



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

(наименование темы)

Студент группы ИКБО-23-22

(учебная группа)

Зубков Георгий Павлович

(Фамилия Имя Отчество)

(подпись студента)

Руководитель курсовой работы

доцент Путуридзе З.Ш.

(Должность, звание, ученая степень)

(подпись руководителя)

Консультант

ассистент Красников К.Е.

(Должность, звание, ученая степень)

(подпись консультанта)

Работа представлена к защите «20» мая 2023 г.

Допущен к защите «20» мая 2023 г.

Москва 2023 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 2023г.

ЗАДАНИЕ

На выполнение курсовой работы

по дисциплине «Объектно-ориентированное программирование»

Студент Зубков Георгий Павлович Группа ИКБО-23-22

Тема К 3 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 «20» мая 2023 г.

Задание на курсовую работу выдал

Подпись

(Красников К.Е.)

ФИО консультанта

«21» февраля 2023 г.

Задание на курсовую работу получил

Подпись

(Зубков Г.П.)

ФИО исполнителя

«21» февраля 2023 г.

Москва 2023 г.

ОТЗЫВ

на курсовую работу

по дисциплине «Объектно-ориентированное программирование»

Студент Зубков Георгий Павлович группа ИКБО-23-22
(ФИО студента) (Группа)

Характеристика курсовой работы

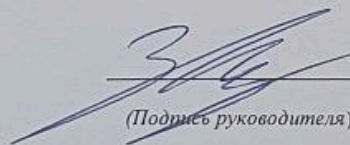
Критерий	Да	Нет	Не полностью
1. Соответствие содержания курсовой работы указанной теме	✓		
2. Соответствие курсовой работы заданию	✓		
3. Соответствие рекомендациям по оформлению текста, таблиц, рисунков и пр.	✓		
4. Полнота выполнения всех пунктов задания			✓
5. Логичность и системность содержания курсовой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

Замечаний:

нет

Рекомендуемая оценка:

отлично


(Подпись руководителя)

доцент Путуридзе З.Ш.

(ФИО руководителя)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	6
1 ПОСТАНОВКА ЗАДАЧИ.....	8
1.1 Описание входных данных.....	10
1.2 Описание выходных данных.....	11
2 МЕТОД РЕШЕНИЯ.....	12
3 ОПИСАНИЕ АЛГОРИТМОВ.....	15
3.1 Алгоритм функции main.....	15
3.2 Алгоритм метода signal_calc_to_screen класса cl_calc.....	15
3.3 Алгоритм метода handler_calc_from_reader класса cl_calc.....	16
3.4 Алгоритм метода handler_cancel_from_reader класса cl_cancel.....	16
3.5 Алгоритм метода handler_reader_from_app класса cl_reader.....	17
3.6 Алгоритм метода signal_reader_to_all класса cl_reader.....	17
3.7 Алгоритм метода handler_screen_from_all класса cl_screen.....	18
3.8 Алгоритм метода toBinary класса cl_screen.....	19
3.9 Алгоритм метода handler_shift_from_reader класса cl_shift.....	19
3.10 Алгоритм метода signal_shift_to_screen класса cl_shift.....	20
3.11 Алгоритм метода build_tree_objects класса cl_application.....	20
3.12 Алгоритм метода exec_app класса cl_application.....	21
3.13 Алгоритм метода signal_app_to_reader класса cl_application.....	21
3.14 Алгоритм метода handler_app_from_reader класса cl_application.....	21
4 БЛОК-СХЕМЫ АЛГОРИТМОВ.....	23
5 КОД ПРОГРАММЫ.....	32
5.1 Файл cl_1.cpp.....	32
5.2 Файл cl_1.h.....	32
5.3 Файл cl_application.cpp.....	33
5.4 Файл cl_application.h.....	37

5.5 Файл cl_base.cpp.....	38
5.6 Файл cl_base.h.....	45
5.7 Файл cl_calc.cpp.....	47
5.8 Файл cl_calc.h.....	48
5.9 Файл cl_cancel.cpp.....	49
5.10 Файл cl_cancel.h.....	50
5.11 Файл cl_reader.cpp.....	50
5.12 Файл cl_reader.h.....	51
5.13 Файл cl_screen.cpp.....	52
5.14 Файл cl_screen.h.....	53
5.15 Файл cl_shift.cpp.....	53
5.16 Файл cl_shift.h.....	54
5.17 Файл main.cpp.....	55
6 ТЕСТИРОВАНИЕ.....	56
ЗАКЛЮЧЕНИЕ.....	57
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....	58

ВВЕДЕНИЕ

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

Объектно-ориентированное программирование (ООП) представляет собой мощный и широко применяемый подход к разработке программного обеспечения. ООП предлагает набор принципов, методов и инструментов, которые помогают создавать модульные, гибкие и легко поддерживаемые приложения. Каждому программисту стоит знать объектно-ориентированное программирование по многим причинам.

Во-первых, из-за его широкого применения. ООП является одним из наиболее распространенных подходов к разработке программного обеспечения. Знание ООП открывает больше возможностей для программиста и делает его более востребованным на рынке труда. Многие компании и проекты используют именно объектно-ориентированное программирование, поэтому владение этим подходом является ценным навыком.

Во-вторых, из-за модульности и возможности повторного использования кода. Объектно-ориентированное программирование позволяет разбить программу на модули, называемые классами, которые объединяют данные и методы, связанные определенной функциональностью. Это позволяет повторно использовать код и также упрощает поддержку и развитие программы.

В-третьих, стоит отметить, что ООП способствует созданию более

читаемого и понятного кода. Классы и объекты, используемые в ООП, позволяют представлять концепции реального мира непосредственно в коде. Это делает код более легким для понимания.

В-четвертых, нельзя забывать и про сами принципы ООП. В объектно-ориентированном программировании существует несколько основных концепций, которые иногда называют "тремя китами ООП". К ним относятся инкапсуляция, наследование и полиморфизм. Инкапсуляция позволяет скрывать внутренние детали, повышая безопасность и понятность кода. Наследование позволяет создавать иерархии классов и избегать переписывания одного и того же кода. Полиморфизм позволяет использовать один и тот же код для работы с различными типами данных. Это сильно упрощает код и повышает его гибкость.

Кроме того, объектно-ориентированное программирование способствует улучшению совместной работы и разделению ответственности. Классы могут быть разработаны и реализованы независимо друг от друга, а затем объединены в единое приложение. Это позволяет команде разработчиков работать параллельно над различными модулями системы.

Несомненно, знание и понимание объектно-ориентированного программирования имеет огромное значение для программиста. ООП является широко применяемым подходом, и его основы являются фундаментальными для разработки современного программного обеспечения, которое отличается модульностью, гибкостью и легкой поддержкой.

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 нуль, 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» `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. 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

Таблица иерархии:

Таблица 1 – Иерархия наследования классов

№	Наименование класса	Классы наследники	Модификатор доступа	Описание	№ класса наследника	Комментарий
1	cl_base			Базовый класс		
		cl_application	public		2	
		cl_calc	public		3	
		cl_cancel	public		4	
		cl_reader	public		5	
		cl_screen	public		6	
		cl_shift	public		7	
2	cl_application			Класс корневого объекта (объекта "Система")		
3	cl_calc			Класс отвечающий за арифметический счёт		
4	cl_cancel			Класс отвечающий за сброс		
5	cl_reader			Класс отвечающий за ввод		
6	cl_screen			Класс отвечающий за вывод		
7	cl_shift			Класс отвечающий за побитовый сдвиг		

- Класс cl_application

- о Поля/свойства:
 - Строковое поле
 - Наименование:
 - Тип данных: строка
 - Модификатор доступа: private
 - о Методы:
 - Метод signal_app_to_reader:
 - Функционал: сигнал
 - Метод handler_app_from_reader:
 - Функционал: обработчик сигнала
- Класс cl_calc
 - о Методы:
 - Метод signal_calc_to_screen:
 - Функционал: сигнал
 - Метод handler_calc_from_reader:
 - Функционал: обработчик сигнала
- Класс cl_cancel
 - о Методы:
 - Метод handler_cancel_from_reader:
 - Функционал: обработчик сигнала
- Класс cl_cancel
 - о Методы:
 - Метод handler_cancel_from_reader:
 - Функционал: обработчик сигнала
- Класс cl_reader
 - о Методы:
 - Метод handler_reader_from_app:

- Функционал: обработчик сигнала
 - Метод `signal_reader_to_all`:
 - Функционал: сигнал
- Класс `cl_screen`
 - Методы:
 - Метод `handler_screen_from_all`:
 - Функционал: обработчик сигнала
 - Метод `toBinary`:
 - Функционал: перевод из 10-ой в 2-ую систему счисления
- Класс `cl_shift`
 - Методы:
 - Метод `handler_shift_from_reader`:
 - Функционал: обработчик сигнала
 - Метод `signal_shift_to_screen`:
 - Функционал: сигнал

3 ОПИСАНИЕ АЛГОРИТМОВ

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

3.1 Алгоритм функции `main`

Функционал: Основная функция.

Параметры: отсутствуют.

Возвращаемое значение: текст.

Алгоритм функции представлен в таблице 2.

Таблица 2 – Алгоритм функции `main`

№	Предикат	Действия	№ перехода
1		Создание объекта <code>obj</code> класса <code>cl_application</code>	2
2		Вызов метода <code>build_tree_objects()</code>	3
3		Запуск программы при помощи метода <code>exes_app</code>	Ø

3.2 Алгоритм метода `signal_calc_to_screen` класса `cl_calc`

Функционал: Сигнал к экрану.

Параметры: `string& msg`.

Возвращаемое значение: Отсутствует.

Алгоритм метода представлен в таблице 3.

Таблица 3 – Алгоритм метода `signal_calc_to_screen` класса `cl_calc`

№	Предикат	Действия	№ перехода
1			Ø

3.3 Алгоритм метода `handler_calc_from_reader` класса `cl_calc`

Функционал: Обработка арифметических операций.

Параметры: `string msg`.

Возвращаемое значение: Отсутствует.

Алгоритм метода представлен в таблице 4.

Таблица 4 – Алгоритм метода `handler_calc_from_reader` класса `cl_calc`

№	Предикат	Действия	№ перехода
1	<code>msg == "+"</code>	Сложение элементов	7
			2
2	<code>msg == "-"</code>	Вычитание элементов	7
			3
3	<code>msg == "*"</code>	Умножение элементов	7
			4
4	<code>msg == "/"</code>		5
			6
5	Деление на 0?	Оповещение <code>screen</code> о делении на 0	7
		Целочисленное деление	7
6	<code>msg == "%"</code>		7
			7
7		Отправка сигнала	Ø

3.4 Алгоритм метода `handler_cancel_from_reader` класса `cl_cancel`

Функционал: Обработчик сигнала сброса.

Параметры: `string msg`.

Возвращаемое значение: Отсутствует.

Алгоритм метода представлен в таблице 5.

Таблица 5 – Алгоритм метода *handler_cancel_from_reader* класса *cl_cancel*

№	Предикат	Действия	№ перехода
1	msg == "C"	Приводит состояние калькулятора к исходному	Ø
			Ø

3.5 Алгоритм метода *handler_reader_from_app* класса *cl_reader*

Функционал: Считывание команд.

Параметры: string msg.

Возвращаемое значение: Отсутствует.

Алгоритм метода представлен в таблице 6.

Таблица 6 – Алгоритм метода *handler_reader_from_app* класса *cl_reader*

№	Предикат	Действия	№ перехода
1		Инициализация переменной хранящей команду	2
2	Комманда == "C" или комманда == "Off"	Отпрака сигнала	Ø
			3
3	Комманда == "+" или комманда == "-" или комманда == "*" или комманда == "/" или комманда == "%" или комманда == "<<" или комманда == ">>"	Отправка сигнала с командой	Ø

3.6 Алгоритм метода *signal_reader_to_all* класса *cl_reader*

Функционал: Сигнал от reader.

Параметры: string& msg.

Возвращаемое значение: Отсутствует.

Алгоритм метода представлен в таблице 7.

Таблица 7 – Алгоритм метода *signal_reader_to_all* класса *cl_reader*

№	Предикат	Действия	№ перехода
1			Ø

3.7 Алгоритм метода *handler_screen_from_all* класса *cl_screen*

Функционал: Вывод результата.

Параметры: string msg.

Возвращаемое значение: Отсутствует.

Алгоритм метода представлен в таблице 8.

Таблица 8 – Алгоритм метода *handler_screen_from_all* класса *cl_screen*

№	Предикат	Действия	№ перехода
1		Создание переменной типа ostringstream хранящей результат в 16-ой системе исчисления	2
2	Комманда == С или комманда == Off		Ø
			3
3	Комманда == "+" или комманда == "-" или комманда == "*" или комманда == "/" или комманда == "%" или комманда == "<<" или комманда == ">>"	Отправка сигнала с командой	Ø
			4
4	Выход за границы 2 битного	Вывод Overflow	Ø

№	Предикат	Действия	№ перехода
	числа		
			5
5	Деление на 0	Вывод Division by zero	Ø
			6
6		Вывод ответа в требуемом формате	Ø

3.8 Алгоритм метода toBinary класса cl_screen

Функционал: Перевод в 2-ую систему исчисления.

Параметры: unsigned int i.

Возвращаемое значение: Отсутствуют.

Алгоритм метода представлен в таблице 9.

Таблица 9 – Алгоритм метода toBinary класса cl_screen

№	Предикат	Действия	№ перехода
1		Перевод в 2-ую систему исчисления при помощи библиотеки bitset	2
2		Разделение по тетрадам с помощью циклов	3
3		Вывод результата	Ø

3.9 Алгоритм метода handler_shift_from_reader класса cl_shift

Функционал: Обработка арифметических операций.

Параметры: string msg.

Возвращаемое значение: Отсутствует.

Алгоритм метода представлен в таблице 10.

Таблица 10 – Алгоритм метода handler_shift_from_reader класса cl_shift

№	Предикат	Действия	№ перехода
1	msg == "<<"	Побитовый сдвиг влево	3

№	Предикат	Действия	№ перехода
			2
2	msg == ">>"	Побитовый сдвиг вправо	3
			3
3		Отправка сигнала	∅

3.10 Алгоритм метода `signal_shift_to_screen` класса `cl_shift`

Функционал: Сигнал к screen.

Параметры: string& msg.

Возвращаемое значение: Отсутствует.

Алгоритм метода представлен в таблице 11.

Таблица 11 – Алгоритм метода `signal_shift_to_screen` класса `cl_shift`

№	Предикат	Действия	№ перехода
1			∅

3.11 Алгоритм метода `build_tree_objects` класса `cl_application`

Функционал: Создание объектов и их связей.

Параметры: Отсутствуют.

Возвращаемое значение: Отсутствует.

Алгоритм метода представлен в таблице 12.

Таблица 12 – Алгоритм метода `build_tree_objects` класса `cl_application`

№	Предикат	Действия	№ перехода
1		Установка имени древа	2
2		Создание объектов, подчиняющиеся этому	3
3		Создание связей между объектами	∅

3.12 Алгоритм метода `exes_app` класса `cl_application`

Функционал: Запуск приложения.

Параметры: Отсутствуют.

Возвращаемое значение: Целый тип.

Алгоритм метода представлен в таблице 13.

Таблица 13 – Алгоритм метода `exes_app` класса `cl_application`

№	Предикат	Действия	№ перехода
1		Устанавливается готовность всех объектов	2
2		Бесконечный цикл с отправкой сигнала	Ø

3.13 Алгоритм метода `signal_app_to_reader` класса `cl_application`

Функционал: Сигнал к reader.

Параметры: `string& msg`.

Возвращаемое значение: Отсутствуют.

Алгоритм метода представлен в таблице 14.

Таблица 14 – Алгоритм метода `signal_app_to_reader` класса `cl_application`

№	Предикат	Действия	№ перехода
1			Ø

3.14 Алгоритм метода `handler_app_from_reader` класса `cl_application`

Функционал: Обработка команды "Off".

Параметры: `string msg`.

Возвращаемое значение: Отсутствуют.

Алгоритм метода представлен в таблице 15.

Таблица 15 – Алгоритм метода *handler_app_from_reader* класса *cl_application*

№	Предикат	Действия	№ перехода
1	Комманда == "Off"	Оканчивает ввод	Ø
			Ø

4 БЛОК-СХЕМЫ АЛГОРИТМОВ

Представим описание алгоритмов в графическом виде на рисунках 1-9.

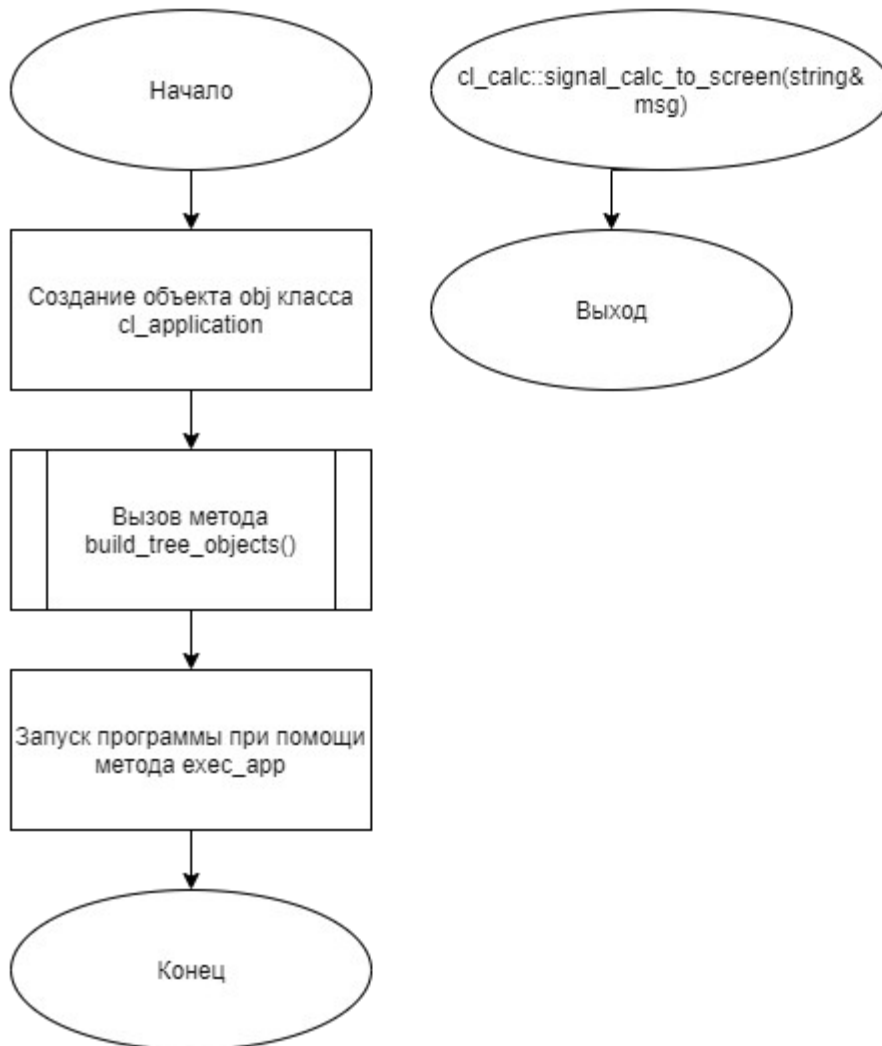


Рисунок 1 – Блок-схема алгоритма

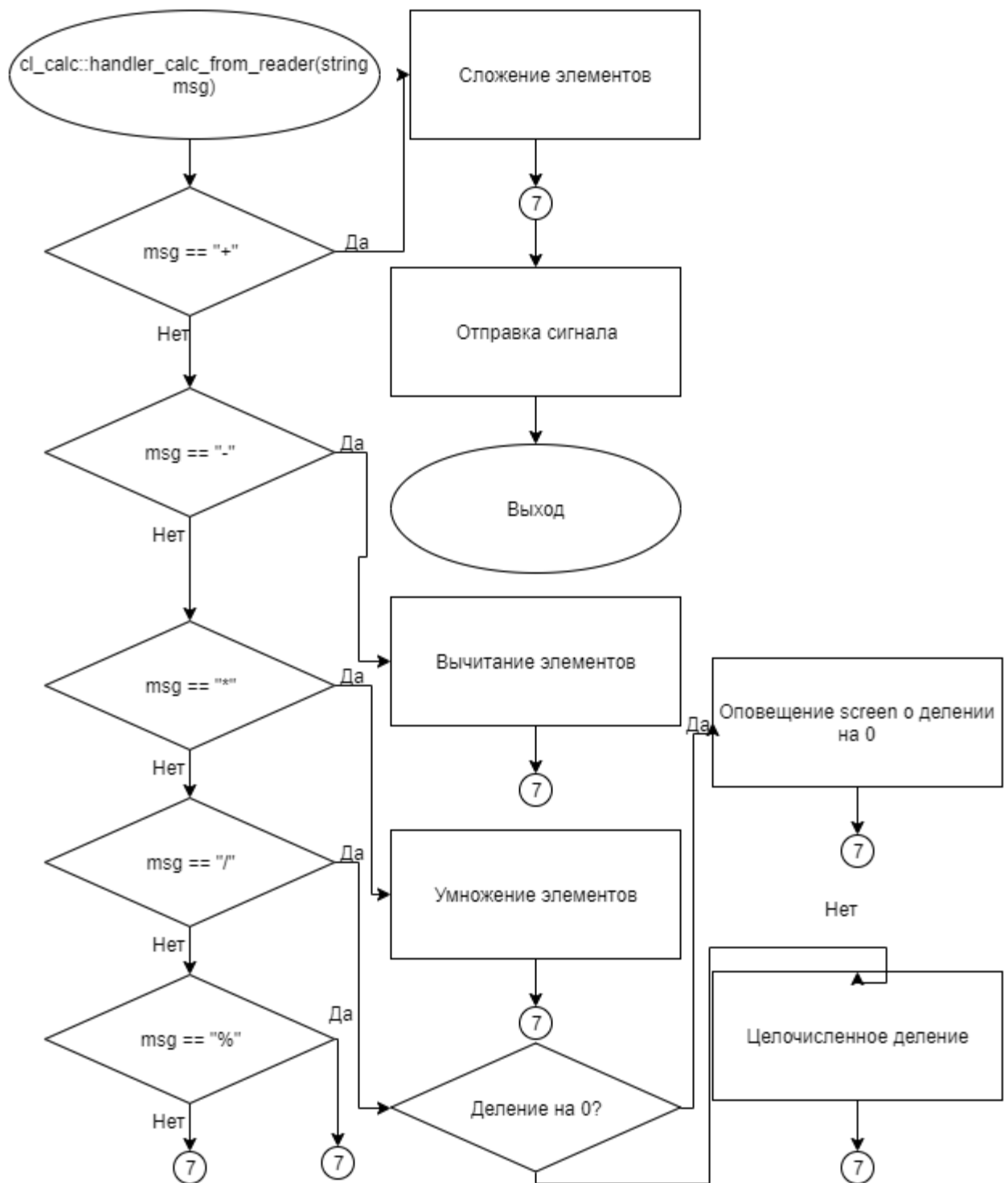


Рисунок 2 – Блок-схема алгоритма

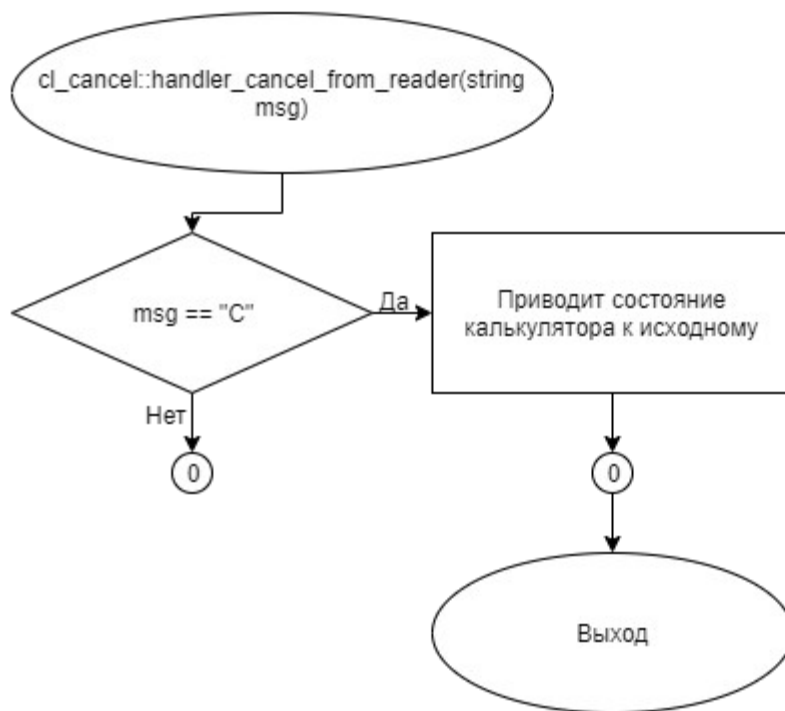


Рисунок 3 – Блок-схема алгоритма

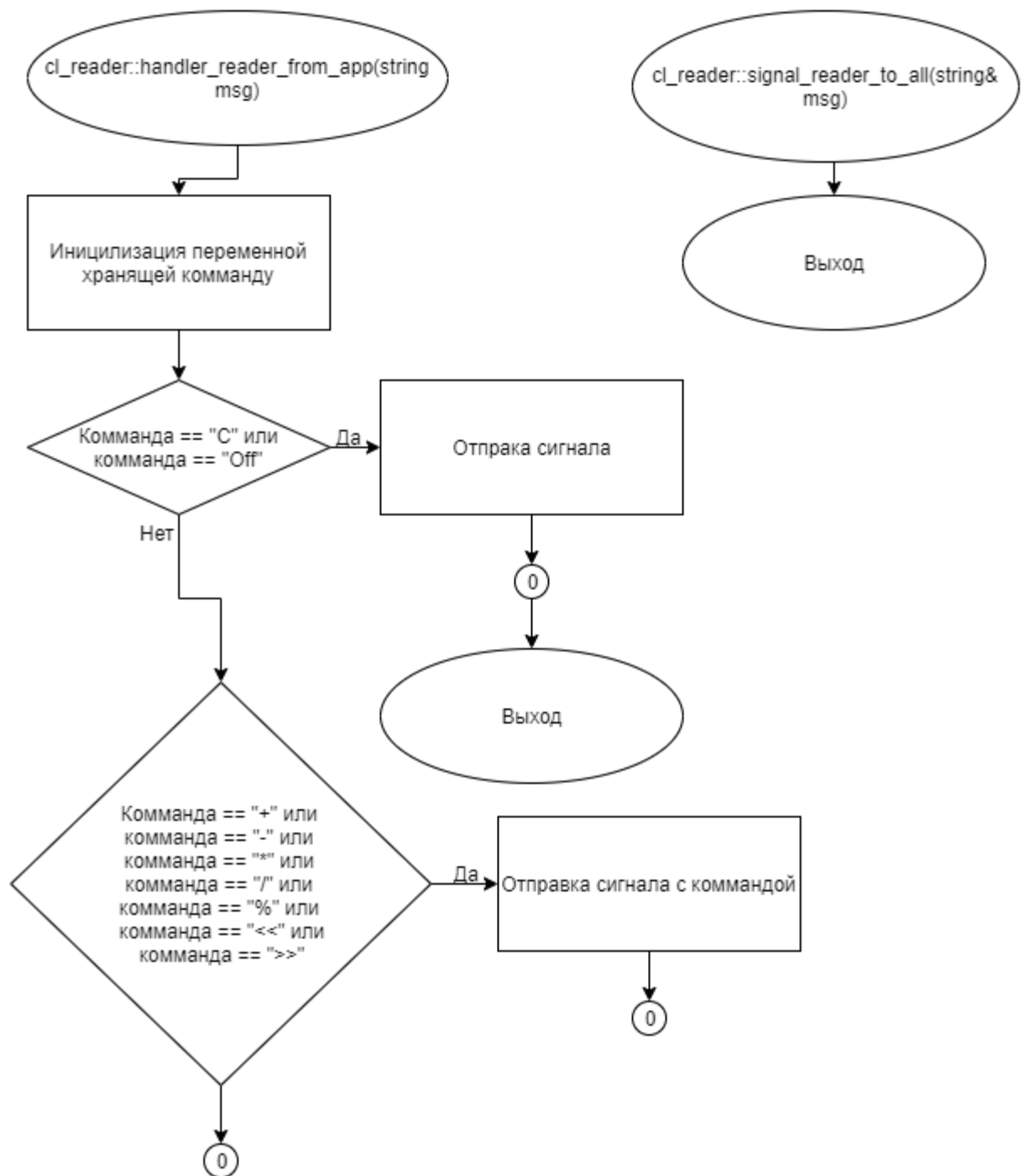


Рисунок 4 – Блок-схема алгоритма

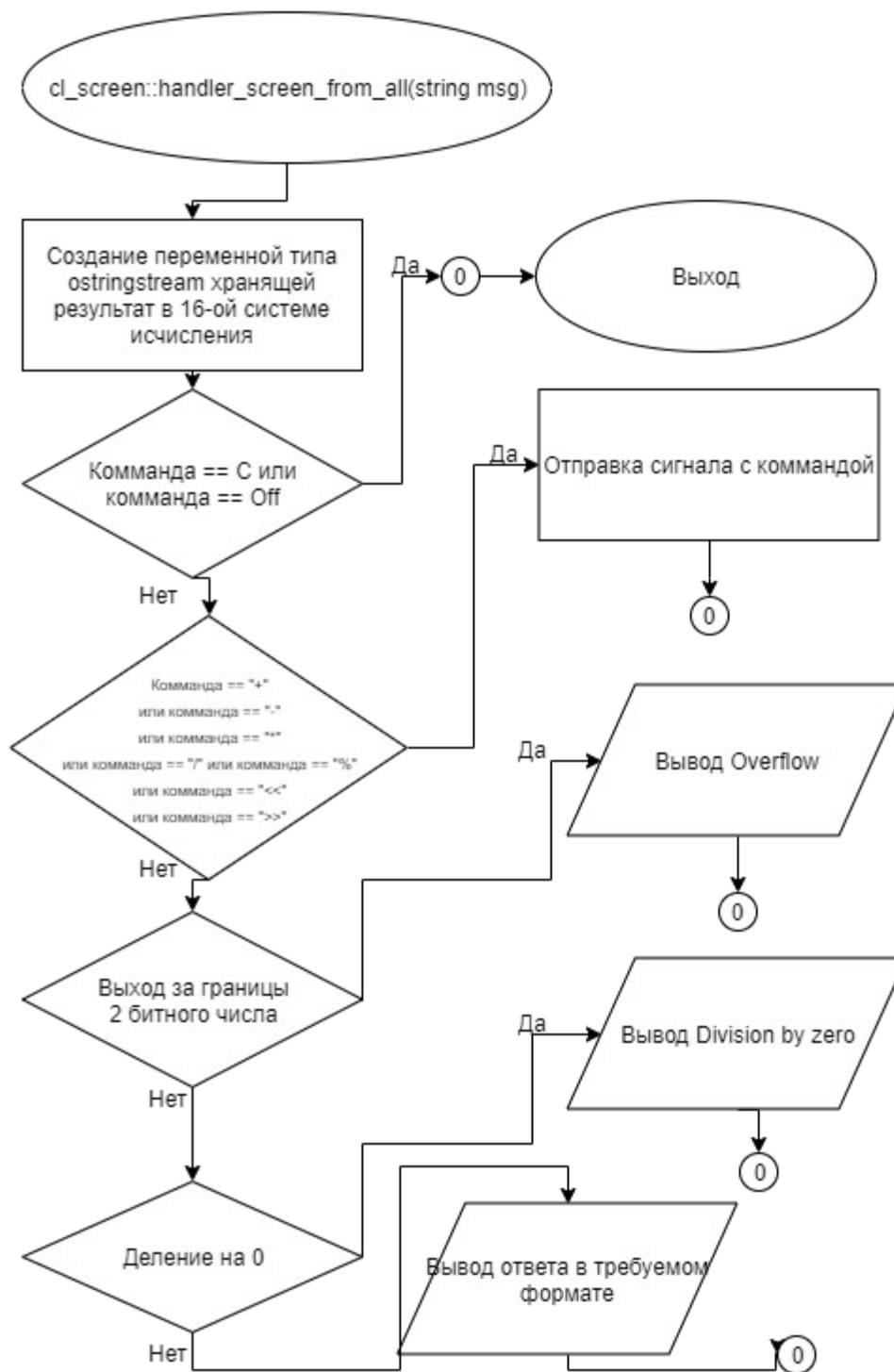


Рисунок 5 – Блок-схема алгоритма

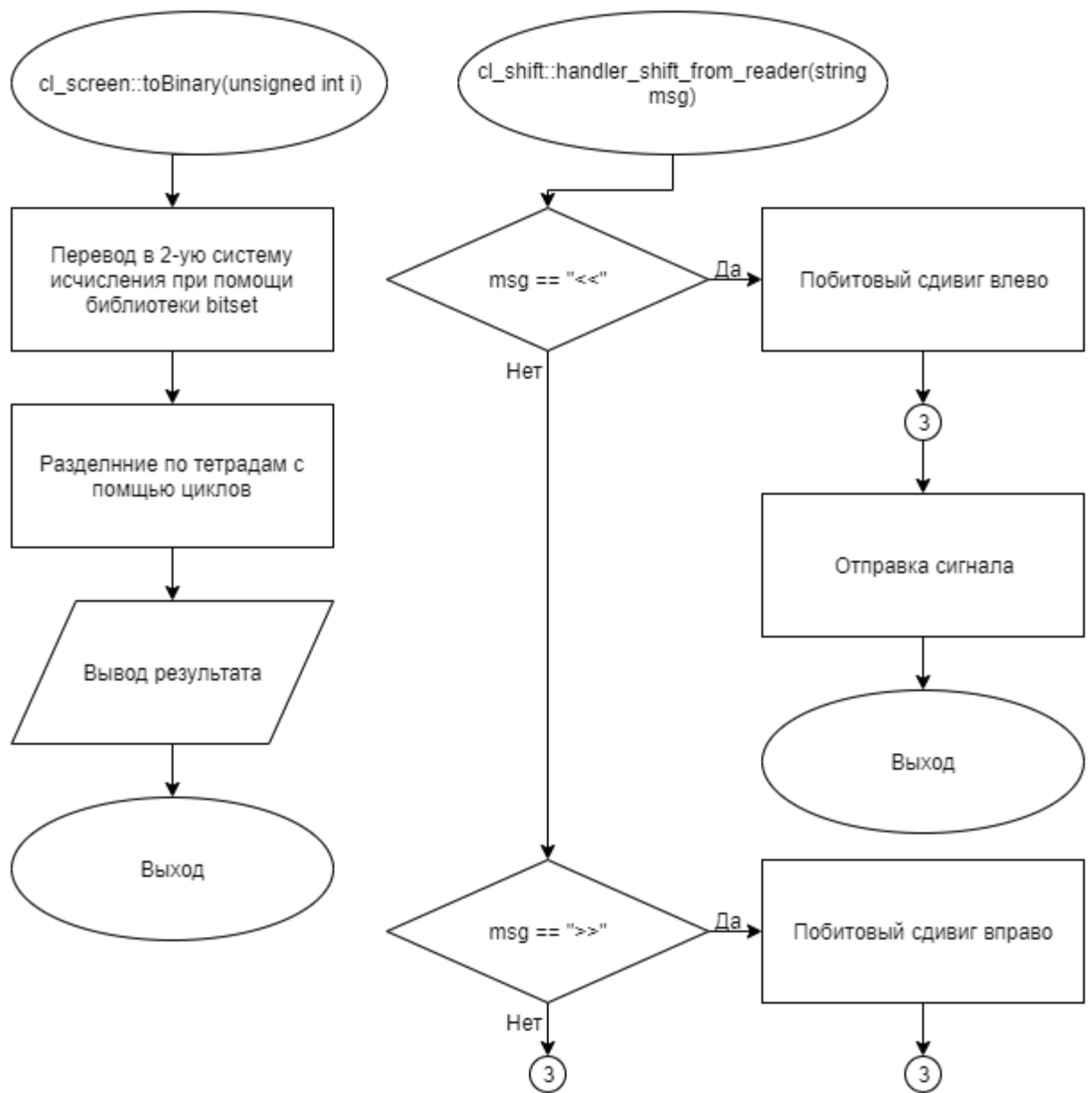


Рисунок 6 – Блок-схема алгоритма

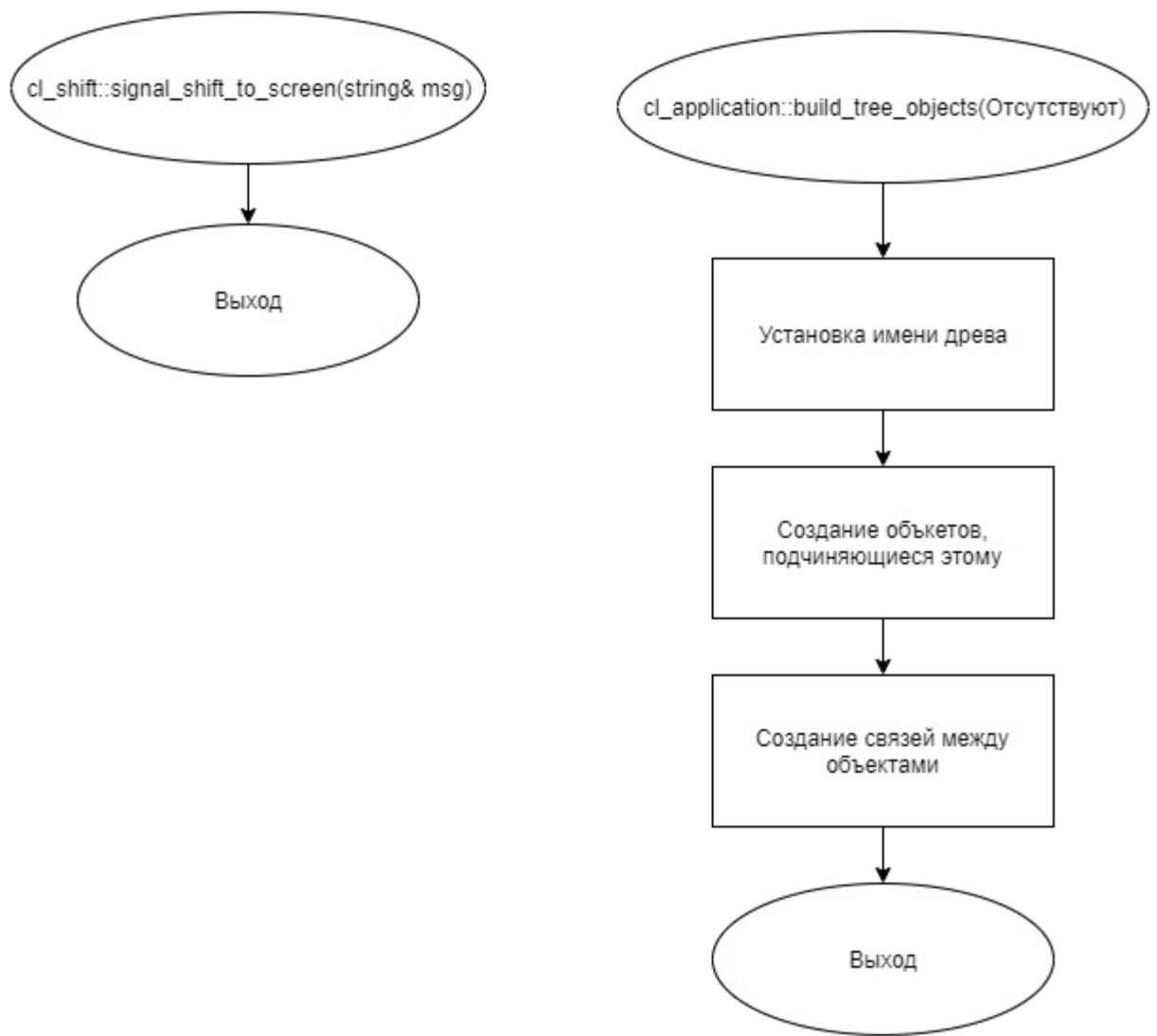


Рисунок 7 – Блок-схема алгоритма

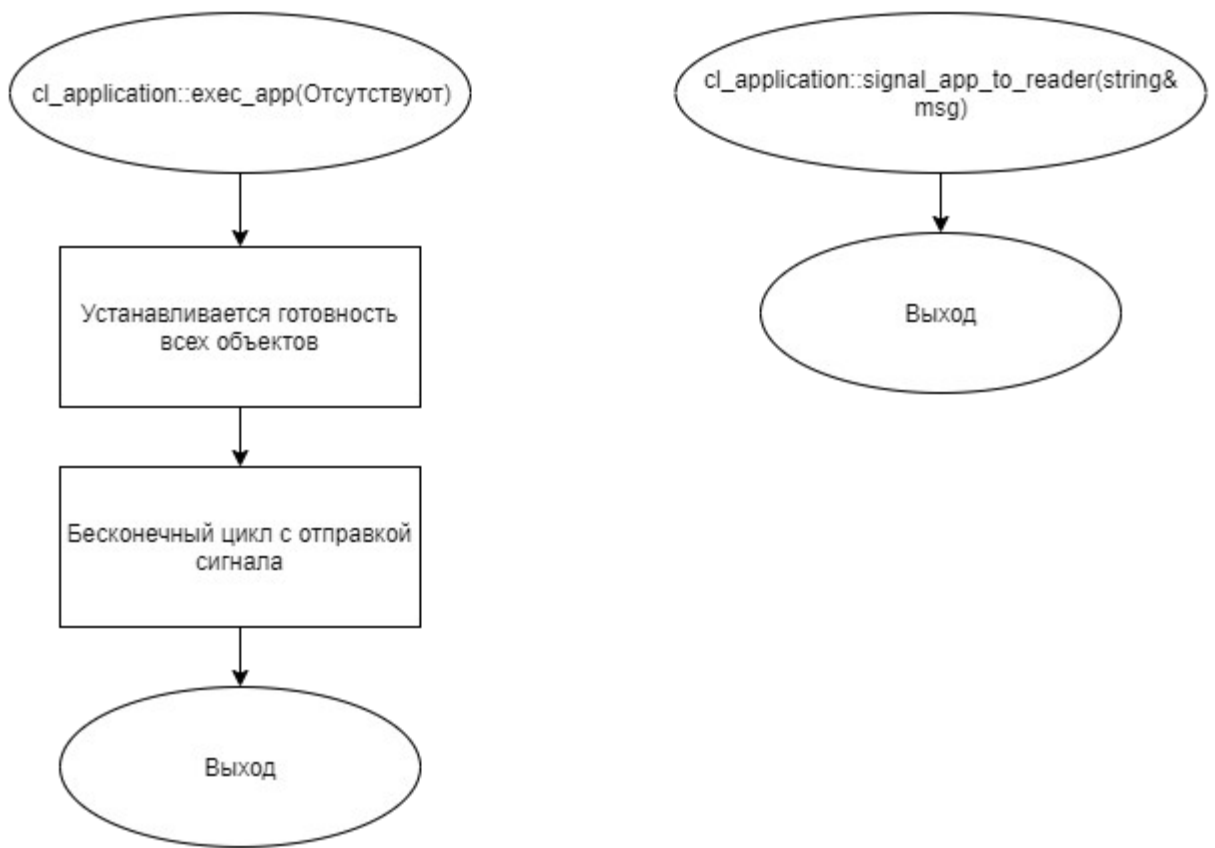


Рисунок 8 – Блок-схема алгоритма

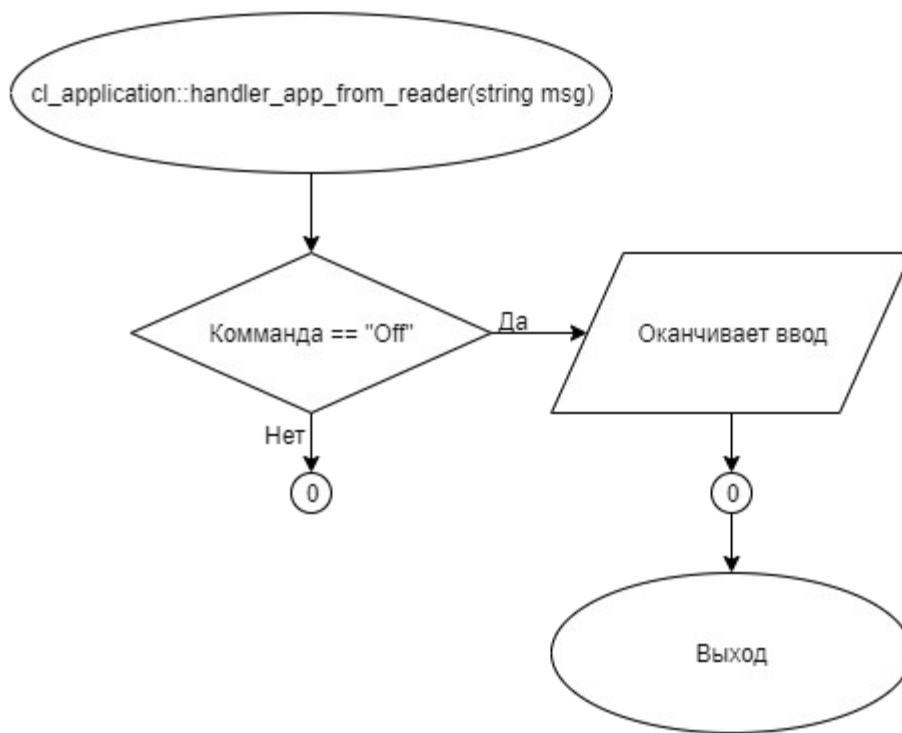


Рисунок 9 – Блок-схема алгоритма

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)
{
    this->number = 1;
}

void cl_1::signal_f(string& msg)
{
    cout << endl << "Signal from " << this->get_path();
    msg += " (class: 1)";
}

void cl_1::handler_f(string msg)
{
    cout << endl << "Signal to " << get_path() << " Text:  " << msg;
}
```

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);
    void signal_f(string& msg);
    void handler_f(string msg);
};
```

```
#endif
```

5.3 Файл cl_application.cpp

Листинг 3 – 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 = this;

    set_name("System");

    p_sub = new cl_reader(this, "Reader");
    p_sub = new cl_calc(this, "Calc");
    p_sub = new cl_shift(this, "Shift");
    p_sub = new cl_cancel(this, "Cancel");
    p_sub = new cl_screen(this, "Screen");

    this->set_connection(SIGNAL_D(cl_application::signal_app_to_reader),
        get_sub_obj("Reader"),
        HANDLER_D(cl_reader::handler_reader_from_app));

    get_sub_obj("Reader")-
    >set_connection(SIGNAL_D(cl_reader::signal_reader_to_all),
        this,
        HANDLER_D(cl_application::handler_app_from_reader));

    get_sub_obj("Reader")-
    >set_connection(SIGNAL_D(cl_reader::signal_reader_to_all),
        get_sub_obj("Cancel"),
        HANDLER_D(cl_cancel::handler_cancel_from_reader));

    get_sub_obj("Reader")-
    >set_connection(SIGNAL_D(cl_reader::signal_reader_to_all),
        get_sub_obj("Shift"),
        HANDLER_D(cl_shift::handler_shift_from_reader));

    get_sub_obj("Reader")-
    >set_connection(SIGNAL_D(cl_reader::signal_reader_to_all),
        get_sub_obj("Calc"),
        HANDLER_D(cl_calc::handler_calc_from_reader));

    get_sub_obj("Calc")-
    >set_connection(SIGNAL_D(cl_calc::signal_calc_to_screen),
        get_sub_obj("Screen"),
```

```

        HANDLER_D(cl_screen::handler_screen_from_all));

//Для KB_1 - KB_4
/*
string s_head, s_sub;
int s_num_obj;
cl_base* p_head = this, * p_sub = nullptr;
cin >> s_head;
set_name(s_head);
while (true)
{
    cin >> s_head;
    if (s_head == "endtree")
    {
        break;
    }
    cin >> s_sub >> s_num_obj;
    if (p_head != nullptr)
    {
        p_head = find_obj_by_coord(s_head);
        switch (s_num_obj)
        {
            case 1:
                p_sub = new cl_1(p_head, s_sub);
                break;
            case 2:
                p_sub = new cl_2(p_head, s_sub);
                break;
            case 3:
                p_sub = new cl_3(p_head, s_sub);
                break;
            case 4:
                p_sub = new cl_4(p_head, s_sub);
                break;
            case 5:
                p_sub = new cl_5(p_head, s_sub);
                break;
            case 6:
                p_sub = new cl_6(p_head, s_sub);
                break;
        }
    }
    else
    {
        cout << "Object tree";
        print_from_current();
        cout << endl << "The head object " << s_head << " is not found";
        exit(1);
    }
}
*/

void cl_application::build_commands()
{

```

```

//Для KB_1 - KB_4
/*
string line, command, coord, text;
vector <TYPE_SIGNAL> SIGNALS_LIST =
{
    SIGNAL_D(cl_1::signal_f),
    SIGNAL_D(cl_2::signal_f),
    SIGNAL_D(cl_3::signal_f),
    SIGNAL_D(cl_4::signal_f),
    SIGNAL_D(cl_5::signal_f),
    SIGNAL_D(cl_6::signal_f)
};
vector<TYPE_HANDLER> HANDLERS_LIST =
{
    HANDLER_D(cl_1::handler_f),
    HANDLER_D(cl_2::handler_f),
    HANDLER_D(cl_3::handler_f),
    HANDLER_D(cl_4::handler_f),
    HANDLER_D(cl_5::handler_f),
    HANDLER_D(cl_6::handler_f)
};
while (true)
{
    getline(cin, line);
    command = line.substr(0, line.find(' '));
    line = line.substr(line.find(' ') + 1, line.size() - 1);
    coord = line.substr(0, line.find(' '));
    text = line.substr(line.find(' ') + 1);
    if (command == "END")
    {
        break;
    }
    if (line == "")
    {
        continue;
    }
    cl_base* pSender = this->find_obj_by_coord(coord);
    if (pSender == nullptr)
    {
        cout << endl << "Object " << coord << " not found";
        continue;
    }
    if (command == "EMIT")
    {
        TYPE_SIGNAL signal = SIGNALS_LIST[pSender->number - 1];
        pSender->emit_signal(signal, text);
    }
    if (command == "SET_CONNECT")
    {
        cl_base* pReceiver = this->find_obj_by_coord(text);
        if (pReceiver == nullptr)
        {
            cout << endl << "Handler object " << text << " not found";
        }
        TYPE_SIGNAL signal = SIGNALS_LIST[pSender->number - 1];
    }
}

```

```

        TYPE_HANDLER handler = HANDLERS_LIST[pReceiver->number - 1];
        pSender->set_connection(signal, pReceiver, handler);
    }
    if (command == "DELETE_CONNECT")
    {
        cl_base* pReceiver = this->find_obj_by_coord(text);
        if (pReceiver == nullptr)
        {
            cout << endl << "Handler object " << text << " not found";
        }
        else
        {
            TYPE_SIGNAL signal = SIGNALS_LIST[pSender->number - 1];
            TYPE_HANDLER handler = HANDLERS_LIST[pReceiver->number - 1];
            pSender->delete_connection(signal, pReceiver, handler);
        }
    }
    if (command == "SET_CONDITION") {
        int state = stoi(text);
        pSender->setState(state);
    }
}
*/
}

int cl_application::exec_app()
{
    string s_msg;
    this->turn_on_subtree();
    while (s_cmd != "Off")
    {
        emit_signal(SIGNAL_D(cl_application::signal_app_to_reader), s_msg);
    }
    //Для KB_1 - KB_4
    /*cout << "Object tree";
    print_from_current();
    this->setConnections();
    build_commands();*/
    return 0;
}

void cl_application::setConnections()
{
    //Для KB_1 - KB_4
    /*
    string senderCoord;
    string receiverCoord;
    cl_base* pSender;
    cl_base* pReceiver;
    vector<TYPE_SIGNAL> SIGNALS_LIST =
    {
        SIGNAL_D(cl_1::signal_f),
        SIGNAL_D(cl_2::signal_f),
        SIGNAL_D(cl_3::signal_f),
        SIGNAL_D(cl_4::signal_f),
    }
    */
}

```



```

        SIGNAL_D(cl_5::signal_f),
        SIGNAL_D(cl_6::signal_f)
    };
    vector<TYPE_HANDLER> HANDLERS_LIST =
    {
        HANDLER_D(cl_1::handler_f),
        HANDLER_D(cl_2::handler_f),
        HANDLER_D(cl_3::handler_f),
        HANDLER_D(cl_4::handler_f),
        HANDLER_D(cl_5::handler_f),
        HANDLER_D(cl_6::handler_f)
    };
    while (true)
    {
        cin >> senderCoord;
        if (senderCoord == "end_of_connections") break;
        cin >> receiverCoord;
        pSender = this->find_obj_by_coord(senderCoord);
        pReceiver = this->find_obj_by_coord(receiverCoord);
        TYPE_SIGNAL signal = SIGNALS_LIST[pSender->number - 1];
        TYPE_HANDLER handler = HANDLERS_LIST[pReceiver->number - 1];
        pSender->set_connection(signal, pReceiver, handler);
    }
    */
}

void cl_application::signal_app_to_reader(string& msg)
{
}

void cl_application::handler_app_from_reader(string msg)
{
    if (msg == "Off")
    {
        s_cmd = "Off";
        s_operand_2 = "Off";
    }
}

```

5.4 Файл cl_application.h

Листинг 4 – cl_application.h

```

#ifndef __CL_APPLICATION_H__
#define __CL_APPLICATION_H__
#include "cl_base.h"
#include "cl_1.h"
#include "cl_calc.h"
#include "cl_cancel.h"

```

```

#include "cl_reader.h"
#include "cl_screen.h"
#include "cl_shift.h"
class cl_application : public cl_base
{
public:
    cl_application(cl_base* p_head_object);
    void build_tree_objects();
    int exec_app();
    void build_commands();
    void setConnections();
    void signal_app_to_reader(string& msg);
    void handler_app_from_reader(string msg);
private:
    string s_cmd = "";
};
#endif

```

5.5 Файл cl_base.cpp

Листинг 5 – cl_base.cpp

```

#include "cl_base.h"

cl_base::cl_base(cl_base* p_head_object, string s_name)
{
    this->s_name = s_name;
    this->p_head_object = p_head_object;
    if (p_head_object != nullptr)
    {
        p_head_object->p_sub_objects.push_back(this);
    }
}

cl_base::~cl_base()
{
    get_root()->delete_links(this);
    for (int i = 0; i < p_sub_objects.size(); i++)
    {
        delete p_sub_objects[i];
    }
    if (p_head_object != nullptr)
    {
        p_head_object->delete_subordinate_obj(this->s_name);
    }
    for (auto i : connects)
    {
        delete i;
    }
    connects.clear();
}

```

```

}

bool cl_base::set_name(string s_new_name)
{
    if (get_head() != nullptr)
    {
        for (int i = 0; i < get_head()->p_sub_objects.size(); i++)
        {
            if (get_head()->p_sub_objects[i]->get_name() == s_new_name)
            {
                return false;
            }
        }
    }
    s_name = s_new_name;
    return true;
}

void cl_base::print_tree(string delay)
{
    cout << endl << delay << get_name();
    for (auto p_sub : p_sub_objects)
    {
        p_sub->print_tree(delay + "    ");
    }
}

void cl_base::print_ready(string delay)
{
    cout << endl << delay;
    get_ready(get_name());
    for (auto p_sub : p_sub_objects)
    {
        p_sub->print_ready(delay + "    ");
    }
}

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 s_name)
{
    for (int i = 0; i < p_sub_objects.size(); i++)
    {
        if (p_sub_objects[i]->s_name == s_name)
        {
            return p_sub_objects[i];
        }
    }
}

```

```

    }
    return nullptr;
}

int cl_base::count(string name)
{
    int count = 0;
    if (get_name() == name)
    {
        count++;
    }
    for (int i = 0; i < p_sub_objects.size(); i++)
    {
        count += p_sub_objects[i]->count(name);
    }
    return count;
}

cl_base* cl_base::search_by_name(string name)
{
    if (s_name == name)
    {
        return this;
    }
    cl_base* p_result = nullptr;
    for (int i = 0; i < p_sub_objects.size(); i++)
    {
        p_result = p_sub_objects[i]->search_by_name(name);
        if (p_result != nullptr)
        {
            return p_result;
        }
    }
    return nullptr;
}

cl_base* cl_base::search_cur(string name)
{
    if (count(name) != 1)
    {
        return nullptr;
    }
    return search_by_name(name);
}

cl_base* cl_base::search_from_root(string name)
{
    if (p_head_object != nullptr)
    {
        return p_head_object->search_from_root(name);
    }
    else
    {
        return search_cur(name);
    }
}

```

```

}

void cl_base::set_ready(int s_new_ready)
{
    if (s_new_ready != 0)
    {
        if (p_head_object == nullptr || p_head_object != nullptr &&
p_head_object->p_ready != 0)
        {
            p_ready = s_new_ready;
        }
    }
    else
    {
        p_ready = s_new_ready;
        for (int i = 0; i < p_sub_objects.size(); i++)
        {
            p_sub_objects[i]->set_ready(s_new_ready);
        }
    }
}

void cl_base::get_ready(string name)
{
    if (get_name() == name)
    {
        if (p_ready != 0)
        {
            cout << get_name() << " is ready";
        }
        else
        {
            cout << get_name() << " is not ready";
        }
    }
    else
    {
        for (int i = 0; i < p_sub_objects.size(); i++)
        {
            return p_sub_objects[i]->get_ready(name);
        }
    }
}

bool cl_base::change_head_obj(cl_base* new_head_obj)
{
    if (new_head_obj != nullptr)
    {
        cl_base* temp = new_head_obj;
        while (temp != nullptr)
        {
            temp = temp->p_head_object;
            if (temp == this)
            {
                return false;
            }
        }
    }
}

```

```

    }
    }
    if (new_head_obj->get_sub_obj(get_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_obj->p_sub_objects.push_back(this);
        p_head_object = new_head_obj;
        return true;
    }
    }
    return false;
}

void cl_base::delete_subordinate_obj(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::find_obj_by_coord(string s_object_path)
{
    if (s_object_path == "")
    {
        return nullptr;
    }
    cl_base* head_obj = this;
    string s_path_item;
    if (s_object_path == "." || s_object_path == "/")
    {
        return head_obj;
    }
    if (s_object_path[0] == '.')
    {
        s_object_path.erase(s_object_path.begin());
        return search_by_name(s_object_path);
    }
    if (s_object_path[1] == '/' && s_object_path[0] == '/')
    {
        s_object_path.erase(s_object_path.begin());
        s_object_path.erase(s_object_path.begin());
        return this->search_from_root(s_object_path);
    }
    if (s_object_path[0] == '/')
    {
        s_object_path.erase(s_object_path.begin());
        while (head_obj->p_head_object != nullptr)
        {
            head_obj = head_obj->p_head_object;

```

```

    }
}
stringstream ss_path(s_object_path);
while (getline(ss_path, s_path_item, '/'))
{
    head_obj = head_obj->get_sub_obj(s_path_item);
    if (head_obj == nullptr)
    {
        return nullptr;
    }
}
return head_obj;
}

void cl_base::print_from_current(int n)
{
    cout << endl;
    for (int i = 0; i < n; i++)
    {
        cout << "    ";
    }
    cout << s_name;
    for (auto p_subordinate_object : p_sub_objects)
    {
        p_subordinate_object->print_from_current(n + 1);
    }
}

void cl_base::set_connection(TYPE_SIGNAL p_signal, cl_base* p_target,
TYPE_HANDLER p_handler)
{
    o_sh* p_value;
    for (int i = 0; i < connects.size(); i++)
    {
        if (connects[i]->p_signal == p_signal &&
            connects[i]->p_handler == p_handler &&
            connects[i]->p_target == p_target)
        {
            return;
        }
    }
    p_value = new o_sh();
    p_value->p_signal = p_signal;
    p_value->p_handler = p_handler;
    p_value->p_target = p_target;

    connects.push_back(p_value);
}

void cl_base::delete_connection(TYPE_SIGNAL p_signal, cl_base* p_target,
TYPE_HANDLER p_handler)
{
    vector<o_sh*>::iterator p_it;
    for (p_it = connects.begin(); p_it != connects.end(); p_it++)
    {

```

```

        if ((*p_it)->p_signal == p_signal &&
            (*p_it)->p_target == p_target &&
            (*p_it)->p_handler == p_handler)
        {
            delete* p_it;
            p_it = connects.erase(p_it);
            p_it--;
        }
    }
}

void cl_base::emit_signal(TYPE_SIGNAL p_signal, string s_masgege)
{
    if (p_ready != 0)
    {
        TYPE_HANDLER pHandler;
        cl_base* pObject;
        (this->*p_signal)(s_masgege);
        for (int i = 0; i < connects.size(); i++)
        {
            if (connects[i]->p_signal == p_signal)
            {
                pHandler = connects[i]->p_handler;
                pObject = connects[i]->p_target;
                if (pObject->p_ready != 0)
                {
                    (pObject->*pHandler)(s_masgege);
                }
            }
        }
    }
}

string cl_base::get_path()
{
    cl_base* p_head_object = this->get_head();
    if (p_head_object != nullptr)
    {
        if (p_head_object->get_head() == nullptr)
        {
            return p_head_object->get_path() + s_name;
        }
        else
        {
            return p_head_object->get_path() + "/" + s_name;
        }
    }
    return "/";
}

void cl_base::setState(int state)
{
    if (state == 0)
    {
        this->p_ready = 0;
        for (int i = 0; i < p_sub_objects.size(); i++) {

```



```

        p_sub_objects[i]->setState(0);
    }
    return;
}
if (this->p_head_object == nullptr || this->p_head_object->p_ready != 0) {
    this->p_ready = state;
}
}

void cl_base::turn_on_subtree()
{
    p_ready = 1;
    for (auto p_sub_obj : p_sub_objects)
    {
        p_sub_obj->turn_on_subtree();
    }
}

void cl_base::delete_links(cl_base* targ)
{
    for (auto p_it = connects.begin(); p_it != connects.end(); p_it++)
    {
        if ((*p_it)->p_target == targ)
        {
            delete (*p_it);
            connects.erase(p_it);
            p_it--;
        }
    }

    for (auto p_sub : p_sub_objects)
    {
        p_sub->delete_links(targ);
    }
}

cl_base* cl_base::get_root()
{
    if (p_head_object != nullptr)
    {
        p_head_object->get_root();
    }
    return this;
}

```

5.6 Файл cl_base.h

Листинг 6 – cl_base.h

```
#ifndef __CL_BASE_H__
```

```

#define __CL_BASE_H__
#include <iostream>
#include <string>
#include <vector>
#include <sstream>
#include <algorithm>
#include <iomanip>
#include <bitset>
#define SIGNAL_D(signal_f) (TYPE_SIGNAL)(amp;signal_f)
#define HANDLER_D(handler_f) (TYPE_HANDLER)(amp;handler_f)
using namespace std;
class cl_base;

typedef void (cl_base::* TYPE_SIGNAL) (string& msg);
typedef void (cl_base::* TYPE_HANDLER) (string msg);

struct o_sh
{
    TYPE_SIGNAL p_signal;
    TYPE_HANDLER p_handler;
    cl_base* p_target;
};

class cl_base
{
private:
    string s_name;
    cl_base* p_head_object;
    vector <cl_base*> p_sub_objects;
    int p_ready = 0;
    vector<o_sh*> connects;
public:
    cl_base(cl_base* p_head_object, string s_name = "Base Object");
    bool set_name(string s_new_name);
    string get_name();
    cl_base* get_head();
    void print_tree(string delay = "");
    cl_base* get_sub_obj(string s_name);
    ~cl_base();
    int count(string name);
    cl_base* search_by_name(string name);
    cl_base* search_cur(string name);
    cl_base* search_from_root(string name);
    void set_ready(int s_new_ready);
    void get_ready(string name);
    void print_ready(string delay = "");
    bool change_head_obj(cl_base* new_head_obj);
    void delete_subordinate_obj(string name);
    cl_base* find_obj_by_coord(string s_object_path);
    void print_from_current(int n = 0);
    void set_connection(TYPE_SIGNAL p_signal, cl_base* p_target, TYPE_HANDLER
p_handler);
    void delete_connection(TYPE_SIGNAL p_signal, cl_base* p_target,
TYPE_HANDLER p_handler);
    void emit_signal(TYPE_SIGNAL p_signal, string massege);

```

```

    string get_path();
    int number = 1;
    typedef void (cl_base::* TYPE_SIGNAL)(string&);
    typedef void (cl_base::* TYPE_HANDLER)(string);
    void setState(int state);
    void turn_on_subtree();
    void delete_links(cl_base* targ);
    cl_base* get_root();
    string s_expression = "", s_operation = "", s_operand_2 = "";
    int i_result = 0, f = 0;
};
#endif

```

5.7 Файл cl_calc.cpp

Листинг 7 – cl_calc.cpp

```

#include "cl_calc.h"

cl_calc::cl_calc(cl_base* p_head_object, string
s_name) :cl_base(p_head_object, s_name)
{
    this->number = 2;
}

void cl_calc::signal_f(string& msg)
{
    cout << endl << "Signal from " << this->get_path();
    msg += " (class: 2)";
}

void cl_calc::handler_f(string msg)
{
    cout << endl << "Signal to " << get_path() << " Text:  " << msg;
}

void cl_calc::signal_calc_to_screen(string& msg)
{
}

void cl_calc::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_calc::signal_calc_to_screen),
        to_string(get_head()->i_result));
}

```

5.8 Файл cl_calc.h

Листинг 8 – cl_calc.h

```

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

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

```

```

void signal_f(string& msg);
void handler_f(string msg);

void signal_calc_to_screen(string& msg);
void handler_calc_from_reader(string msg);
};
#endif

```

5.9 Файл cl_cancel.cpp

Листинг 9 – cl_cancel.cpp

```

#include "cl_cancel.h"

cl_cancel::cl_cancel(cl_base* p_head_object, string
s_name) :cl_base(p_head_object, s_name)
{
    this->number = 3;
}

void cl_cancel::signal_f(string& msg)
{
    cout << endl << "Signal from " << this->get_path();
    msg += " (class: 3)";
}

void cl_cancel::handler_f(string msg)
{
    cout << endl << "Signal to " << get_path() << " Text: " << msg;
}

void cl_cancel::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;
    }
}

```

5.10 Файл cl_cancel.h

Листинг 10 – cl_cancel.h

```
#ifndef __CL_CANCEL_H__
#define __CL_CANCEL_H__
#include "cl_base.h"

class cl_cancel : public cl_base
{
public:
    cl_cancel(cl_base* p_head_object, string s_name);
    void signal_f(string& msg);
    void handler_f(string msg);

    void handler_cancel_from_reader(string msg);
};
#endif
```

5.11 Файл cl_reader.cpp

Листинг 11 – cl_reader.cpp

```
#include "cl_reader.h"

cl_reader::cl_reader(cl_base* p_head_object, string
s_name) :cl_base(p_head_object, s_name)
{
    this->number = 4;
}

void cl_reader::signal_f(string& msg)
{
    cout << endl << "Signal from " << this->get_path();
    msg += " (class: 4)";
}

void cl_reader::handler_f(string msg)
{
    cout << endl << "Signal to " << get_path() << " Text:  " << msg;
}

void cl_reader::handler_reader_from_app(string msg)
{
    string s_cmd;
    cin >> s_cmd;
    if (s_cmd == "C" || s_cmd == "Off")
    {
```

```

        emit_signal(SIGNAL_D(cl_reader::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_reader::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_reader::signal_reader_to_all(string& msg)
{
}

```

5.12 Файл cl_reader.h

Листинг 12 – cl_reader.h

```

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

class cl_reader : public cl_base
{
public:
    cl_reader(cl_base* p_head_object, string s_name);
    void signal_f(string& msg);
    void handler_f(string msg);
    void handler_reader_from_app(string msg);
    void signal_reader_to_all(string& msg);
};
#endif

```

5.13 Файл cl_screen.cpp

Листинг 13 – cl_screen.cpp

```
#include "cl_screen.h"

cl_screen::cl_screen(cl_base* p_head_object, string
s_name) :cl_base(p_head_object, s_name)
{
    this->number = 5;
}

void cl_screen::signal_f(string& msg)
{
    cout << endl << "Signal from " << this->get_path();
    msg += " (class: 5)";
}

void cl_screen::handler_f(string msg)
{
    cout << endl << "Signal to " << get_path() << " Text:  " << msg;
}

void cl_screen::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";
        get_head()->i_result = 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 (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_screen::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;
}

```

5.14 Файл cl_screen.h

Листинг 14 – cl_screen.h

```

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

class cl_screen : public cl_base
{
public:
    cl_screen(cl_base* p_head_object, string s_name);
    void signal_f(string& msg);
    void handler_f(string msg);

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

```

5.15 Файл cl_shift.cpp

Листинг 15 – cl_shift.cpp

```

#include "cl_shift.h"

cl_shift::cl_shift(cl_base* p_head_object, string
s_name) :cl_base(p_head_object, s_name)
{
    this->number = 6;
}

```

```

void cl_shift::signal_f(string& msg)
{
    cout << endl << "Signal from " << this->get_path();
    msg += " (class: 6)";
}

void cl_shift::handler_f(string msg)
{
    cout << endl << "Signal to " << get_path() << " Text:  " << msg;
}

void cl_shift::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_shift::signal_shift_to_screen),
        to_string(get_head()->i_result));
}

void cl_shift::signal_shift_to_screen(string& msg)
{
}

```

5.16 Файл cl_shift.h

Листинг 16 – cl_shift.h

```

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

class cl_shift : public cl_base
{
public:
    cl_shift(cl_base* p_head_object, string s_name);
    void signal_f(string& msg);
    void handler_f(string msg);

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

```

```
};  
#endif
```

5.17 Файл main.cpp

Листинг 17 – main.cpp

```
#include "cl_application.h"  
int main()  
{  
    cl_application obj ( nullptr );  
    obj.build_tree_objects ( );  
    return obj.exec_app ( );  
}
```

6 ТЕСТИРОВАНИЕ

Результат тестирования программы представлен в таблице 16.

Таблица 16 – 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

Изучение курса "Объектно-ориентированное программирование" позволило мне ознакомиться с методологией объектно-ориентированного программирования. Я освоил концепцию классов и объектов и их использования для описания систем. У меня появилось понимание таких основных компонентов парадигмы объектно-ориентированного программирования, как наследование, полиморфизм и инкапсуляция. Обучение проходило на языке программирования C++, что дало мне возможность изучить такие понятия, как указатели, ссылки, встраиваемые и дружественные функции, дружественные классы, виртуальные методы, статические методы, абстрактные классы, перегрузка и переопределение функций, контейнеры и структуры, шаблоны функций и классов.

- Отзыв о системе для написания кода "Avrora":
 - Плюсы:
 - Удобное хранение кода и отчёта в облаке приложения
 - Возможности для быстрого создания блок-схем
 - Оперативное внесение изменений и исправление ошибок
 - Быстрое создание отчёта
 - Минусы:
 - Отсутствие привычного цветового выделения "endl", "\n"
 - Отсутствие тёмной темы Visual studio
 - Отсутствие вывода ввода при ошибке во время контрольного тестирования
 - Частые проблемы с соединением к серверам
 - Проблемы с границами страниц в редакторе блок-схем
 - Отсутствие возможности посмотреть код, пока работа не принята или не возвращена

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/methodichescoc_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).