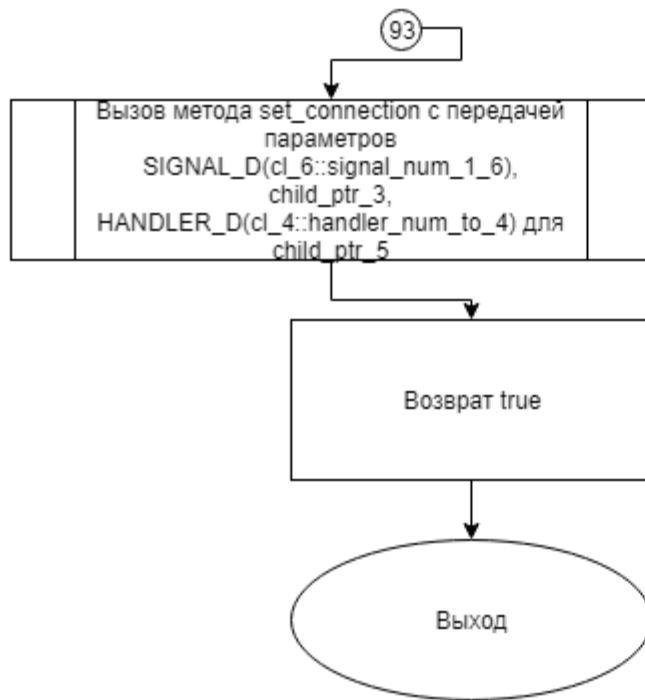


Рисунок 50 – Блок-схема алгоритма



**Рисунок 51 – Блок-схема алгоритма**

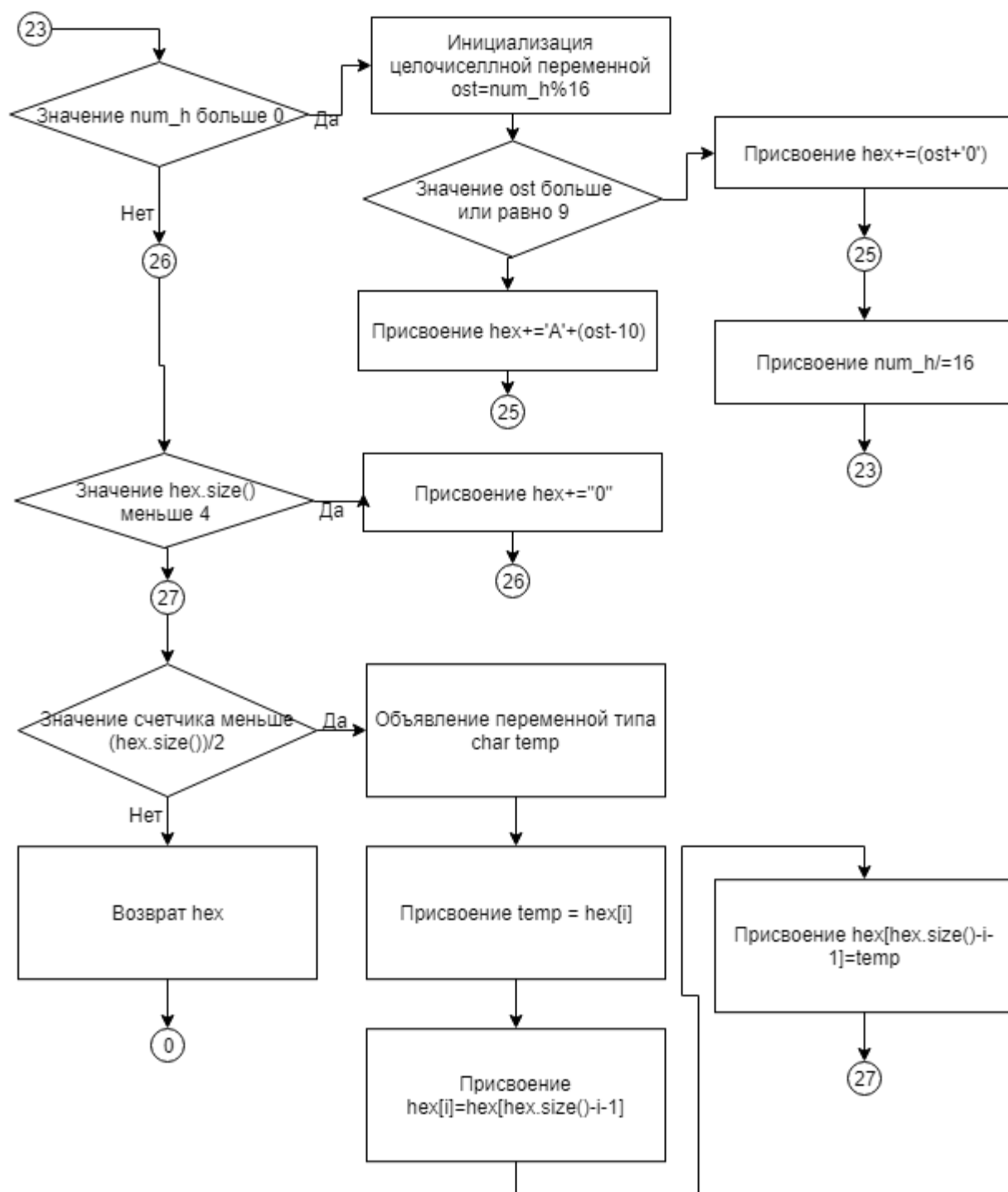


Рисунок 52 – Блок-схема алгоритма

## 5 КОД ПРОГРАММЫ

Программная реализация алгоритмов для решения задачи представлена ниже.

### 5.0 Файл cl\_1.h

#### Листинг 1 – cl\_1.h

```
#ifndef __CL_1_H__
#define __CL_1_H__
#include <iostream>
#include "cl_base.h"
using namespace std;
class cl_1:public cl_base
{
public:
    cl_1(cl_base *parent, string name): cl_base(parent,name){};
};
#endif
```

### 5.1 Файл cl\_2.cpp

#### Листинг 2 – cl\_2.cpp

```
#include <iostream>
#include <string>
#include <vector>
#include <iomanip>
#include "cl_2.h"
using namespace std;

void cl_2::handler_reader(string msg)
{
    string znak;
    cin>>znak;
    emit_signal(SIGNAL_D(cl_2::signal_operation), znak);
}
```

### 5.2 Файл cl\_2.h

#### Листинг 3 – cl\_2.h

```
#ifndef __CL_2_H__
#define __CL_2_H__
#include <iostream>
#include "cl_base.h"
using namespace std;
class cl_2:public cl_base
{
public:
    cl_2(cl_base *parent, string name): cl_base(parent,name){};
    void handler_reader(string msg);
    void signal_operation(string &msg){};
};
```

```
};
#endif
```

## 5.3 Файл cl\_3.cpp

### Листинг 4 – cl\_3.cpp

```
#include <iostream>
#include <string>
#include <vector>
#include <iomanip>
#include "cl_3.h"
using namespace std;

void cl_3::handler_operation(string msg)
{
    bool tem=false;
    if (msg!="Off"&& msg!="C" && (msg=="+" || msg=="-
" || msg=="*" || msg=="/" || msg=="%") && msg!="SHOWTREE")
    {
        operation=msg;
        result+=" "+operation;
        emit_signal(SIGNAL_D(cl_3::signal_res_to_4), result);
        emit_signal(SIGNAL_D(cl_3::signal_oper_to_4), operation);
        flag_C=false;
    }
    else
    {
        if (msg!="Off" && msg!="C"&& msg!="SHOWTREE")
        {
            if (operation!=""&& operation!="<<"&&operation!=">>")
            {
                result+=" "+msg;
                int num_2=stoi(msg);
                if (operation=="+")
                    num_1=plus(num_1, num_2);
                if (operation=="-")
                    num_1=minus(num_1, num_2);
                if (operation=="*")
                    num_1=umn(num_1, num_2);
                if (operation=="/")
                {
                    if (msg!="0")
                        num_1=del(num_1,num_2);
                    else
                    {
                        result+="      Division by zero";
                        emit_signal(SIGNAL_D(cl_3::signal_print),
result);

                        num_1=0;
                        result="0";
                        string numb=to_string(num_1);
                        emit_signal(SIGNAL_D(cl_3::signal_num_to_4),
numb);

                        tem=true;
                        emit_signal(SIGNAL_D(cl_3::signal_res_to_4),
```

```

result);
                                operation="";
                                emit_signal(SIGNAL_D(cl_3::signal_oper_to_4),
operation);
                                }
                                }
                                if (operation=="%")
                                {
                                    if (msg!="0")
                                        num_1=ost(num_1,num_2);
                                    else
                                    {
                                        result+="    Division by zero";
                                        emit_signal(SIGNAL_D(cl_3::signal_print),
result);
                                        num_1=0;
                                        result="0";
                                        string numb=to_string(num_1);
                                        emit_signal(SIGNAL_D(cl_3::signal_num_to_4),
numb);
                                        tem=true;
                                        emit_signal(SIGNAL_D(cl_3::signal_res_to_4),
result);
                                        operation="";
                                        emit_signal(SIGNAL_D(cl_3::signal_oper_to_4),
operation);
                                    }
                                }
                                if (tem==false)
                                {
                                    if (num_1<=32767 && num_1>=-32768)
                                    {
                                        operation="";
                                        string nu=to_string(num_1);
                                        emit_signal(SIGNAL_D(cl_3::signal_num_to_4),
nu);
                                        string prin=result + " " + to_string(num_1);
                                        emit_signal(SIGNAL_D(cl_3::signal_print),
prin);
                                        emit_signal(SIGNAL_D(cl_3::signal_res_to_4),
result);
                                        emit_signal(SIGNAL_D(cl_3::signal_oper_to_4),
operation);
                                    }
                                    else
                                    {
                                        result+="    Overflow";
                                        emit_signal(SIGNAL_D(cl_3::signal_print),
result);
                                        num_1=0;
                                        result="0";
                                        string numb=to_string(num_1);
                                        emit_signal(SIGNAL_D(cl_3::signal_num_to_4),
numb);
                                        tem=true;
                                        emit_signal(SIGNAL_D(cl_3::signal_res_to_4),
result);

```

```

                                operation="";
                                emit_signal(SIGNAL_D(cl_3::signal_oper_to_4),
operation);
                                }
                                }
                                else if (msg!="<<" && msg!=">>"
&&(operation!="<<"&&operation!=">>") &&msg!="SHOWTREE")
                                {
                                    num_1=stoi(msg);
                                    if (result!=""&& operation!="")
                                        result+=" "+to_string(num_1);
                                    else
                                    {
                                        if (flag_C!=false)
                                            result+=msg;
                                        else
                                        {
                                            result=msg;
                                            flag_C=false;
                                        }
                                    }
                                    string nu=to_string(num_1);
                                    emit_signal(SIGNAL_D(cl_3::signal_num_to_4), nu);
                                    emit_signal(SIGNAL_D(cl_3::signal_res_to_4), result);
                                    emit_signal(SIGNAL_D(cl_3::signal_oper_to_4), operation);
                                }
                            }
                        }
}
int cl_3::plus (int num_1, int num_2)
{
    return num_1+num_2;
}

int cl_3::minus (int num_1, int num_2)
{
    return num_1-num_2;
}

int cl_3::umn (int num_1, int num_2)
{
    return num_1*num_2;
}

int cl_3::del(int num_1, int num_2)
{
    return num_1/num_2;
}

int cl_3::ost (int num_1, int num_2)
{
    return num_1%num_2;
}

void cl_3::handler_num_to_3(string msg)
{

```

```

        num_1=stoi(msg);
    }

void cl_3::handler_oper_to_3(string msg)
{
    operation=msg;
}
void cl_3::handler_res_to_3(string msg)
{
    result=msg;
}

```

## 5.4 Файл cl\_3.h

### Листинг 5 – cl\_3.h

```

#ifndef __CL_3_H__
#define __CL_3_H__
#include <iostream>
#include "cl_base.h"
using namespace std;
class cl_3:public cl_base
{
    int num_1;
    string result;
    string operation;
public:
    cl_3(cl_base *parent, string name): cl_base(parent,name){};
    int plus (int num_1, int num_2);
    int minus (int num_1, int num_2);
    int umn (int num_1, int num_2);
    int del (int num_1, int num_2);
    int ost (int num_1, int num_2);
    void handler_operation(string msg);
    void signal_print(string &msg){};
    void signal_num_to_4(string &msg){};
    void signal_oper_to_4(string &msg){};
    void signal_res_to_4(string &msg){};
    void handler_num_to_3(string msg);
    void handler_oper_to_3(string msg);
    void handler_res_to_3(string msg);
};
#endif

```

## 5.5 Файл cl\_4.cpp

### Листинг 6 – cl\_4.cpp

```

#include <iostream>
#include <string>
#include <vector>
#include <iomanip>
#include <cstdlib>
#include "cl_4.h"
using namespace std;

```



```

void cl_4::handler_operation(string msg)
{
    if (msg!="Off"&& msg!="C" && (msg=="<<"|| msg==">>")&& msg!="SHOWTREE")
    {
        operation=msg;
        result+=" "+operation;
        emit_signal(SIGNAL_D(cl_4::signal_res_to_3), result);
        emit_signal(SIGNAL_D(cl_4::signal_oper_to_3), operation);
    }
    else
    {
        if (msg!="Off" && msg!="C" && (msg!="+"&& msg!="-
"&&msg!="*"&&msg!="/"&&msg!="%") && msg!="SHOWTREE")
        {
            if (operation!="")
            {
                result+=" "+msg;
                if (operation=="<<")
                    num_1=sdv_lev(num_1, stoi(msg));
                if (operation==">>")
                    num_1=sdv_prav(num_1, stoi(msg));
                if (num_1<=32767 && num_1>=-32768)
                {
                    operation="";
                    string nu=to_string(num_1);
                    emit_signal(SIGNAL_D(cl_4::signal_num_to_3), nu);
                    emit_signal(SIGNAL_D(cl_4::signal_res_to_3),
result);
                    emit_signal(SIGNAL_D(cl_4::signal_oper_to_3),
operation);
                    string prin=result + " "+ to_string(num_1);
                    emit_signal(SIGNAL_D(cl_4::signal_print), prin);
                }
                else
                {
                    result+="      Overflow";
                    emit_signal(SIGNAL_D(cl_4::signal_print), result);
                    num_1=0;
                    result="0";
                    string numb=to_string(num_1);
                    emit_signal(SIGNAL_D(cl_4::signal_num_to_3), numb);
                    emit_signal(SIGNAL_D(cl_4::signal_res_to_3),
result);
                    operation="";
                    emit_signal(SIGNAL_D(cl_4::signal_oper_to_3),
operation);
                }
            }
        }
    }
}

int cl_4::sdv_lev (int num_1, int num_2)
{
    int num_num=num_1;
    while (num_2>0)
    {
        num_num*=2;

```

```

        num_2--;
    }
    return num_num;
}

int cl_4::sdv_prav (int num_1, int num_2)
{
    int num_num=num_1;
    while (num_2>0)
    {
        num_num/=2;
        num_2--;
    }
    return num_num;
}

void cl_4::handler_num_to_4(string msg)
{
    num_1=stoi(msg);
}

void cl_4::handler_oper_to_4(string msg)
{
    operation=msg;
}

void cl_4::handler_res_to_4(string msg)
{
    result=msg;
}

```

## 5.6 Файл cl\_4.h

### Листинг 7 – cl\_4.h

```

#ifndef __CL_4_H__
#define __CL_4_H__
#include <iostream>
#include "cl_base.h"
using namespace std;
class cl_4:public cl_base
{
    int num_1;
    string result;
    string operation;
public:
    cl_4(cl_base *parent, string name): cl_base(parent,name){};
    int sdv_lev (int num_1, int num_2);
    int sdv_prav (int num_1, int num_2);
    void handler_operation(string msg);
    void signal_print(string &msg){};
    void signal_num_to_3(string &msg){};
    void signal_oper_to_3(string &msg){};
    void signal_res_to_3(string &msg){};
    void handler_num_to_4(string msg);
    void handler_oper_to_4(string msg);
    void handler_res_to_4(string msg);
};
#endif

```

## 5.7 Файл cl\_5.cpp

### Листинг 8 – cl\_5.cpp

```
#include <iostream>
#include <string>
#include <vector>
#include <iomanip>
#include "cl_5.h"
using namespace std;

void cl_5::handler_operation(string msg)
{
    if (msg=="C" && msg!="SHOWTREE")
    {
        result="";
        num_1="0";
        operation = "";
        emit_signal(SIGNAL_D(cl_5::signal_oper_5), operation);
        emit_signal(SIGNAL_D(cl_5::signal_num_5), num_1);
        emit_signal(SIGNAL_D(cl_5::signal_res_5), result);
        flag_C=true;
    }
}
```

## 5.8 Файл cl\_5.h

### Листинг 9 – cl\_5.h

```
#ifndef __CL_5_H__
#define __CL_5_H__
#include <iostream>
#include "cl_base.h"
using namespace std;
class cl_5:public cl_base
{
    string result;
    string num_1;
    string operation;
public:
    cl_5(cl_base *parent, string name): cl_base(parent,name){};
    void handler_operation(string msg);
    void signal_res_5(string &msg){};
    void signal_num_5(string &msg){};
    void signal_oper_5(string &msg){};
};
#endif
```

## 5.9 Файл cl\_6.cpp

### Листинг 10 – cl\_6.cpp

```
#include <iostream>
```

```

#include <string>
#include <vector>
#include <iomanip>
#include <cmath>
#include "cl_6.h"
using namespace std;

void cl_6::handler_print(string msg)
{
    if(msg.size()>8)
    {
        if (msg.substr(msg.size()-8)=="Overflow")
        {
            if (flagg==true)
                cout<<endl;
            cout<<msg;
            flagg=true;
            string result="";
            num_1=0;
            string nu=to_string(num_1);
            emit_signal(SIGNAL_D(cl_6::signal_res_6), result);
            emit_signal(SIGNAL_D(cl_6::signal_num_1_6), nu);
        }
        else if (msg.substr(msg.size()-4)=="zero")
        {
            if (flagg==true)
                cout<<endl;
            cout<<msg;
            string result="";
            num_1=0;
            string nu=to_string(num_1);
            emit_signal(SIGNAL_D(cl_6::signal_res_6), result);
            emit_signal(SIGNAL_D(cl_6::signal_num_1_6), nu);
            flagg=true;
        }
        else
        {
            int ind=msg.rfind(" ");
            int num_print=stoi(msg.substr(ind+1));
            if (flagg==true)
                cout<<endl;
            cout<<msg.substr(0, ind)<<"          "<<"HEX  "<<HEX_16(num_print)<<"
"<<"DEC  "<<num_print<<"  "<<"BIN  "<<BIN_2(num_print);
            flagg=true;
        }
    }
    else
    {
        int ind=msg.rfind(" ");
        int num_print=stoi(msg.substr(ind+1));
        if (flagg==true)
            cout<<endl;
        cout<<msg.substr(0, ind)<<"          "<<"HEX  "<<HEX_16(num_print)<<"
"<<"DEC  "<<num_print<<"  "<<"BIN  "<<BIN_2(num_print);
        flagg=true;
    }
}

```

```

string cl_6::BIN_2(int number)
{
    bool temp=true;
    if (number<0)
        temp=false;
    int bin_number=0, k=1, num=abs(number);
    string num_BIN;
    while (num>0)
    {
        num_BIN+=to_string(num%2);
        num/=2;
    }
    while (num_BIN.size()<16)
    {
        num_BIN+="0";
    }
    num_BIN.insert(4, 1, ' ');
    num_BIN.insert(9, 1, ' ');
    num_BIN.insert(14, 1, ' ');
    for (int i=0; i<(num_BIN.size())/2;i++)
    {
        char temp;
        temp =num_BIN[i];
        num_BIN[i]=num_BIN[num_BIN.size()-i-1];
        num_BIN[num_BIN.size()-i-1]=temp;
    }
    if (temp==true)
        return num_BIN;
    else
    {
        int ind_1=num_BIN.rfind("1");
        if (ind_1!=-1)
        {
            for (int i=0; i<ind_1; i++)
            {
                if (num_BIN[i]=='1')
                    num_BIN[i]='0';
                else if (num_BIN[i]=='0')
                    num_BIN[i]='1';
            }
            return num_BIN;
        }
        else return num_BIN;
    }
}

string cl_6::HEX_16(int number)
{
    bool tem=true;
    if (number<0)
        tem=false;
    if (tem==true)
    {
        string hex_num1="", hex_number;
        int num=abs(number);
        while (num>0)
        {

```

```

        int ost=num%16;
        if (ost<=9)
            hex_num1+=(ost+'0');
        else
        {
            hex_num1+='A'+(ost-10);
        }
        num/=16;
    }
    while (hex_num1.size()<4)
        hex_num1+="0";
    for (int i=0; i<(hex_num1.size())/2;i++)
    {
        char temp;
        temp = hex_num1[i];
        hex_num1[i]=hex_num1[hex_num1.size()-i-1];
        hex_num1[hex_num1.size()-i-1]=temp;
    }
    return hex_num1;
}
else
{
    int num_h=0;
    string hex="";
    string numb=BIN_2(number);
    numb.erase(14, 1);
    numb.erase(9, 1);
    numb.erase(4, 1);
    for (int i=0; i<numb.size()-1; i++)
    {
        if (numb[i]=='1')
        {
            num_h+=pow(2, numb.size()-i-1);
        }
    }
    while (num_h>0)
    {
        int ost=num_h%16;
        if (ost<=9)
            hex+=(ost+'0');
        else
        {
            hex+='A'+(ost-10);
        }
        num_h/=16;
    }
    while (hex.size()<4)
        hex+="0";
    for (int i=0; i<(hex.size())/2;i++)
    {
        char temp;
        temp = hex[i];
        hex[i]=hex[hex.size()-i-1];
        hex[hex.size()-i-1]=temp;
    }
    return hex;
}

```

```
}
```

## 5.10 Файл cl\_6.h

### Листинг 11 – cl\_6.h

```
#ifndef __CL_6_H__
#define __CL_6_H__
#include <iostream>
#include <string>
#include "cl_base.h"
using namespace std;
class cl_6:public cl_base
{
    bool flagg=false;
    int num_1;
public:
    cl_6(cl_base *parent, string name): cl_base(parent,name){};
    void handler_print(string msg);
    string HEX_16(int number);
    string BIN_2(int number);
    void signal_res_6(string &msg){};
    void signal_num_1_6(string &msg){};
};
#endif
```

## 5.11 Файл cl\_application.cpp

### Листинг 12 – cl\_application.cpp

```
#include <iostream>
#include <string>
#include <stdio.h>
#include "cl_application.h"
#include "cl_base.h"
#include "cl_2.h"
#include "cl_3.h"
#include "cl_4.h"
#include "cl_5.h"
#include "cl_6.h"
using namespace std;

bool cl_application::build_tree_objects()
{
    cl_base *parent_ptr, *child_ptr_1, *child_ptr_2, *child_ptr_3, *child_ptr_4,
    *child_ptr_5;
    string parent_name, child_name, name_cin, name_1;
    parent_name="SYSTEM";
    set_name(parent_name);
    parent_ptr=this;
    parent_ptr->set_state(1);
    child_ptr_1 = new cl_2(parent_ptr, "READER");
    child_ptr_1->set_state(1);
    child_ptr_2 = new cl_3(parent_ptr, "OPERATION");
```

```

    child_ptr_2->set_state(1);
    child_ptr_3 = new cl_4(parent_ptr, "BIN_SDV");
    child_ptr_3->set_state(1);
    child_ptr_4 = new cl_5(parent_ptr, "COM_C");
    child_ptr_4->set_state(1);
    child_ptr_5 = new cl_6(parent_ptr, "PRINT");
    child_ptr_5->set_state(1);

    set_connection(SIGNAL_D(cl_application::signal_reader),      child_ptr_1,
HANDLER_D(cl_2::handler_reader));
    child_ptr_1->set_connection(SIGNAL_D(cl_2::signal_operation), child_ptr_2,
HANDLER_D(cl_3::handler_operation));
    child_ptr_1->set_connection(SIGNAL_D(cl_2::signal_operation), child_ptr_3,
HANDLER_D(cl_4::handler_operation));
    child_ptr_1->set_connection(SIGNAL_D(cl_2::signal_operation), child_ptr_4,
HANDLER_D(cl_5::handler_operation));
    child_ptr_1->set_connection(SIGNAL_D(cl_2::signal_operation),      this,
HANDLER_D(cl_application::handler_operation));

    child_ptr_2->set_connection(SIGNAL_D(cl_3::signal_num_to_4),  child_ptr_3,
HANDLER_D(cl_4::handler_num_to_4));
    child_ptr_3->set_connection(SIGNAL_D(cl_4::signal_num_to_3),  child_ptr_2,
HANDLER_D(cl_3::handler_num_to_3));

    child_ptr_2->set_connection(SIGNAL_D(cl_3::signal_oper_to_4), child_ptr_3,
HANDLER_D(cl_4::handler_oper_to_4));
    child_ptr_3->set_connection(SIGNAL_D(cl_4::signal_oper_to_3), child_ptr_2,
HANDLER_D(cl_3::handler_oper_to_3));

    child_ptr_2->set_connection(SIGNAL_D(cl_3::signal_res_to_4),  child_ptr_3,
HANDLER_D(cl_4::handler_res_to_4));
    child_ptr_3->set_connection(SIGNAL_D(cl_4::signal_res_to_3),  child_ptr_2,
HANDLER_D(cl_3::handler_res_to_3));

    child_ptr_2->set_connection(SIGNAL_D(cl_3::signal_print),     child_ptr_5,
HANDLER_D(cl_6::handler_print));
    child_ptr_3->set_connection(SIGNAL_D(cl_4::signal_print),     child_ptr_5,
HANDLER_D(cl_6::handler_print));

    child_ptr_4->set_connection(SIGNAL_D(cl_5::signal_res_5),     child_ptr_2,
HANDLER_D(cl_3::handler_res_to_3));
    child_ptr_4->set_connection(SIGNAL_D(cl_5::signal_res_5),     child_ptr_3,
HANDLER_D(cl_4::handler_res_to_4));

    child_ptr_4->set_connection(SIGNAL_D(cl_5::signal_num_5),     child_ptr_2,
HANDLER_D(cl_3::handler_num_to_3));
    child_ptr_4->set_connection(SIGNAL_D(cl_5::signal_num_5),     child_ptr_3,
HANDLER_D(cl_4::handler_num_to_4));

    child_ptr_4->set_connection(SIGNAL_D(cl_5::signal_oper_5),    child_ptr_2,
HANDLER_D(cl_3::handler_oper_to_3));
    child_ptr_4->set_connection(SIGNAL_D(cl_5::signal_oper_5),    child_ptr_3,
HANDLER_D(cl_4::handler_oper_to_4));

    child_ptr_5->set_connection(SIGNAL_D(cl_6::signal_res_6),     child_ptr_2,
HANDLER_D(cl_3::handler_res_to_3));
    child_ptr_5->set_connection(SIGNAL_D(cl_6::signal_res_6),     child_ptr_3,

```



```

HANDLER_D(cl_4::handler_res_to_4));

    child_ptr_5->set_connection(SIGNAL_D(cl_6::signal_num_1_6),      child_ptr_2,
HANDLER_D(cl_3::handler_num_to_3));
    child_ptr_5->set_connection(SIGNAL_D(cl_6::signal_num_1_6),      child_ptr_3,
HANDLER_D(cl_4::handler_num_to_4));
    return true;
}
int cl_application::exec_app(bool b)
{
    string ms="";
    flag_input=true;
    while(flag_input)
    {
        emit_signal(SIGNAL_D(cl_application::signal_reader), ms);
    }
    return 0;
}

void cl_application::handler_operation(string msg)
{
    if (msg=="Overflow")
    {
        flag_input=false;
    }
    else if (msg=="SHOWTREE")
    {
        cout<<"Object tree";
        flag_input=false;
        cout<<endl<<"SYSTEM";
        if (get_ptr_by_name("SYSTEM")->get_state()==1)
            cout<<" is ready";
        else cout<<" is not ready";
        print_state(1);
    }
    else if (msg=="Off")
    {
        flag_input=false;
    }
}

```

## 5.12 Файл cl\_application.h

### Листинг 13 – cl\_application.h

```

#ifndef __CL_APPLICATION_H__
#define __CL_APPLICATION_H__
#include <iostream>
#include "cl_base.h"
#include "cl_1.h"
using namespace std;
class cl_application:public cl_base
{
    bool flag_input;
public:
    cl_application(cl_base*parent):cl_base(parent){};

```

```

        bool build_tree_objects();
        int exec_app(bool b);
        void signal_reader(string &msg){};
        void handler_operation(string msg);
};
#endif

```

## 5.13 Файл cl\_base.cpp

### Листинг 14 – cl\_base.cpp

```

#include <iostream>
#include <string>
#include <vector>
#include <iomanip>
#include <cmath>
#include "cl_base.h"
using namespace std;

cl_base :: cl_base(cl_base *parent, string name)
{
    this->name=name;
    this->parent = parent;
    if(parent !=nullptr)
    {
        parent->children.push_back(this);
    }
}

cl_base :: ~cl_base()
{
    for(int i=0; i<children.size(); i++)
    {
        delete children[i];
    }
    for(int i=0; i<connections.size(); i++)
    {
        delete connections[i];
    }
}

void cl_base::set_parent(cl_base * new_parent)
{
    if(parent!=nullptr)
    {
        for(int i=0; i<parent->children.size(); i++)
        {
            if (parent->children[i]==this)
            {
                parent->children.erase(parent->children.begin()+i);
                break;
            }
        }
    }
    parent = new_parent;
    if(parent!=nullptr)

```

```

        {
            parent->children.push_back(this);
        }
    }

string cl_base::get_name()
{
    return name;
}

void cl_base::set_name(string name1)
{
    name=name1;
}

cl_base * cl_base::get_parent()
{
    return parent;
}

cl_base * cl_base::get_ptr_by_name(string name)
{
    if(this->get_name()==name)
    {
        return this;
    }
    for (int i=0; i<this->children.size(); i++)
    {
        cl_base* buff = children[i]->get_ptr_by_name(name);
        if(buff!=nullptr)
            return buff;
    }
    return nullptr;
}

int cl_base::get_state()
{
    return state;
}

void cl_base::set_state(int state1)
{
    if (parent!=nullptr)
    {
        if(parent->get_state()!=0 && state1!=0)
            state=state1;
        else
        {
            state=0;
            for (int i=0; i < this->children.size(); i++)
            {
                children[i]->set_state(0);
            }
        }
    }
    else
    {

```

```

        state=state1;
        if (state==0)
        {
            for (int i=0; i < this->children.size(); i++)
            {
                children[i]->set_state(0);
            }
        }
    }
}

void cl_base::print_state(int num_rec2)
{
    num_rec=num_rec2;
    if (children.size()!=0)
    {
        for (int i=0; i<children.size(); i++)
        {
            string len=children[i]->get_name();
            if((children[i]->get_state())==0)
                len+=" is not ready";
            else len+=" is ready";
            cout<<endl<<setw(len.size()+4*num_rec)<<len;
            num_rec++;
            children[i]->print_state(num_rec);
            num_rec--;
        }
    }
}

void cl_base::set_connection(TYPE_SIGNAL signal, cl_base* target_obj, TYPE_HANDLER
handler)
{
    o_sh* connection=new o_sh(signal, target_obj, handler);
    connections.push_back(connection);
}

void cl_base::delete_connection(TYPE_SIGNAL signal, cl_base* target_obj,
TYPE_HANDLER handler)
{
    for(int i=0; i<this->connections.size(); i++)
    {
        if ((connections[i]->signal)==signal && (connections[i])-
>target_obj==target_obj)
        {
            this->connections.erase(this->connections.begin()+i);
        }
    }
}

void cl_base::emit_signal(TYPE_SIGNAL signal, string& msg)
{
    cl_base* target_obj_ptr;
    TYPE_HANDLER handler_obj;
    if (this->get_state()!=0)
    {
        (this->*signal) (msg);
    }
}

```

```

        for(int i=0; i<this->connections.size(); i++)
        {
            if (connections[i]->signal==signal)
            {
                target_obj_ptr= connections[i]-> target_obj;
                handler_obj= connections[i]-> handler;
                if (target_obj_ptr->get_state())
                    (target_obj_ptr->*handler_obj) (msg);
            }
        }
    }
}

```

## 5.14 Файл cl\_base.h

### Листинг 15 – cl\_base.h

```

#ifndef __CL_BASE_H__
#define __CL_BASE_H__
#include <iostream>
#include <string>
#include <vector>
using namespace std;
class cl_base;
typedef void(cl_base::*TYPE_SIGNAL) (string&);
typedef void(cl_base::*TYPE_HANDLER) (string);
#define SIGNAL_D(signal_f) (TYPE_SIGNAL) (&signal_f)
#define HANDLER_D(handler_f) (TYPE_HANDLER) (&handler_f)

class cl_base
{
private:
    struct o_sh
    {
        TYPE_SIGNAL signal;
        cl_base* target_obj;
        TYPE_HANDLER handler;
        o_sh (TYPE_SIGNAL signal, cl_base* target_obj, TYPE_HANDLER handler):
            signal(signal), target_obj(target_obj), handler(handler){};
    };
private:
    string name;
    cl_base* parent;
    vector <cl_base*> children;
    int num_rec;
    int state=1;
    vector <o_sh*> connections;
protected:
    bool flag_C=false;
public:
    cl_base(cl_base *parent, string name="");
    ~cl_base();
    string get_name();
    void set_name(string name);
    cl_base * get_parent();
    void set_parent(cl_base *new_parent);

```

```

        cl_base * get_ptr_by_name (string name);
        int get_state();
        void set_state(int state1);
        void print_state(int num_rec2);
        void set_connection(TYPE_SIGNAL signal, cl_base* target_obj, TYPE_HANDLER
handler);
        void delete_connection (TYPE_SIGNAL signal, cl_base* target_obj,
TYPE_HANDLER handler);
        void emit_signal(TYPE_SIGNAL signal, string& msg);
};
#endif

```

## 5.15 Файл main.cpp

### Листинг 16 – main.cpp

```

#include "cl_application.h"
using namespace std;
int main()
{
    cl_application cl_application_obj(nullptr);
    bool b;
    b=cl_application_obj.build_tree_objects();
    return cl_application_obj.exec_app(b);
}

```

## 6 ТЕСТИРОВАНИЕ

Результат тестирования программы представлен в таблице 61.

Таблица 61 – Результат тестирования программы

Входные данные	Ожидаемые выходные данные	Фактические выходные данные
5 + 5 << 1 / 0 + 5 C 7 8 / -3 C 9 % -4 + 7 * 11 Off	5 + 5      HEX 000A    DEC 10 BIN 0000 0000 0000 1010 5 + 5 << 1      HEX 0014 DEC 20    BIN 0000 0000 0001 0100 5 + 5 << 1 / 0 Division by zero 0 + 5      HEX 0005    DEC 5 BIN 0000 0000 0000 0101 8 / -3      HEX FFFE    DEC - 2    BIN 1111 1111 1111 1110 9 % -4      HEX 0001    DEC 1 BIN 0000 0000 0000 0001 9 % -4 + 7      HEX 0008 DEC 8    BIN 0000 0000 0000 1000 9 % -4 + 7 * 11      HEX 0058    DEC 88    BIN 0000 0000 0101 1000	5 + 5      HEX 000A    DEC 10 BIN 0000 0000 0000 1010 5 + 5 << 1      HEX 0014 DEC 20    BIN 0000 0000 0001 0100 5 + 5 << 1 / 0 Division by zero 0 + 5      HEX 0005    DEC 5 BIN 0000 0000 0000 0101 8 / -3      HEX FFFE    DEC - 2    BIN 1111 1111 1111 1110 9 % -4      HEX 0001    DEC 1 BIN 0000 0000 0000 0001 9 % -4 + 7      HEX 0008 DEC 8    BIN 0000 0000 0000 1000 9 % -4 + 7 * 11      HEX 0058    DEC 88    BIN 0000 0000 0101 1000
5 7 + 5 << 1 >> 2 / 0 + 5 C 7 8 % 0 C 9 + 32767 % -4 + 7 * 11 C 1 - 32770 5 * 5 / 5 / 3 Off	7 + 5      HEX 000C    DEC 12 BIN 0000 0000 0000 1100 7 + 5 << 1      HEX 0018 DEC 24    BIN 0000 0000 0001 1000 7 + 5 << 1 >> 2      HEX 0006    DEC 6    BIN 0000 0000 0000 0110 7 + 5 << 1 >> 2 / 0 Division by zero 0 + 5      HEX 0005    DEC 5 BIN 0000 0000 0000 0101 8 % 0      Division by zero 9 + 32767      Overflow 0 % -4      HEX 0000    DEC 0 BIN 0000 0000 0000 0000 0 % -4 + 7      HEX 0007 DEC 7    BIN 0000 0000 0000 0111 0 % -4 + 7 * 11      HEX 004D    DEC 77    BIN 0000 0000 0100 1101 1 - 32770      Overflow 5 * 5      HEX 0019    DEC 25 BIN 0000 0000 0001 1001 5 * 5 / 5      HEX 0005 DEC 5    BIN 0000 0000 0000 0101	7 + 5      HEX 000C    DEC 12 BIN 0000 0000 0000 1100 7 + 5 << 1      HEX 0018 DEC 24    BIN 0000 0000 0001 1000 7 + 5 << 1 >> 2      HEX 0006    DEC 6    BIN 0000 0000 0000 0110 7 + 5 << 1 >> 2 / 0 Division by zero 0 + 5      HEX 0005    DEC 5 BIN 0000 0000 0000 0101 8 % 0      Division by zero 9 + 32767      Overflow 0 % -4      HEX 0000    DEC 0 BIN 0000 0000 0000 0000 0 % -4 + 7      HEX 0007 DEC 7    BIN 0000 0000 0000 0111 0 % -4 + 7 * 11      HEX 004D    DEC 77    BIN 0000 0000 0100 1101 1 - 32770      Overflow 5 * 5      HEX 0019    DEC 25 BIN 0000 0000 0001 1001 5 * 5 / 5      HEX 0005 DEC 5    BIN 0000 0000 0000 0101

Входные данные	Ожидаемые выходные данные	Фактические выходные данные
	5 * 5 / 5 / 3      HEX 0001 DEC 1    BIN 0000 0000 0000 0001	5 * 5 / 5 / 3      HEX 0001 DEC 1    BIN 0000 0000 0000 0001



## **ЗАКЛЮЧЕНИЕ**

В процессе изучения ООП во втором семестре я научилась таким парадигмам ООП, как наследование, инкапсуляция, полиморфизм. Я научилась использовать классы и дружественные функции, создавать методы и макросы для реализации сигналов и обработчиков.

Полученные знания помогут мне в будущем для анализа "больших данных" в различных сферах специальности "Прикладная математика".

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Васильев А.Н. Объектно-ориентированное программирование на C++. Издательство: Наука и Техника. Санкт-Петербург, 2016г. 543 стр.
2. Шилдт Г. C++: базовый курс. 3-е изд. Пер. с англ.. — М.: Вильямс, 2017. — 624 с.
3. Методическое пособие для проведения практических заданий, контрольных и курсовых работ по дисциплине «Объектно-ориентированное программирование» [Электронный ресурс] – URL: [https://mirea.aco-avrrora.ru/student/files/methodicheskoe\\_posobie\\_dlya\\_laboratornyh\\_rabot\\_3.pdf](https://mirea.aco-avrrora.ru/student/files/methodicheskoe_posobie_dlya_laboratornyh_rabot_3.pdf) (дата обращения 05.05.2021).
4. Приложение к методическому пособию студента по выполнению заданий в рамках курса «Объектно-ориентированное программирование» [Электронный ресурс]. URL: [https://mirea.aco-avrrora.ru/student/files/Prilozheniye\\_k\\_methodichke.pdf](https://mirea.aco-avrrora.ru/student/files/Prilozheniye_k_methodichke.pdf) (дата обращения 05.05.2021).
5. Видео лекции по курсу «Объектно-ориентированное программирование» [Электронный ресурс]. АСО «Аврора».
6. Антик М.И. Дискретная математика [Электронный ресурс]: Учебное пособие /Антик М.И., Казанцева Л.В. — М.: МИРЭА — Российский технологический университет, 2018 — 1 электрон. опт. диск (CD-ROM).