



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт информационных технологий
Кафедра вычислительной техники

КУРСОВАЯ РАБОТА

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

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

Студент группы ИКБО-20-23 Коротков Алексей Александрович
(учебная группа) (Фамилия Имя Отчество) (подпись студента)

Руководитель курсовой работы доцент Лозовский В.В.
(Должность, звание, ученая степень) (подпись руководителя)

Консультант доцент Штрекер Е.Н.
(Должность, звание, ученая степень) (подпись консультанта)

Работа представлена к защите «31» мая 2024 г.

Допущен к защите «31» мая 2024 г.

Отлично

Москва 2024 г.



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт информационных технологий
Кафедра вычислительной техники

Утверждаю

Заведующий кафедрой

Подпись

Платонова О.В.

ФИО

«21» февраля 2024г.

ЗАДАНИЕ

На выполнение курсовой работы

по дисциплине «Объектно-ориентированное программирование»

Студент Коротков Алексей Александрович Группа ИКБО-20-23

Тема К_3 Моделирование работы инженерного арифметического калькулятора

Исходные данные:

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

Перечень вопросов, подлежащих разработке, и обязательного графического материала:

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

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

Задание на курсовую работу выдал

Подпись

(Лозовский В.В.)

ФИО консультанта

«21» февраля 2024 г.

Задание на курсовую работу получил

Подпись

(Коротков А.А.)

ФИО исполнителя

«21» февраля 2024 г.

Москва 2024 г.

ОТЗЫВ

на курсовую работу

по дисциплине «Объектно-ориентированное программирование»

Студент Коротков Алексей Александрович группа ИКБО-20-23
(ФИО студента) (Группа)

Характеристика курсовой работы

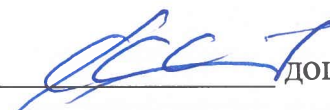
Критерий	Да	Нет	Не полностью
1. Соответствие содержания курсовой работы указанной теме	✓		
2. Соответствие курсовой работы заданию	✓		
3. Соответствие рекомендациям по оформлению текста, таблиц, рисунков и пр.	✓		
4. Полнота выполнения всех пунктов задания	✓		
5. Логичность и системность содержания курсовой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

Замечаний:

нет

Рекомендуемая оценка:

отлично



(Подпись руководителя)

доцент Лозовский В.В.

(ФИО руководителя)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	6
1 ПОСТАНОВКА ЗАДАЧИ.....	7
1.1 Описание входных данных.....	9
1.2 Описание выходных данных.....	10
2 МЕТОД РЕШЕНИЯ.....	11
3 ОПИСАНИЕ АЛГОРИТМОВ.....	13
3.1 Алгоритм метода signal_calc_to_screen класса cl_1.....	13
3.2 Алгоритм метода handler_calc_from_reader класса cl_1.....	13
3.3 Алгоритм метода handler_cancel_from_reader класса cl_2.....	14
3.4 Алгоритм метода handler_reader_from_app класса cl_3.....	15
3.5 Алгоритм метода signal_reader_to_all класса cl_3.....	16
3.6 Алгоритм метода handler_screen_from_all класса cl_4.....	17
3.7 Алгоритм метода toBinary класса cl_4.....	18
3.8 Алгоритм метода handler_shift_from_reader класса cl_5.....	18
3.9 Алгоритм метода signal_shift_to_screen класса cl_5.....	19
3.10 Алгоритм метода build_tree_objects класса cl_application.....	19
3.11 Алгоритм метода exec_app класса cl_application.....	21
3.12 Алгоритм функции main.....	21
4 БЛОК-СХЕМЫ АЛГОРИТМОВ.....	22
5 КОД ПРОГРАММЫ.....	32
5.1 Файл cl_1.cpp.....	32
5.2 Файл cl_1.h.....	33
5.3 Файл cl_2.cpp.....	34
5.4 Файл cl_2.h.....	34
5.5 Файл cl_3.cpp.....	35
5.6 Файл cl_3.h.....	36

5.7 Файл cl_4.cpp.....	36
5.8 Файл cl_4.h.....	38
5.9 Файл cl_5.cpp.....	38
5.10 Файл cl_5.h.....	39
5.11 Файл cl_application.cpp.....	39
5.12 Файл cl_application.h.....	41
5.13 Файл cl_base.cpp.....	41
5.14 Файл cl_base.h.....	46
5.15 Файл main.cpp.....	47
6 ТЕСТИРОВАНИЕ.....	48
ЗАКЛЮЧЕНИЕ.....	49
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....	50

ВВЕДЕНИЕ

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

В современном мире программирования объектно-ориентированное программирование (ООП) играет ключевую роль в разработке сложных структурированных программных систем. Язык программирования C++, сочетающий в себе выразительность языка высокого уровня и эффективность языка низкого уровня, является одним из наиболее распространенных инструментов для реализации принципов ООП. В данной курсовой работе я рассмотрю моделирование работы инженерного арифметического калькулятора с использованием сигналов и обработчиков на языке C++. Инженерные калькуляторы являются важными инструментами для инженеров и специалистов в области точных наук, обеспечивая выполнение математических операций с высокой эффективностью. Целью данной работы является разработка программы, которая моделирует работу инженерного арифметического калькулятора, позволяя пользователю выполнять различные арифметические операции с числами, а также обрабатывать сигналы и ошибки при выполнении этих операций. Для этого я буду использовать основные принципы ООП, такие как инкапсуляция, наследование и полиморфизм, а также сигналы и обработчики в C++.

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 “С” 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:

- «целое число» - первый аргумент выражения, целое не отрицательное число, можно последовательно вводить несколько раз, предыдущее значение меняется. При вводе не первым аргументом выражения - игнорируется;
- «знак операции» «целое число» - второе и последующие операции выражения;
- «С» - приведение калькулятора в исходное состояние;
- «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:

1. Объект «система».
2. Объект для чтения команд. После чтения очередной команды объект выдает сигнал с текстом, содержащим команду. Все команды синтаксический корректны (моделирует пульт управления калькулятора).
3. 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.
4. Объект для выполнения операции побитового сдвига. После завершения выдается сигнал с текстом результата.
5. Объект для выполнения операции «С».
6. 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» `exes_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. Отработка 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

Класс cl_1:

- функционал:
 - метод signal_calc_to_screen — сигнал для вывода;
 - метод handler_calc_from_reader — обработчик сигнала из ввода.

Класс cl_2:

- функционал:
 - метод handler_cancel_from_reader — обработчик сигнала ввода.

Класс cl_3:

- функционал:
 - метод handler_reader_from_app — обработчик сигнала из программы;
 - метод signal_reader_to_all — сигнал для других обработчиков.

Класс cl_4:

- функционал:
 - метод handler_screen_from_all — обработчик сигналов;
 - метод toBinary — преобразование в двоичную сс.

Класс cl_5:

- функционал:
 - метод handler_shift_from_reader — обработчик сигнала из ввода;
 - метод signal_shift_to_screen — сигнал для вывода.

Таблица 1 – Иерархия наследования классов

№	Имя класса	Классы-наследники	Модификатор доступа при наследовании	Описание	Номер
1	cl_1			класс, отвечающий за арифметический счет	
2	cl_2			класс, отвечающий за сброс	

№	Имя класса	Классы-наследники	Модификатор доступа при наследовании	Описание	Номер
3	cl_3			класс, отвечающий за ввод	
4	cl_4			класс, отвечающий за вывод	
5	cl_5			класс, отвечающий за побитовый сдвиг	

3 ОПИСАНИЕ АЛГОРИТМОВ

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

3.1 Алгоритм метода `signal_calc_to_screen` класса `cl_1`

Функционал: сигнал для вывода.

Параметры: `string& msg`.

Возвращаемое значение: `void`.

Алгоритм метода представлен в таблице 2.

Таблица 2 – Алгоритм метода `signal_calc_to_screen` класса `cl_1`

№	Предикат	Действия	№ перехода
1			Ø

3.2 Алгоритм метода `handler_calc_from_reader` класса `cl_1`

Функционал: обработчик сигнала из ввода.

Параметры: `string msg`.

Возвращаемое значение: `void`.

Алгоритм метода представлен в таблице 3.

Таблица 3 – Алгоритм метода `handler_calc_from_reader` класса `cl_1`

№	Предикат	Действия	№ перехода
1	<code>msg == "+"</code>	присвоение полю <code>i_result</code> объекта по указателю <code>get_head</code> значения суммы поля <code>i_result</code> и <code>s_operand_2</code>	8

№	Предикат	Действия	№ перехода
			2
2	msg == "-"	присвоение полю i_result объекта по указателю get_head значения разности поля i_result и s_operand_2	8
			3
3	msg == "*"	присвоение полю i_result объекта по указателю get_head значения произведения поля i_result и s_operand_2	8
			4
4	msg == "/"		5
			7
5	s_operand!=0	присвоение полю i_result объекта по указателю get_head значения деления поля i_result на s_operand_2	8
		присвоение полю объекта по указателю get_head s_operand_2 значения "Division by zero"	6
6		присвоение i_result объекта по указателю get_head значения 0	8
7	msg == "%"	присвоение полю i_result объекта по указателю get_head значения остатка от деления поля i_result на s_operand_2	8
			8
8		вызов метода emit_signal с методом сигнала signal_calc_to_screen и параметром i_result	∅

3.3 Алгоритм метода handler_cancel_from_reader класса cl_2

Функционал: обработчик сигнала ввода.

Параметры: string msg.

Возвращаемое значение: void.

Алгоритм метода представлен в таблице 4.

Таблица 4 – Алгоритм метода *handler_cancel_from_reader* класса *cl_2*

№	Предикат	Действия	№ перехода
1	msg=="C"	присовение полю s_expression объекта по указателю get_head значения "0"	2
			Ø
2		присовение полю s_operation объекта по указателю get_head значения ""	3
3		присовение полю s_operand2 объекта по указателю get_head значения "C"	4
4		присовение полю i_result объекта по указателю get_head значения 0	Ø

3.4 Алгоритм метода *handler_reader_from_app* класса *cl_3*

Функционал: обработчик сигнала из программы.

Параметры: string msg.

Возвращаемое значение: void.

Алгоритм метода представлен в таблице 5.

Таблица 5 – Алгоритм метода *handler_reader_from_app* класса *cl_3*

№	Предикат	Действия	№ перехода
1		объявление переменной s_cmd типа string	2
2		ввод значения s_cmd	3
3	s_cmd == "C" s_cmd == "Off"	вызов метода emit_signal с методом сигнала signal_reader_to_all и параметром s_cmd	Ø
			4
4	s_cmd == "+" s_cmd == "-" s_cmd == "*" s_cmd == "/"	присвоение полю s_operation объекта по	5

№	Предикат	Действия	№ перехода
	s_cmd == "%" s_cmd == "<<" s_cmd == ">>"	указателю get_head значения s_cmd	
		присвоение полю s_expression объекта по указателю get_head значения s_cmd	9
5		присвоение полю s_expression объекта по указателю get_head значения s_expression + " " + s_cmd	6
6		ввод значения поля s_operand2 объекта по указателю get_head	7
7		присвоение полю s_expression объекта по указателю get_head значения s_expression + " " + s_operand_2	8
8		вызов метода emit_signal с методом сигнала signal_reader_to_all и параметром s_cmd	∅
9		присвоение полю i_result объекта по указателю get_head значения s_cmd	∅

3.5 Алгоритм метода signal_reader_to_all класса cl_3

Функционал: сигнал для других обработчиков.

Параметры: string& msg.

Возвращаемое значение: void.

Алгоритм метода представлен в таблице 6.

Таблица 6 – Алгоритм метода signal_reader_to_all класса cl_3

№	Предикат	Действия	№ перехода
1			∅

3.6 Алгоритм метода `handler_screen_from_all` класса `cl_4`

Функционал: обработчик сигналов.

Параметры: `string msg`.

Возвращаемое значение: `void`.

Алгоритм метода представлен в таблице 7.

Таблица 7 – Алгоритм метода `handler_screen_from_all` класса `cl_4`

№	Предикат	Действия	№ перехода
1		объявление переменной потока <code>ss</code> типа <code>ostringstream</code>	2
2		запись в <code>ss</code> результата преобразования поля <code>i_result</code> объекта по указателю <code>get_head</code> в шестнадцатеричную <code>ss</code>	3
3	<code>get_head()->s_operand_2 == "C" get_head()->s_operand_2 == "Off"</code>	присвоение полю <code>s_operand_2</code> объекта по указателю <code>get_head</code> значения ""	Ø
			4
4	<code>get_head()->i_result > 32767 get_head()->i_result < -32768</code>	вывод на экран поля <code>s_expression</code> объекта по указателю <code>get_head</code> и "Overflow"	5
			6
5		присвоение полю <code>s_expression</code> объекта по указателю <code>get_head</code> значения "0"	Ø
6	<code>get_head()->s_operand_2 == "Division by zero"</code>	вывод на экран поля <code>s_expression</code> объекта по указателю <code>get_head</code> и "Division by zero"	7
			8
7		присвоение полю <code>s_expression</code> объекта по указателю <code>get_head</code> значения "0"	Ø
8		вывод поля <code>s_expression</code> объекта по указателю <code>get_head</code> , "HEX, потока <code>ss</code>	9

№	Предикат	Действия	№ перехода
9		вывод "DEC" и поля i_result объекта по указателю get_head	10
10		вывод "BIN"	11
11		вызов метода toBinary с аргументом i_result объекта по указателю get_head	∅

3.7 Алгоритм метода toBinary класса cl_4

Функционал: преобразование в двоичную сс.

Параметры: unsigned int i.

Возвращаемое значение: void.

Алгоритм метода представлен в таблице 8.

Таблица 8 – Алгоритм метода toBinary класса cl_4

№	Предикат	Действия	№ перехода
1		перевод числа i в двоичную сс при помощи bitset	2
2		вывод числа i в двоичной сс	∅

3.8 Алгоритм метода handler_shift_from_reader класса cl_5

Функционал: обработчик сигнала из ввода.

Параметры: string msg.

Возвращаемое значение: void.

Алгоритм метода представлен в таблице 9.

Таблица 9 – Алгоритм метода handler_shift_from_reader класса cl_5

№	Предикат	Действия	№ перехода
1	msg == "<<"	присвоение полю i_result объекта по указателю p_head значения i_result << s_operand_2	3

№	Предикат	Действия	№ перехода
			2
2	msg == "<<"	присвоение полю i_result объекта по указателю p_head значения i_result >> s_operand_2	3
			3
3		вызов метода emit_signal с методом сигнала signal_shift_to_screen и параметром i_result	Ø

3.9 Алгоритм метода signal_shift_to_screen класса cl_5

Функционал: сигнал для вывода.

Параметры: string& msg.

Возвращаемое значение: void.

Алгоритм метода представлен в таблице 10.

Таблица 10 – Алгоритм метода signal_shift_to_screen класса cl_5

№	Предикат	Действия	№ перехода
1			Ø

3.10 Алгоритм метода build_tree_objects класса cl_application

Функционал: построение программы.

Параметры: нет.

Возвращаемое значение: void.

Алгоритм метода представлен в таблице 11.

Таблица 11 – Алгоритм метода build_tree_objects класса cl_application

№	Предикат	Действия	№ перехода
1		объявление указателя p_sub на нулевой объект класса cl_base	2
2		вызов метода set_name с аргументов "System"	3

№	Предикат	Действия	№ перехода
3		присвоение указателю p_sub указателя на новый объект класса cl_1 с параметрами this и "Calc"	4
4		присвоение указателю p_sub указателя на новый объект класса cl_2 с параметрами this и "Cancel"	5
5		присвоение указателю p_sub указателя на новый объект класса cl_3 с параметрами this и "Reader"	6
6		присвоение указателю p_sub указателя на новый объект класса cl_4 с параметрами this и "Screen"	7
7		присвоение указателю p_sub указателя на новый объект класса cl_5 с параметрами this и "Shift"	8
8		вызов метода set_connect с методом signal класса cl_application, значением, возвращаемым методом get_sub_obj с параметром "Reader" и методом handler_reader_from_app класса cl_3	9
9		вызов метода set_connect объекта с именем "Reader" с методом signal_reader_to_all класса cl_3, указателем this и методом handler класса cl_application	10
10		вызов метода set_connect объекта с именем "Reader" с методом signal_reader_to_all класса cl_3, значением, возвращаемым методом get_sub_obj с параметром "Cancel" и методом handler_cancel_from_reader класса cl_2	11
11		вызов метода set_connect объекта с именем "Reader" с методом signal_reader_to_all класса cl_3, значением, возвращаемым методом get_sub_obj с параметром "Shift" и методом handler_shift_from_reader класса cl_5	12
12		вызов метода set_connect объекта с именем "Reader" с методом signal_reader_to_all класса cl_3, значением, возвращаемым методом get_sub_obj с параметром "Calc" и методом handler_calc_from_reader класса cl_1	13
13		вызов метода set_connect объекта с именем "Calc" с методом signal_cacl_to_screen класса cl_1, значением, возвращаемым методом	∅

№	Предикат	Действия	№ перехода
		get_sub_obj с параметром "Screen" и методом handler_screen_from_all класса cl_4	

3.11 Алгоритм метода exes_app класса cl_application

Функционал: работа системы.

Параметры: нет.

Возвращаемое значение: int.

Алгоритм метода представлен в таблице 12.

Таблица 12 – Алгоритм метода exes_app класса cl_application

№	Предикат	Действия	№ перехода
1		объявление переменной s_msg типа string	2
2		вызов метода set_state_branch с параметром 1	3
3	s_cmd!="Off"	вызов метода emit_signal с методом сигнала signal класса cl_application и параметром s_msg	3
		возврат значения 0	Ø

3.12 Алгоритм функции main

Функционал: основная функция программы.

Параметры: нет.

Возвращаемое значение: int, код ошибки.

Алгоритм функции представлен в таблице 13.

Таблица 13 – Алгоритм функции main

№	Предикат	Действия	№ перехода
1		<...>	Ø

4 БЛОК-СХЕМЫ АЛГОРИТМОВ

Представим описание алгоритмов в графическом виде на рисунках 1-10.

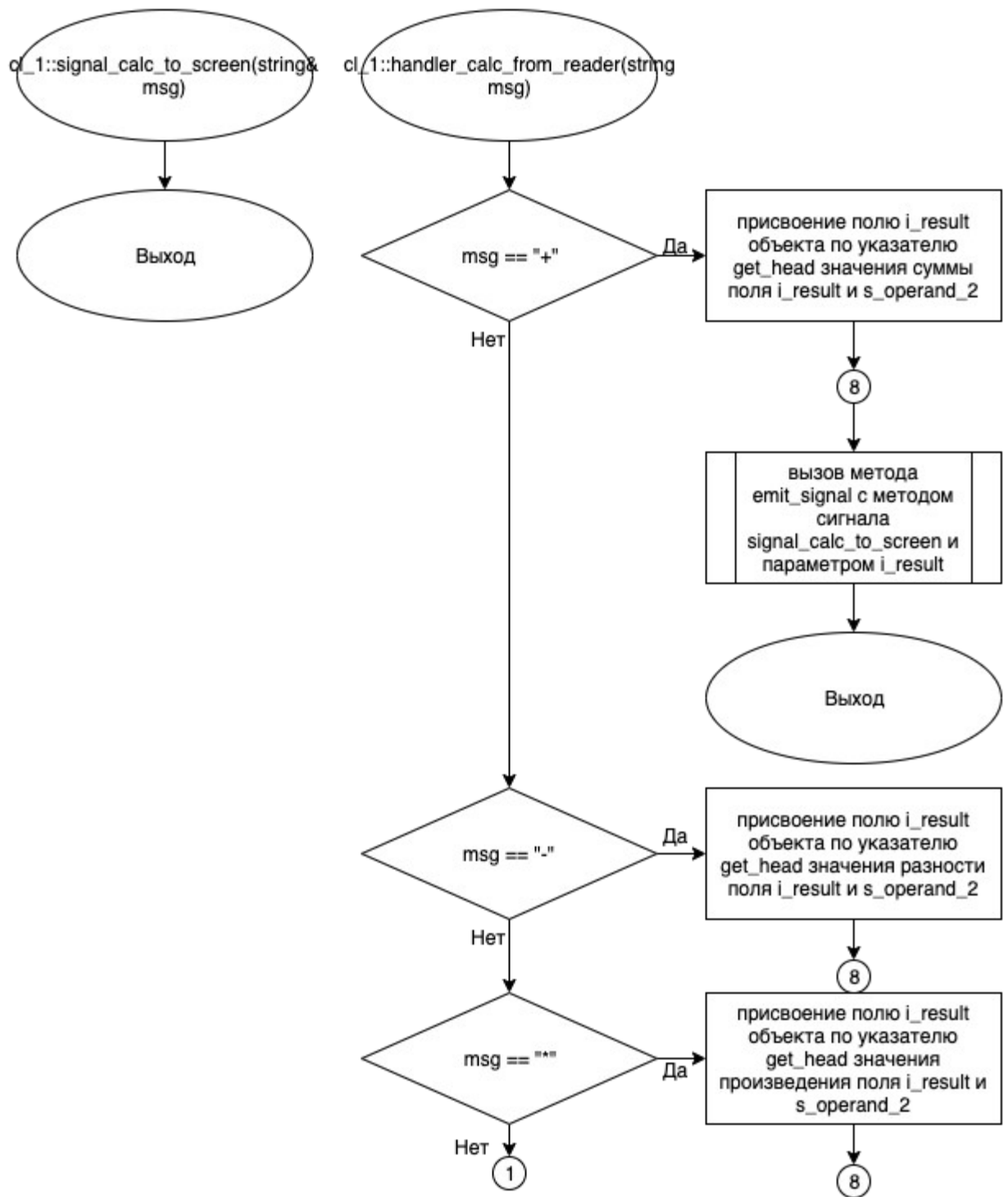


Рисунок 1 – Блок-схема алгоритма

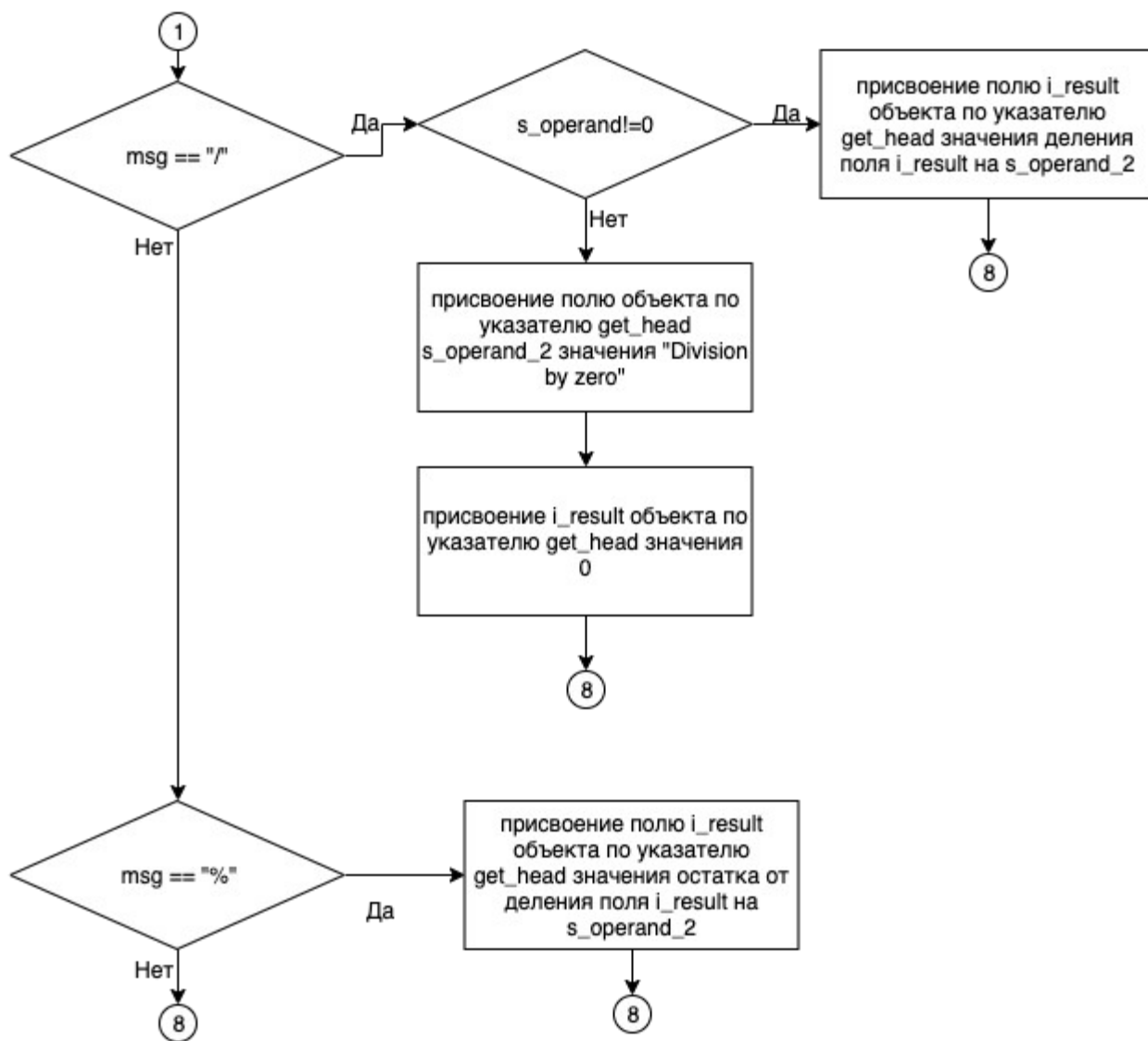


Рисунок 2 – Блок-схема алгоритма

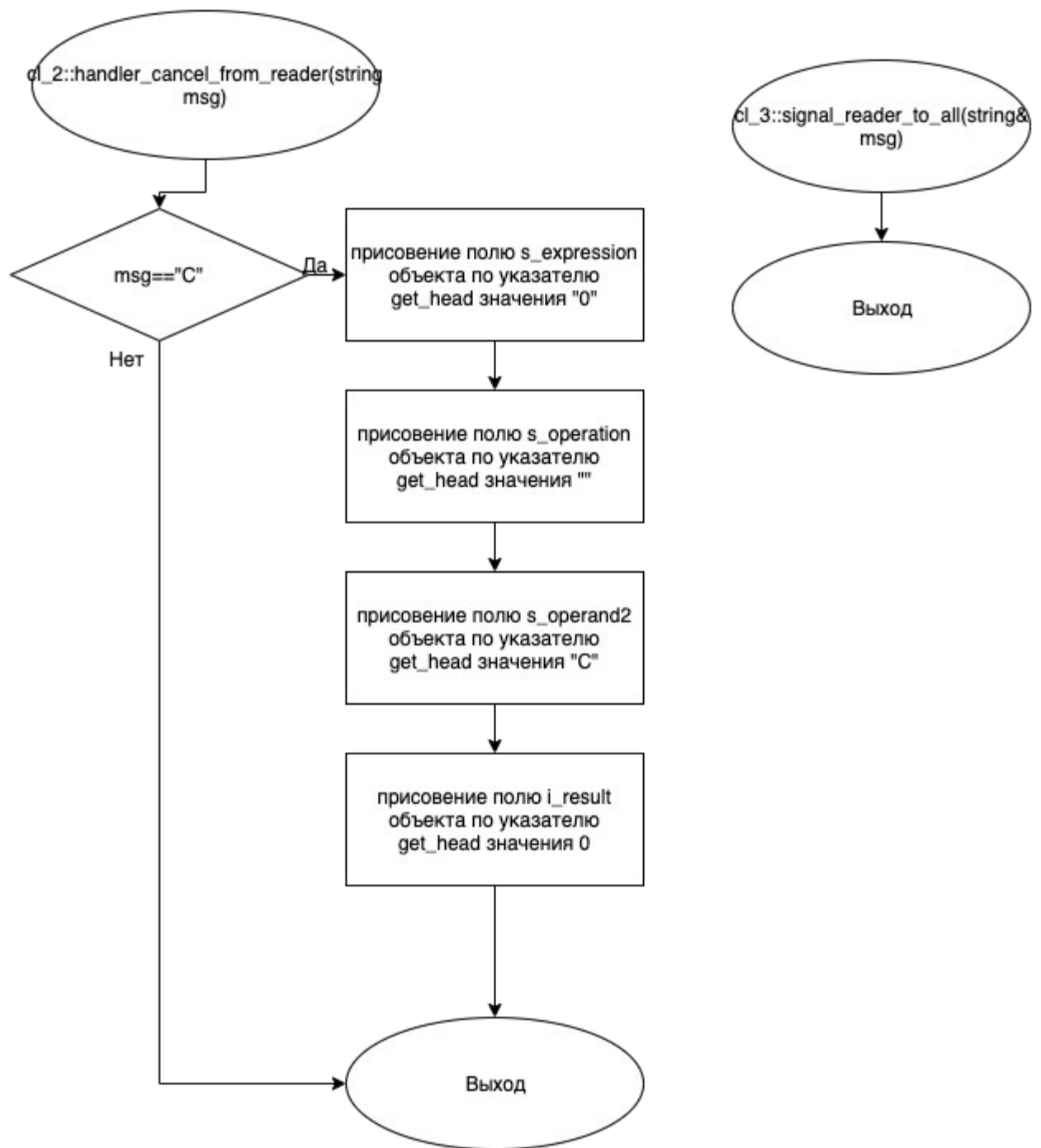


Рисунок 3 – Блок-схема алгоритма

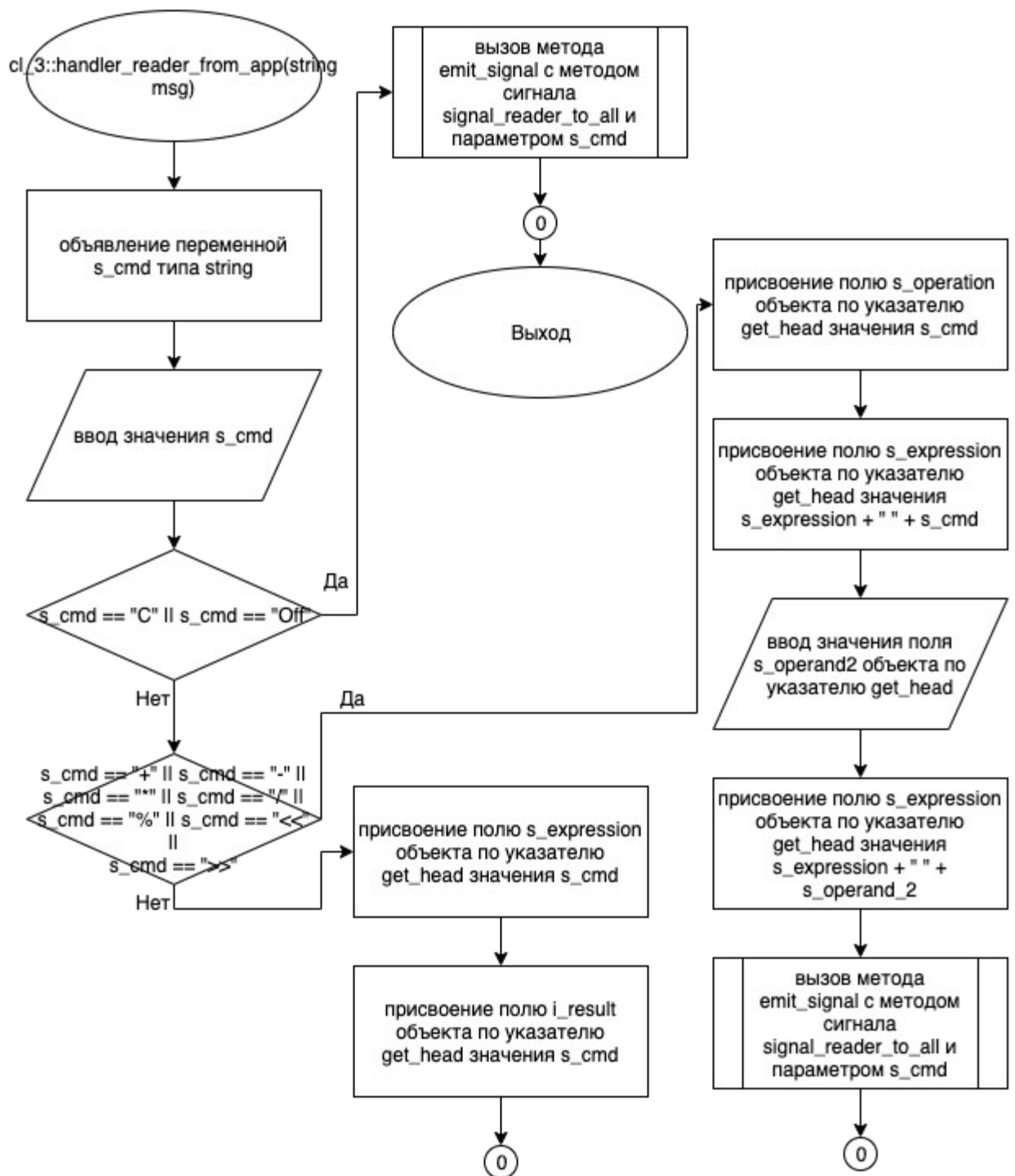


Рисунок 4 – Блок-схема алгоритма

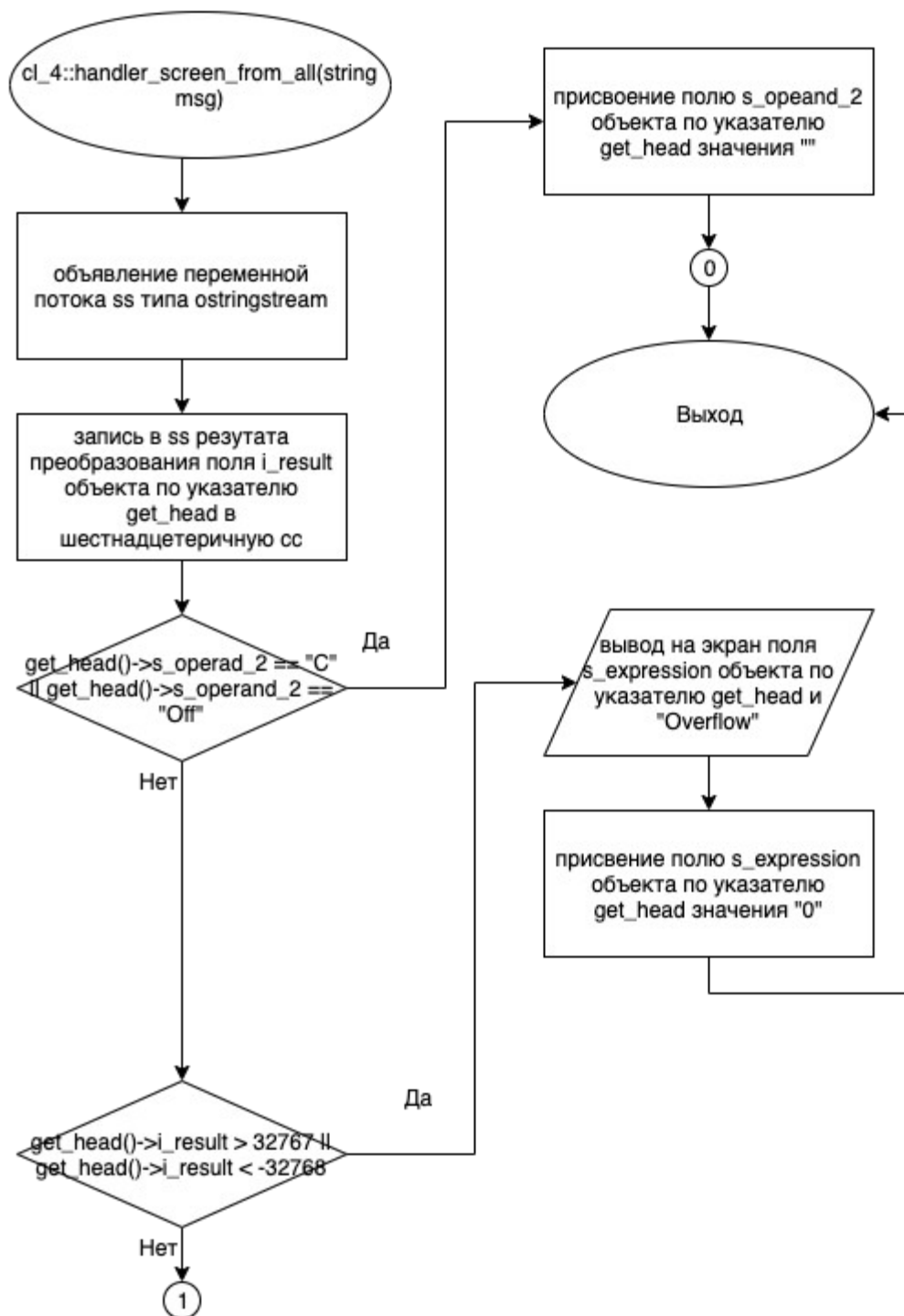


Рисунок 5 – Блок-схема алгоритма

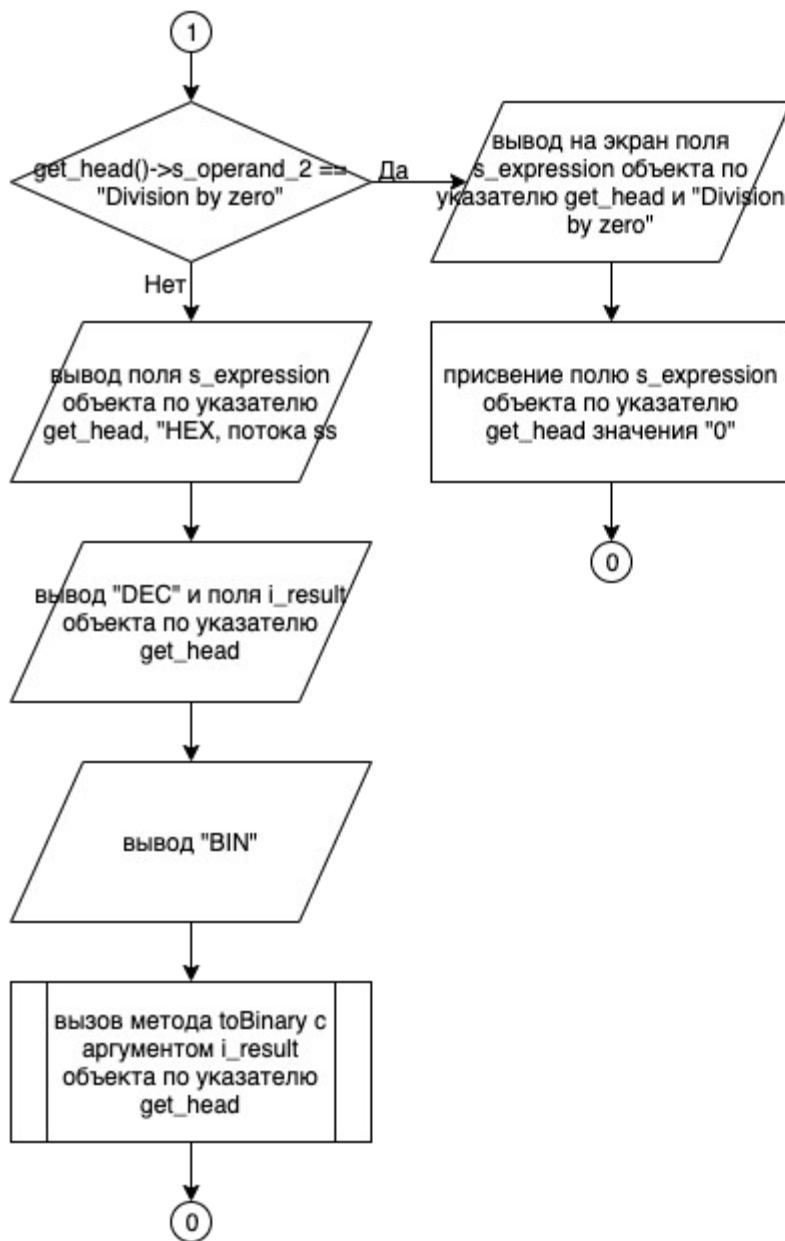


Рисунок 6 – Блок-схема алгоритма

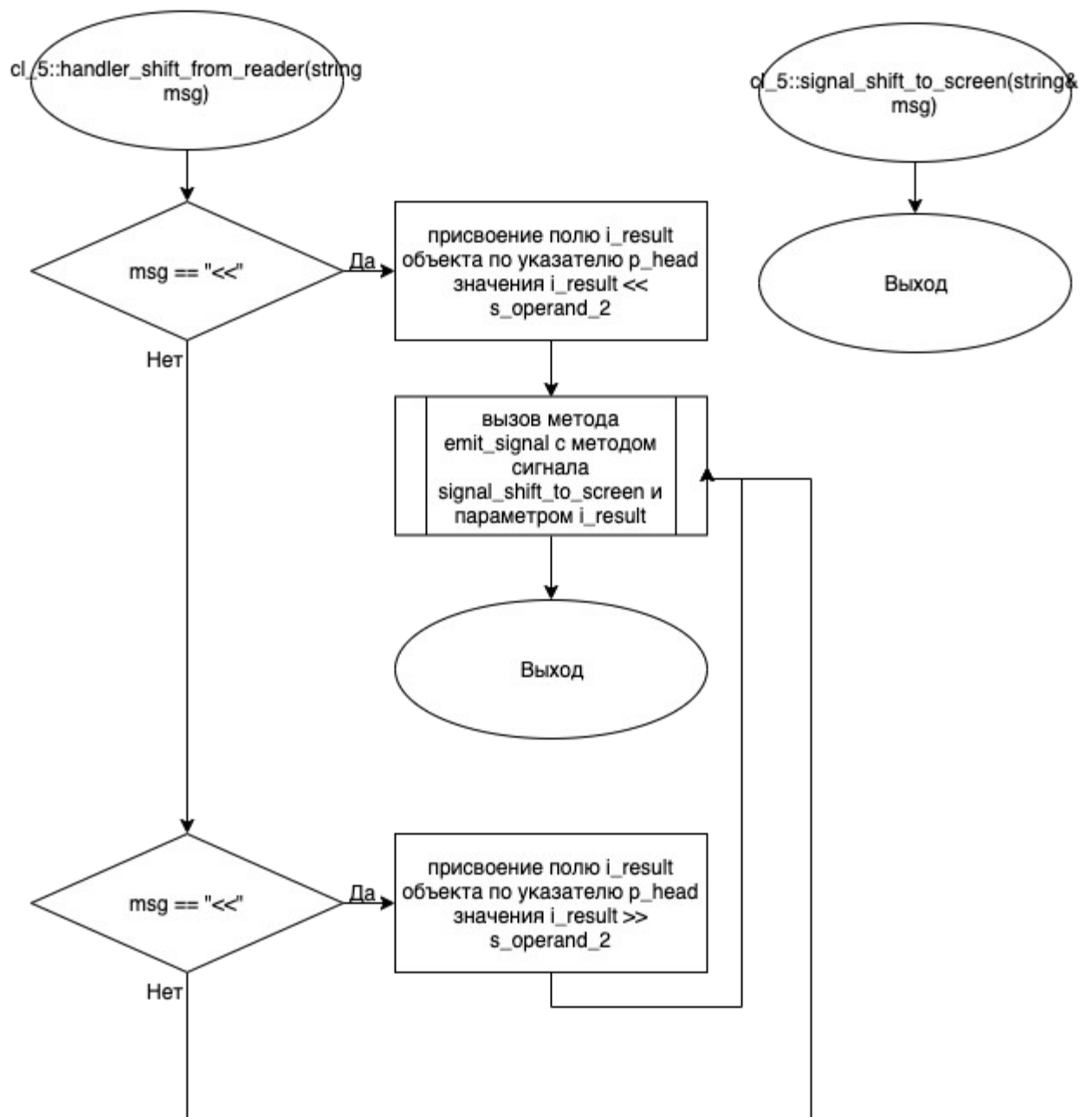


Рисунок 7 – Блок-схема алгоритма

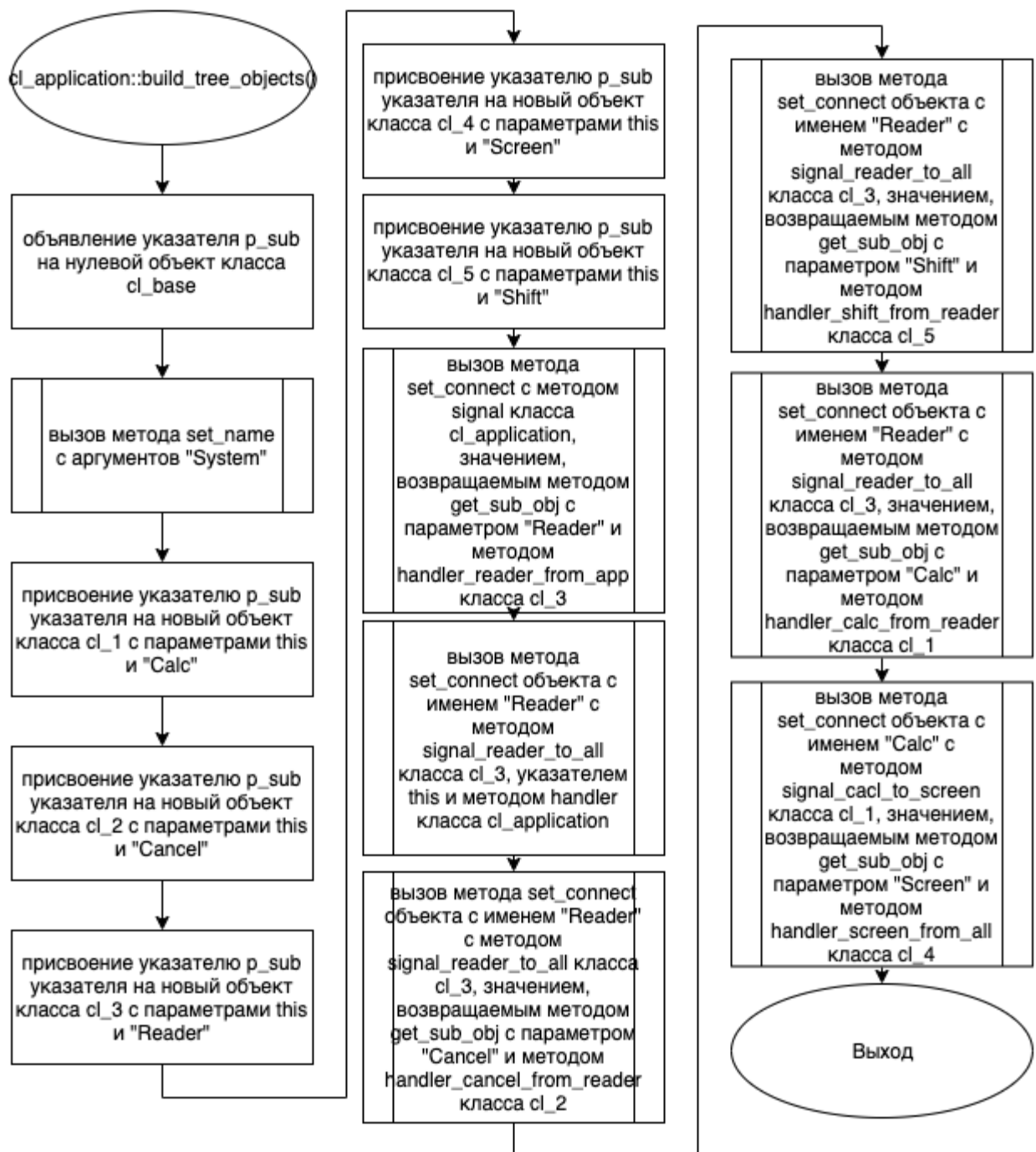


Рисунок 8 – Блок-схема алгоритма

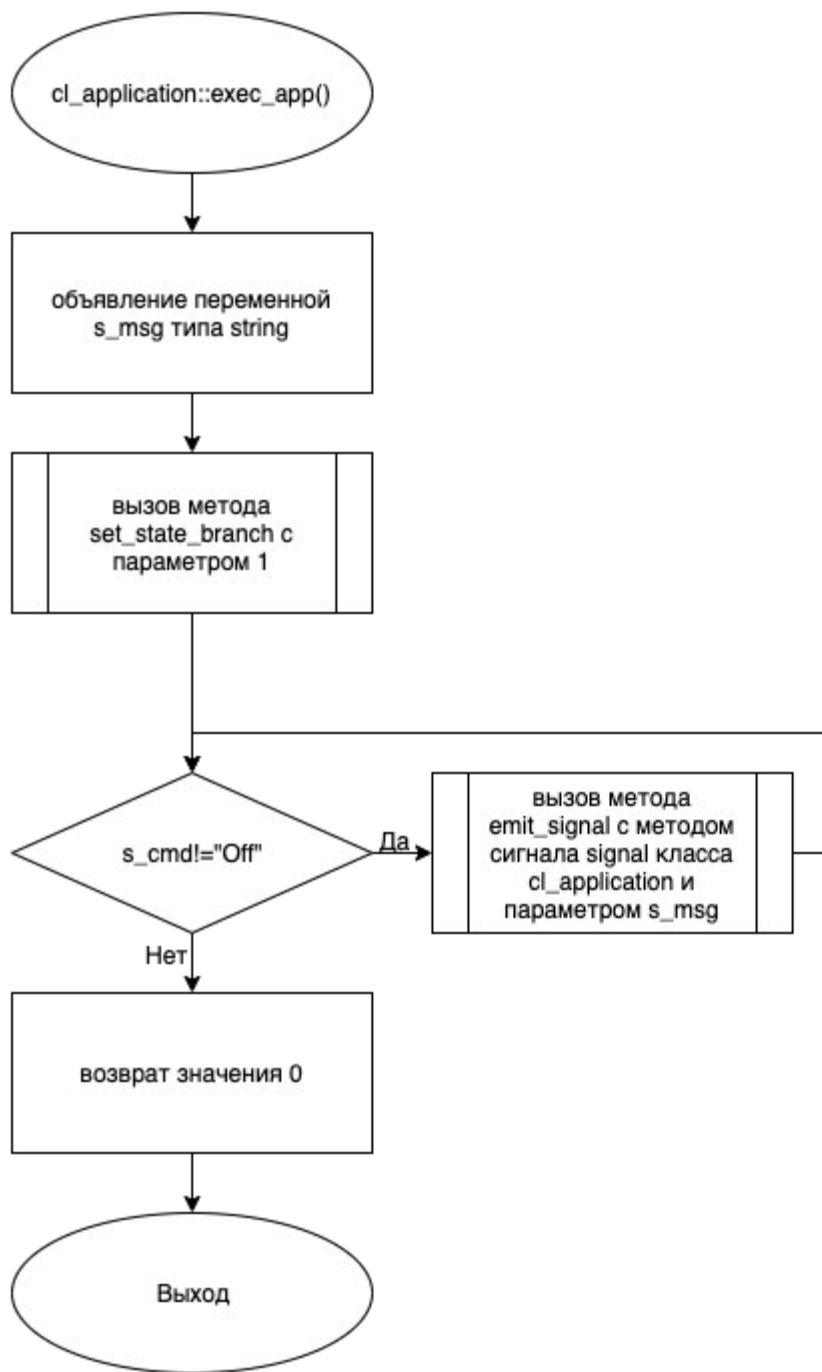


Рисунок 9 – Блок-схема алгоритма

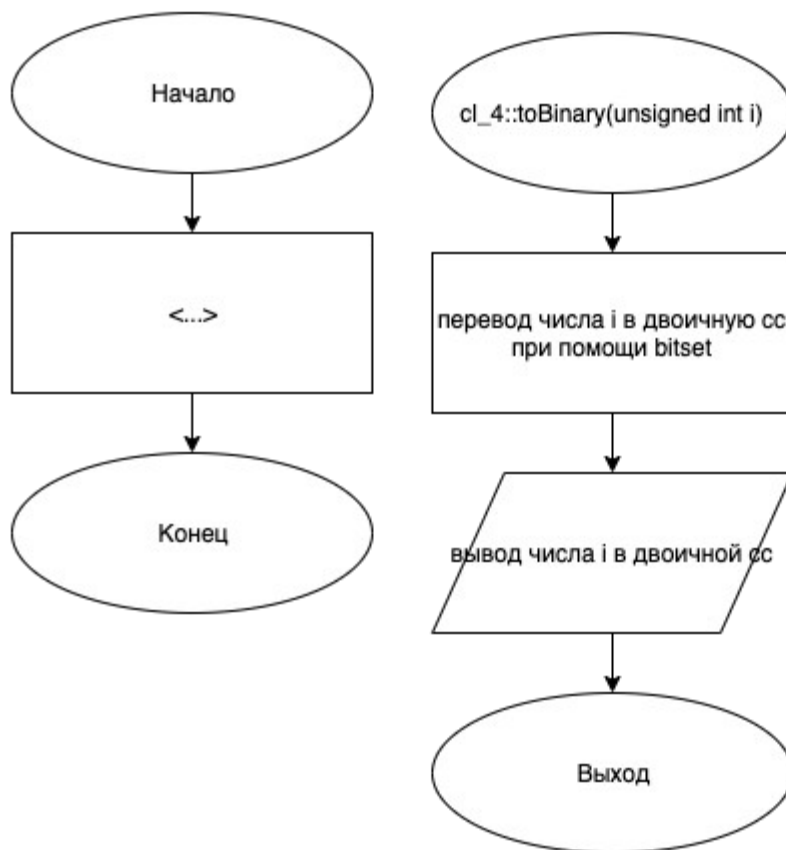


Рисунок 10 – Блок-схема алгоритма

5 КОД ПРОГРАММЫ

Программная реализация алгоритмов для решения задачи представлена ниже.

5.1 Файл cl_1.cpp

Листинг 1 – cl_1.cpp

```
#include "cl_1.h"

cl_1::cl_1(cl_base* p_head_object, string s_name) : cl_base(p_head_object,
s_name){}

int cl_1::get_class_number() {
    return 2; }
void cl_1::signal(string & message) {
    cout << endl << "Signal from " << get_absolute_path();
    message += " (class: " + to_string(get_class_number()) + ")";
}
void cl_1::handler(string message) {
    cout << endl << "Signal to " << get_absolute_path() << " Text: " <<
message;
}

void cl_1::signal_calc_to_screen(string& msg)
{

}

void cl_1::handler_calc_from_reader(string msg)
{
    if (msg == "+")
    {
        get_head()->i_result = get_head()->i_result + atoi((get_head()-
>s_operand_2).c_str());
    }
    else if (msg == "-")
    {
        get_head()->i_result = get_head()->i_result - atoi((get_head()-
>s_operand_2).c_str());
    }
    else if (msg == "*")
    {
        get_head()->i_result = get_head()->i_result * atoi((get_head()-
>s_operand_2).c_str());
    }
}
```



```

        else if (msg == "/")
        {
            if (atoi((get_head()->s_operand_2).c_str()) != 0)
            {
                get_head()->i_result = get_head()->i_result / atoi((get_head()-
>s_operand_2).c_str());
            }
            else
            {
                get_head()->s_operand_2 = "    Division by zero";
                get_head()->i_result = 0;
            }
        }
        else if (msg == "%")
        {
            if (atoi((get_head()->s_operand_2).c_str()) != 0)
            {
                get_head()->i_result = get_head()->i_result % atoi((get_head()-
>s_operand_2).c_str());
            }
            else
            {
                get_head()->s_operand_2 = "    Division by zero";
                get_head()->i_result = 0;
            }
        }
    }

    emit_signal(SIGNAL_D(cl_1::signal_calc_to_screen),std::to_string(get_head()-
>i_result) );
}

```

5.2 Файл cl_1.h

Листинг 2 – cl_1.h

```

#ifndef __CL_1__H
#define __CL_1__H
#include "cl_base.h"

class cl_1 : public cl_base {
public:
    cl_1(cl_base* p_head_object, string s_name);

    int get_class_number();
    void signal(string & message);
    void handler(string message);

    void signal_calc_to_screen(string& msg);

```

```

        void handler_calc_from_reader(string msg);
    };

#endif

```

5.3 Файл cl_2.cpp

Листинг 3 – cl_2.cpp

```

#include "cl_2.h"

cl_2::cl_2(cl_base* p_head_object, string s_name) : cl_base(p_head_object,
s_name){}
int cl_2::get_class_number() {
    return 3; }
void cl_2::signal(string & message) {
    cout << endl << "Signal from " << get_absolute_path();
    message += " (class: " + to_string(get_class_number()) + ")";
}
void cl_2::handler(string message) {
    cout << endl << "Signal to " << get_absolute_path() << " Text: " <<
message;
}

void cl_2::handler_cancel_from_reader(string msg)
{
    if (msg == "C")
    {
        get_head()->s_expression = "0";
        get_head()->s_operation = "";
        get_head()->s_operand_2 = "C";
        get_head()->i_result = 0;
    }
}

```

5.4 Файл cl_2.h

Листинг 4 – cl_2.h

```

#ifndef __CL_2__H
#define __CL_2__H

#include "cl_base.h"

```

```

class cl_2 : public cl_base {
public:
    cl_2(cl_base* p_head_object, string s_name);
    int get_class_number();
    void signal(string & message);
    void handler(string message);

    void handler_cancel_from_reader(string msg);
};

#endif

```

5.5 Файл cl_3.cpp

Листинг 5 – cl_3.cpp

```

#include "cl_3.h"

cl_3::cl_3(cl_base* p_head_object, string s_name) : cl_base(p_head_object,
s_name){}
int cl_3::get_class_number() {
    return 4; }
void cl_3::signal(string & message) {
    cout << endl << "Signal from " << get_absolute_path();
    message += " (class: " + to_string(get_class_number()) + ")";
}
void cl_3::handler(string message) {
    cout << endl << "Signal to " << get_absolute_path() << " Text: " <<
message;
}

void cl_3::handler_reader_from_app(string msg)
{
    string s_cmd;
    cin >> s_cmd;
    if (s_cmd == "C" || s_cmd == "Off")
    {
        emit_signal(SIGNAL_D(cl_3::signal_reader_to_all), s_cmd);
    }
    else if (s_cmd == "+" || s_cmd == "-" || s_cmd == "*" || s_cmd == "/" ||
s_cmd == "%" || s_cmd == "<<" || s_cmd == ">>")
    {
        get_head()->s_operation = s_cmd;
        get_head()->s_expression += (" " + s_cmd);

        cin >> get_head()->s_operand_2;
        get_head()->s_expression += (" " + get_head()->s_operand_2);
        emit_signal(SIGNAL_D(cl_3::signal_reader_to_all), s_cmd);
    }
}

```

```

    }
    else
    {
        get_head()->s_expression = s_cmd;
        get_head()->i_result = atoi((get_head()->s_expression).c_str());
    }
}

void cl_3::signal_reader_to_all(string& msg)
{

}

```

5.6 Файл cl_3.h

Листинг 6 – cl_3.h

```

#ifndef __CL_3__H
#define __CL_3__H

#include "cl_base.h"

class cl_3 : public cl_base {
public:
    cl_3(cl_base* p_head_object, string s_name);
    int get_class_number();
    void signal(string & message);
    void handler(string message);

    void handler_reader_from_app(string msg);
    void signal_reader_to_all(string& msg);
};

#endif

```

5.7 Файл cl_4.cpp

Листинг 7 – cl_4.cpp

```

#include "cl_4.h"
#include "bitset"

cl_4::cl_4(cl_base* p_head_object, string s_name) : cl_base(p_head_object,
s_name){}
int cl_4::get_class_number() {

```

```

        return 5; }
void cl_4::signal(string & message) {
    cout << endl << "Signal from " << get_absolute_path();
    message += " (class: " + to_string(get_class_number()) + ")";
}
void cl_4::handler(string message) {
    cout << endl << "Signal to " << get_absolute_path() << " Text: " <<
message;
}

void cl_4::handler_screen_from_all(string msg)
{
    ostringstream ss;
    ss << setfill('0') << setw(4) << hex << uppercase << (unsigned
short)get_head()->i_result;
    if (get_head()->s_operand_2 == "C" || get_head()->s_operand_2 == "Off")
    {
        get_head()->s_operand_2 = "";
    }
    else if (get_head()->i_result > 32767 || get_head()->i_result < -32768)
    {
        cout << endl << get_head()->s_expression << "      Overflow";
        get_head()->s_expression = "0";
    }
    else if (get_head()->s_operand_2 == "      Division by zero")
    {
        cout << endl << get_head()->s_expression << "      Division by zero";
        get_head()->s_expression = "0";
    }
    else if (get_head()->s_operand_2 != "      Division by zero")
    {
        if (f != 0)
        {
            cout << endl;
        }
        cout << get_head()->s_expression << "      HEX " << ss.str();
        cout << "      DEC " << get_head()->i_result;
        cout << "      BIN";
        toBinary(get_head()->i_result);
        f++;
    }
}

void cl_4::toBinary(unsigned int i)
{
    bitset<16> binary(i);
    string binaryString = binary.to_string();
    for (int i = 12; i >= 0; i -= 4) {
        binaryString.insert(i, " ");
    }
    cout << binaryString;
}

```

5.8 Файл cl_4.h

Листинг 8 – cl_4.h

```
#ifndef __CL_4__H
#define __CL_4__H

#include "cl_base.h"

class cl_4 : public cl_base {
public:
    cl_4(cl_base* p_head_object, string s_name);
    int get_class_number();
    void signal(string & message);
    void handler(string message);

    void handler_screen_from_all(string msg);
    void toBinary(unsigned int i);
};

#endif
```

5.9 Файл cl_5.cpp

Листинг 9 – cl_5.cpp

```
#include "cl_5.h"

cl_5::cl_5(cl_base* p_head_object, string s_name) : cl_base(p_head_object,
s_name){}
int cl_5::get_class_number() {
    return 6; }
void cl_5::signal(string & message) {
    cout << endl << "Signal from " << get_absolute_path();
    message += " (class: " + to_string(get_class_number()) + ")";
}
void cl_5::handler(string message) {
    cout << endl << "Signal to " << get_absolute_path() << " Text: " <<
message;
}

void cl_5::handler_shift_from_reader(string msg)
{
    if (msg == "<<")
    {
        get_head()->i_result = get_head()->i_result << atoi((get_head()-
>s_operand_2).c_str());
    }
}
```

```

    }
    else if (msg == ">>")
    {
        get_head()->i_result = get_head()->i_result >> atoi((get_head()-
>s_operand_2).c_str());
    }

    emit_signal(SIGNAL_D(cl_5::signal_shift_to_screen),to_string(get_head()-
>i_result));
}

void cl_5::signal_shift_to_screen(string& msg)
{

}

```

5.10 Файл cl_5.h

Листинг 10 – cl_5.h

```

#ifndef __CL_5__H
#define __CL_5__H

#include "cl_base.h"

class cl_5 : public cl_base {
public:
    cl_5(cl_base* p_head_object, string s_name);
    int get_class_number();
    void signal(string & message);
    void handler(string message);

    void handler_shift_from_reader(string msg);
    void signal_shift_to_screen(string& msg);
};

#endif

```

5.11 Файл cl_application.cpp

Листинг 11 – cl_application.cpp

```

#include "cl_application.h"

cl_application::cl_application(cl_base* p_head_object) :

```

```

cl_base(p_head_object) {}

void cl_application::build_tree_objects()
{
    cl_base* p_sub = nullptr;

    set_name("System");

    p_sub = new cl_3(this, "Reader");
    p_sub = new cl_1(this, "Calc");
    p_sub = new cl_5(this, "Shift");
    p_sub = new cl_2(this, "Cancel");
    p_sub = new cl_4(this, "Screen");
    this->set_connect(SIGNAL_D(cl_application::signal),
        get_sub_obj("Reader"),
        HANDLER_D(cl_3::handler_reader_from_app));
    get_sub_obj("Reader")->set_connect(SIGNAL_D(cl_3::signal_reader_to_all),
        this,
        HANDLER_D(cl_application::handler));
    get_sub_obj("Reader")->set_connect(SIGNAL_D(cl_3::signal_reader_to_all),
        get_sub_obj("Cancel"),
        HANDLER_D(cl_2::handler_cancel_from_reader));
    get_sub_obj("Reader")->set_connect(SIGNAL_D(cl_3::signal_reader_to_all),
        get_sub_obj("Shift"),
        HANDLER_D(cl_5::handler_shift_from_reader));
    get_sub_obj("Reader")->set_connect(SIGNAL_D(cl_3::signal_reader_to_all),
        get_sub_obj("Calc"),
        HANDLER_D(cl_1::handler_calc_from_reader));
    get_sub_obj("Calc")->set_connect(SIGNAL_D(cl_1::signal_calc_to_screen),
        get_sub_obj("Screen"),
        HANDLER_D(cl_4::handler_screen_from_all));
}

int cl_application::exec_app()
{
    string s_msg;
    this->set_state_branch(1);
    while (s_cmd != "Off")
    {
        emit_signal(SIGNAL_D(cl_application::signal), s_msg);
    }

    return 0;
}

int cl_application::get_class_number() {
    return 1; }
void cl_application::signal(string & message) {

}

void cl_application::handler(string message){
    if (message == "Off")
    {

```



```

        s_cmd = "Off";
        s_operand_2 = "Off";
        exit(0);
    }
}

```

5.12 Файл cl_application.h

Листинг 12 – cl_application.h

```

#ifndef __CL_APPLICATION__H
#define __CL_APPLICATION__H

#include "cl_base.h"
#include "cl_1.h"
#include "cl_2.h"
#include "cl_3.h"
#include "cl_4.h"
#include "cl_5.h"

class cl_application : public cl_base {
public:
    cl_application(cl_base* p_head_object ) ;
    void build_tree_objects();
    int exec_app();
    int get_class_number();
    void signal(string & message);
    void handler(string message);
};

#endif

```

5.13 Файл cl_base.cpp

Листинг 13 – cl_base.cpp

```

#include "cl_base.h"
using namespace std;
cl_base::cl_base(cl_base* head, string name){
    this->s_name = move(name);
    this->p_head_object = head;
    if(head != nullptr){
        head->p_sub_objects.push_back(this);
    }
}

```

```

cl_base::~~cl_base(){
    for(auto &i: p_sub_objects)
        delete i;
}
bool cl_base::set_name(string new_name){
    if (get_head() != nullptr){
        for(auto &i: get_head()->p_sub_objects){
            if(i->get_name() == s_name)
                return false;
        }
    }
    s_name = new_name;
    return true;
}
void cl_base::print_tree(){
    string delay = "";
    cl_base* buf = this;
    while(buf->p_head_object != nullptr){
        delay+="    ";
        buf = buf->p_head_object;
    }
    cout<<endl<<delay<<get_name();
    for (auto p_sub: p_sub_objects)
        p_sub->print_tree();
}
string cl_base::get_name(){
    return s_name;
}
cl_base* cl_base::get_head(){
    return p_head_object;
}
cl_base* cl_base::get_sub_obj(string name){
    for(auto &i: p_sub_objects){
        if(i->s_name == name)
            return i;
    }
    return nullptr;
}
void cl_base::print_ready(){
    string delay = "";
    cl_base* buf = this;
    while(buf->p_head_object != nullptr){
        delay+="    ";
        buf = buf->p_head_object;
    }
    cout<<endl<<delay;
    if(p_ready!=0){
        cout<<get_name()<<" is ready";}
    else{
        cout<<get_name()<<" is not ready";}
    for(auto p_sub: p_sub_objects)
        p_sub->print_ready();
}
cl_base* cl_base::search_by_name(string name){

```

```

        if(name == s_name)
            return this;
        cl_base* p_result = nullptr;
        cl_base* result = nullptr;
        bool found = false;
        for(auto & p_sub_object: p_sub_objects){
            p_result = p_sub_object->search_by_name(name);
            if(p_result == (cl_base*) -1) return (cl_base*) -1;
            if(p_result != nullptr){
                if(found){return (cl_base*) -1;}
                found = true;
                result = p_result;
            }
        }
        return result;
    }

void cl_base::set_ready(int s_new_ready){
    if(s_new_ready!=0){
        if(p_head_object == nullptr || p_head_object->p_ready != 0)
            p_ready = s_new_ready;
    } else {
        p_ready = s_new_ready;
        for(auto & p_sub_object: p_sub_objects)
            p_sub_object->set_ready(s_new_ready);
    }
}

cl_base* cl_base::search_from_root(string name){
    if(p_head_object!=nullptr){
        return p_head_object->search_from_root(name);
    }
    cl_base* answ = search_by_name(name);
    if(answ==(cl_base*) -1)
        return nullptr;
    return search_by_name(name);
}

bool cl_base::head_change(cl_base *new_head){
    if(new_head != nullptr){
        cl_base* tmp = new_head;
        while(tmp != nullptr){
            tmp = tmp->p_head_object;
            if(tmp == this)
                return false;
        }
        if(new_head->get_sub_obj(s_name) == nullptr && p_head_object !=
        nullptr){
            p_head_object->p_sub_objects.erase(find(p_head_object-
            >p_sub_objects.begin(),p_head_object->p_sub_objects.end(), this));
            new_head->p_sub_objects.emplace_back(this);
            p_head_object = new_head;
            return true;
        }
    }
}

```

```

    return false;
}

void cl_base::remove_sub(string name) {
    cl_base* subordinate_obj = get_sub_obj(name);
    if (subordinate_obj != nullptr)
    {
        p_sub_objects.erase(find(p_sub_objects.begin(), p_sub_objects.end(),
subordinate_obj));
        delete subordinate_obj;
    }
}

cl_base* cl_base::cladmen(string coordinaty){
    //посмеялся и хватит, пусть будет нормальное название
    string s = coordinaty;
    cl_base* answ = nullptr;
    if(s==""){
        if(p_head_object==nullptr)
            return this;
        return p_head_object->cladmen(s);
    }
    if(s.substr(0,2)=="//"){
        answ = search_from_root(s.substr(2));
        if(answ == (cl_base*) -1)
            return nullptr;
        return answ;
    }

    if(s == ".")
        return this;
    if(s.substr(0,1)==""){
        answ = search_by_name(s.substr(1));
        if(answ == (cl_base*) -1)
            return nullptr;
        return answ;
    }
    if(s[0]=='/'){
        if(p_head_object==nullptr)
            return cladmen(s.substr(1));
        return p_head_object->cladmen(s);
    }
    if(s.find('/') == s.npos){
        if(this==nullptr || this==(cl_base*) -1)
            return nullptr;
        return get_sub_obj(s);
    }
    return get_sub_obj(s.substr(0,s.find('/'))->cladmen(s.substr(s.find('/')
+1)));
}

void cl_base::set_connect(TYPE_SIGNAL signal_ptr, cl_base* target_ptr,
TYPE_HANDLER handler_ptr) {
    for (auto & connect : connects) {
        if (connect->signal_ptr == signal_ptr && connect->target_ptr
== target_ptr && connect->handler_ptr == handler_ptr) {

```

```

        return;
    }
}
connect *new_connect = new connect();
new_connect->signal_ptr = signal_ptr;
new_connect->target_ptr = target_ptr;
new_connect->handler_ptr = handler_ptr;
connects.push_back(new_connect);
}
void cl_base::remove_connect(TYPE_SIGNAL signal_ptr, cl_base* target_ptr,
TYPE_HANDLER handler_ptr) {
    for (int i = 0; i < connects.size(); ++i) {
        if (connects[i]->signal_ptr == signal_ptr && connects[i]->target_ptr ==
target_ptr && connects[i]->handler_ptr == handler_ptr) {
            delete connects[i];
            connects.erase(connects.begin() + i);
            return; }
    }
}

void cl_base::emit_signal(TYPE_SIGNAL signal_ptr, string command) {
    if (this->p_ready == 0) {
        return; }
    (this->*signal_ptr)(command);
    for (auto & connect : connects) {
        if (connect->signal_ptr == signal_ptr) {
            TYPE_HANDLER handler_ptr = connect->handler_ptr;
            cl_base* target_ptr = connect->target_ptr;
            if (target_ptr->p_ready != 0) {
                (target_ptr->*handler_ptr)(command);
            }
        }
    }
}

string cl_base::get_absolute_path() {
    string result;
    stack<string> st;
    cl_base* root_ptr = this;
    while (root_ptr->get_head() != nullptr) {
        st.push(root_ptr->get_name());
        root_ptr = root_ptr->get_head();
    }
    while (!st.empty()) {
        result += '/' + st.top();
        st.pop();
    }
    if (result.empty()) {
        return "/"; }
    return result;
}

int cl_base::get_class_number() {
    return 0;
}

void cl_base::set_state_branch(int new_state) {
    if (get_head() != nullptr && get_head()->p_ready == 0) {
        return;
    }
}

```

```

    }
    set_ready(new_state);
    for (auto sub: p_sub_objects) {
        sub->set_state_branch(new_state);
    }
}

```

5.14 Файл cl_base.h

Листинг 14 – cl_base.h

```

#ifndef __CL_BASE__H
#define __CL_BASE__H
#include "iostream"
#include "vector"
#include "string"
#include "algorithm"
#include "stack"
#include "string"
#include "iomanip"
#include "sstream"

#define SIGNAL_D(signal_f)(TYPE_SIGNAL)(&signal_f)
#define HANDLER_D(handler_f)(TYPE_HANDLER)(&handler_f)
using namespace std;
class cl_base;
typedef void(cl_base::*TYPE_SIGNAL)(string &);
typedef void(cl_base::*TYPE_HANDLER)(string);

class cl_base{
    struct connect {
        TYPE_SIGNAL signal_ptr;
        cl_base* target_ptr;
        TYPE_HANDLER handler_ptr;
    };
    vector<connect*> connects;
    string s_name;
    cl_base* p_head_object;
    vector<cl_base*> p_sub_objects;
    int p_ready = 0;
public:
    cl_base(cl_base* head, string name = "Base Object");
    bool set_name(string new_name);
    string get_name();
    cl_base* get_head();
    void print_tree();
    cl_base* get_sub_obj(string name);
    ~cl_base();
    cl_base* search_by_name(string name);
    cl_base* search_from_root(string name);

```

```

        void set_ready(int s_new_ready);
        void print_ready();
        bool head_change(cl_base* new_head);
        void remove_sub(string sub_name);
        cl_base* cladmen(string coordinaty);
        void set_connect(TYPE_SIGNAL signal_ptr, cl_base* target_ptr, TYPE_HANDLER
handler_ptr);
        void remove_connect(TYPE_SIGNAL signal_ptr, cl_base* target_ptr,
TYPE_HANDLER handler_ptr);
        void emit_signal(TYPE_SIGNAL signal_ptr, string command);
        string get_absolute_path();
        virtual int get_class_number();
        void set_state_branch(int new_state);

        string s_cmd, s_expression, s_operation, s_operand_2;
        int i_result = 0, f = 0;
    };

#endif

```

5.15 Файл main.cpp

Листинг 15 – main.cpp

```

#include "cl_application.h"
int main ( )
{
    cl_application    ob_cl_application ( nullptr ); // создание корневого
объекта
    ob_cl_application.build_tree_objects ( );          // конструирование
системы, построение дерева объектов
    return ob_cl_application.ехес_app ( );           // запуск системы
}

```

6 ТЕСТИРОВАНИЕ

Результат тестирования программы представлен в таблице 14.

Таблица 14 – Результат тестирования программы

Входные данные	Ожидаемые выходные данные	Фактические выходные данные
5	5 + 5 HEX 000A	5 + 5 HEX 000A
+ 5	DEC 10 BIN 0000	DEC 10 BIN 0000
<< 1	0000 0000 1010	0000 0000 1010
/ 0	5 + 5 << 1 HEX	5 + 5 << 1 HEX
+ 5	0014 DEC 20 BIN	0014 DEC 20 BIN
C	0000 0000 0001 0100	0000 0000 0001 0100
7	5 + 5 << 1 / 0	5 + 5 << 1 / 0
8	Division by zero	Division by zero
/ -3	0 + 5 HEX 0005	0 + 5 HEX 0005
C	DEC 5 BIN 0000 0000	DEC 5 BIN 0000 0000
9	0000 0101	0000 0101
% -4	8 / -3 HEX FFFE	8 / -3 HEX FFFE
+ 7	DEC -2 BIN 1111	DEC -2 BIN 1111
* 11	1111 1111 1110	1111 1111 1110
Off	9 % -4 HEX 0001	9 % -4 HEX 0001
	DEC 1 BIN 0000 0000	DEC 1 BIN 0000 0000
	0000 0001	0000 0001
	9 % -4 + 7 HEX	9 % -4 + 7 HEX
	0008 DEC 8 BIN	0008 DEC 8 BIN
	0000 0000 0000 1000	0000 0000 0000 1000
	9 % -4 + 7 * 11	9 % -4 + 7 * 11
	HEX 0058 DEC 88	HEX 0058 DEC 88
	BIN 0000 0000 0101	BIN 0000 0000 0101
	1000	1000

ЗАКЛЮЧЕНИЕ

В ходе работы я разработал программу, реализующую моделирование работы инженерного арифметического калькулятора на языке C++, используя принципы ООП, сигналы и обработчики. Эта работа помогла мне лучше понять принципы ООП и применение их на практике, в программе со сложной структурой.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ГОСТ 19 Единая система программной документации.
2. Методическое пособие студента для выполнения практических заданий, контрольных и курсовых работ по дисциплине «Объектно-ориентированное программирование» [Электронный ресурс] – URL: https://mirea.aco-avvora.ru/student/files/methodichescoe_posobie_dlya_laboratornyh_rabot_3.pdf (дата обращения 05.05.2021).
3. Приложение к методическому пособию студента по выполнению заданий в рамках курса «Объектно-ориентированное программирование» [Электронный ресурс]. URL: https://mirea.aco-avvora.ru/student/files/Prilozheniye_k_methodichke.pdf (дата обращения 05.05.2021).
4. Шилдт Г. С++: базовый курс. 3-е изд. Пер. с англ.. — М.: Вильямс, 2019. — 624 с.
5. Видео лекции по курсу «Объектно-ориентированное программирование» [Электронный ресурс]. АСО «Аврора».
6. Антик М.И. Дискретная математика [Электронный ресурс]: Учебное пособие /Антик М.И., Казанцева Л.В. — М.: МИРЭА — Российский технологический университет, 2018 — 1 электрон. опт. диск (CD-ROM).