Здесь будет титульник, листай ниже

СОДЕРЖАНИЕ

1 ПОСТАНОВКА ЗАДАЧИ

Множественное наследование

Даны 8 классов, которые нумеруются от 1 до 8. Классы 2, 3, 4 и 5 наследованы от первого класса. Шестой класс от второго и третьего. Седьмой от четвертого и пятого. Восьмой от шестого и седьмого.

У каждого класса есть параметризированный конструктор с одним параметром строкового типа и закрытое свойство строкового типа для хранения наименования объекта класса. Значение данного свойства определяется в параметризированном конструкторе согласно шаблону:

```
«значение строкового параметра»_«номер класса»
```

У каждого класса есть метод в открытом разделе с одинаковым наименованием, который возвращает наименование объекта класса.

В реализации конструкторов со второго по восьмой класс, вызвать конструктор или конструкторы родительских классов. При вызове передать в качестве параметра выражение:

```
«параметр производного класса + «_» + «номер производного класса»
```

Например, для конструктора второго класса

```
cl_2 :: cl_2 ( string s_name ) : cl_1 ( s_name + “_2” )
```

В основной функции реализовать алгоритм:

1. Объявить один указатель на объект класса x.
2. Объявить переменную строкового типа.
3. Ввести значение строковой переменной. Вводимое значение является идентификатором.
4. Создать объект класса 8 посредством параметризированного конструктора, передав в качестве аргумента строковую переменную.

5. Адрес созданного объекта присвоить указателю на объект класса x.

6. Используя только указатель на объект класса x вывести имена всех объектов в составе объекта класса 8 и имя самого объекта класса 8. Вывод выполнить построчно, упорядочивая согласно возрастанию номеров класса. Наименования объектов первого класса вывести последовательно для производных объектов 2,3.4 и 5 класса.

Наследственность реализовать так, чтобы всего объектов было 10 и обеспечить вывод по аналогии приведенному примеру вывода.

1.1 Описание входных данных

Первая строка:

«идентификатор»

Пример ввода

```
Object
```

1.2 Описание выходных данных

Построчно (одиннадцать строк):

«наименование объекта»

Пример вывода:

```
Object_8_6_2_1
Object_8_6_3_1
Object_8_1
Object_8_1
Object_8_6_2
Object_8_6_3
Object_8_7_4
Object_8_7_5
Object_8_6
```

Object_8_7
Object_8

2 МЕТОД РЕШЕНИЯ

Для решения задачи используется:

- объект x класса cl_8 предназначен для Вывода имен объектов;
- объект cl класса cl_8 предназначен для Инициализация x;
- new - оператор выделения динамической памяти;
- cin/cout - объекты стандартного потока ввода вывода.

Класс cl_1:

- свойства/поля:
  - поле имя:
    - наименование — name;
    - тип — string;
    - модификатор доступа — private;
- функционал:
  - метод cl_1 — параметризированный конструктор;
  - метод get_name — получение имени.

Класс cl_2:

- свойства/поля:
  - поле имя:
    - наименование — name;
    - тип — string;
    - модификатор доступа — private;
- функционал:
  - метод cl_2 — параметризированный конструктор;
  - метод get_name — получение имени.

Класс cl_3:

- свойства/поля:

    - поле имя:
        - наименование — name;
        - тип — string;
        - модификатор доступа — private;
- функционал:
    - метод cl_3 — параметризированный конструктор;
    - метод get_name — получение имени.

Класс cl_4:

- свойства/поля:
    - поле имя:
        - наименование — name;
        - тип — string;
        - модификатор доступа — private;
- функционал:
    - метод cl_4 — параметризированный конструктор;
    - метод get_name — получение имени.

Класс cl_5:

- свойства/поля:
    - поле имя:
        - наименование — name;
        - тип — string;
        - модификатор доступа — private;
- функционал:
    - метод cl_5 — параметризированный конструктор;
    - метод get_name — получение имени.

Класс cl_6:

- свойства/поля:

    - поле имя:
        - наименование — name;
        - тип — string;
        - модификатор доступа — private;
- функционал:
    - метод cl_6 — параметризированный конструктор;
    - метод get_name — получение имени.

Класс cl_7:

- свойства/поля:
    - поле имя:
        - наименование — name;
        - тип — string;
        - модификатор доступа — private;
- функционал:
    - метод cl_7 — параметризированный конструктор;
    - метод get_name — получение имени.

Класс cl_8:

- свойства/поля:
    - поле имя:
        - наименование — name;
        - тип — string;
        - модификатор доступа — private;
- функционал:
    - метод cl_8 — параметризированный конструктор;
    - метод get_name — получение имени.

Таблица 1 – Иерархия наследования классов

№	Имя класса	Классы-наследники	Модификатор доступа при наследовании	Описание	Номер
1	cl_1			Базовый класс	
		cl_2	public		2
		cl_3	public		3
		cl_4	virtual public		4
		cl_5	virtual public		5
2	cl_2			Базовый класс	
		cl_6	public		6
3	cl_3			Базовый класс	
		cl_6	public		6
4	cl_4			Базовый класс	
		cl_7	public		7
5	cl_5			Базовый класс	
		cl_7	public		7
6	cl_6			Базовый класс	
		cl_8	public		8
7	cl_7			Базовый класс	
		cl_8	public		8
8	cl_8			Базовый класс	

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 конструктора класса cl_1

Функционал: параметризированный конструктор.

Параметры: string name - имя объекта.

Алгоритм конструктора представлен в таблице 2.

Таблица 2 – Алгоритм конструктора класса cl_1

№	Предикат	Действия	№ перехода
1		Установка значения поля name равному значению параметра name + _1	∅

3.2 Алгоритм метода get_name класса cl_1

Функционал: получение имени.

Параметры: нет.

Возвращаемое значение: void.

Алгоритм метода представлен в таблице 3.

Таблица 3 – Алгоритм метода get_name класса cl_1

№	Предикат	Действия	№ перехода
1		Возврат значения поля name	∅

3.3 Алгоритм конструктора класса cl_2

Функционал: параметризированный конструктор.

Параметры: string name - имя объекта.

Алгоритм конструктора представлен в таблице 4.

Таблица 4 – Алгоритм конструктора класса cl_2

№	Предикат	Действия	№ перехода
1		Передача параметризированному родительскому конструктору класса cl_1 параметра name + _2	2
2		Установка значения поля name равному значению параметра name + _2	∅

3.4 Алгоритм метода get_name класса cl_2

Функционал: получение имени.

Параметры: нет.

Возвращаемое значение: void.

Алгоритм метода представлен в таблице 5.

Таблица 5 – Алгоритм метода get_name класса cl_2

№	Предикат	Действия	№ перехода
1		Возврат значения поля name	∅

3.5 Алгоритм метода get_name класса cl_3

Функционал: получение имени.

Параметры: нет.

Возвращаемое значение: void.

Алгоритм метода представлен в таблице 6.

Таблица 6 – Алгоритм метода get_name класса cl_3

№	Предикат	Действия	№ перехода
1		Возврат значения поля name	∅

3.6 Алгоритм метода get_name класса cl_4

Функционал: получение имени.

Параметры: нет.

Возвращаемое значение: void.

Алгоритм метода представлен в таблице 7.

Таблица 7 – Алгоритм метода get_name класса cl_4

№	Предикат	Действия	№ перехода
1		Возврат значения поля name	∅

3.7 Алгоритм метода get_name класса cl_5

Функционал: получение имени.

Параметры: нет.

Возвращаемое значение: void.

Алгоритм метода представлен в таблице 8.

Таблица 8 – Алгоритм метода get_name класса cl_5

№	Предикат	Действия	№ перехода
1		Возврат значения поля name	∅

3.8 Алгоритм метода get_name класса cl_6

Функционал: получение имени.

Параметры: нет.

Возвращаемое значение: void.

Алгоритм метода представлен в таблице 9.

Таблица 9 – Алгоритм метода get_name класса cl_6

№	Предикат	Действия	№ перехода
1		Возврат значения поля name	∅

3.9 Алгоритм метода get_name класса cl_7

Функционал: получение имени.

Параметры: нет.

Возвращаемое значение: void.

Алгоритм метода представлен в таблице 10.

Таблица 10 – Алгоритм метода get_name класса cl_7

№	Предикат	Действия	№ перехода
1		Возврат значения поля name	∅

3.10 Алгоритм метода get_name класса cl_8

Функционал: получение имени.

Параметры: нет.

Возвращаемое значение: void.

Алгоритм метода представлен в таблице 11.

Таблица 11 – Алгоритм метода get_name класса cl_8

№	Предикат	Действия	№ перехода
1		Возврат значения поля name	∅

3.11 Алгоритм конструктора класса cl_3

Функционал: параметризированный конструктор.

Параметры: string name - имя объекта.

Алгоритм конструктора представлен в таблице 12.

Таблица 12 – Алгоритм конструктора класса cl_3

№	Предикат	Действия	№ перехода
1		Передача параметризированному родительскому конструктору класса cl_1 параметра name + _3	2
2		Установка значения поля name равному значению параметра name + _3	∅

3.12 Алгоритм конструктора класса cl_4

Функционал: параметризированный конструктор.

Параметры: string name - имя объекта.

Алгоритм конструктора представлен в таблице 13.

Таблица 13 – Алгоритм конструктора класса cl_4

№	Предикат	Действия	№ перехода
1		Передача параметризированному родительскому конструктору класса cl_1 параметра name + _4	2
2		Установка значения поля name равному значению параметра name + _4	∅

3.13 Алгоритм конструктора класса cl_5

Функционал: параметризированный конструктор.

Параметры: string name - имя объекта.

Алгоритм конструктора представлен в таблице 14.

Таблица 14 – Алгоритм конструктора класса cl_5

№	Предикат	Действия	№ перехода
1		Передача параметризированному родительскому конструктору класса cl_1 параметра name + _5	2
2		Установка значения поля name равному значению параметра name + _5	∅

3.14 Алгоритм конструктора класса cl_6

Функционал: параметризированный конструктор.

Параметры: string name - имя объекта.

Алгоритм конструктора представлен в таблице 15.

Таблица 15 – Алгоритм конструктора класса cl_6

№	Предикат	Действия	№ перехода
1		Передача параметризированному родительскому конструктору класса cl_2 и cl_3 параметра name + _6	2
2		Установка значения поля name равному значению параметра name + _6	∅

3.15 Алгоритм конструктора класса cl_7

Функционал: параметризированный конструктор.

Параметры: string name - имя объекта.

Алгоритм конструктора представлен в таблице 16.

Таблица 16 – Алгоритм конструктора класса cl_7

№	Предикат	Действия	№ перехода
1		Передача параметризированному родительскому конструктору класса	2

№	Предикат	Действия	№ перехода
		cl_4 и cl_5 параметра name + _7	
2		Установка значения поля name равному значению параметра name + _7	∅

3.16 Алгоритм конструктора класса cl_8

Функционал: параметризированный конструктор.

Параметры: string name - имя объекта.

Алгоритм конструктора представлен в таблице 17.

Таблица 17 – Алгоритм конструктора класса cl_8

№	Предикат	Действия	№ перехода
1		Передача параметризированному родительскому конструктору класса cl_6 и cl_7 параметра name + _8	2
2		Установка значения поля name равному значению параметра name + _8	∅

3.17 Алгоритм функции main

Функционал: главная функция программы.

Параметры: нет.

Возвращаемое значение: int - код ошибки.

Алгоритм функции представлен в таблице 18.

Таблица 18 – Алгоритм функции main

№	Предикат	Действия	№ перехода
1		объявление указателя x класса cl_8	2
2		объявление переменной name типа string	3
3		ввод значения name	4

№	Предикат	Действия	№ перехода
4		инициализация переменной cl объекта класса cl_8 с передачей name как параметр конструктору	5
5		присвоение x указателя на cl	6
6		Вывод значения вызова метода get_name у объекта по указателю x с приведённым типом на cl_1 по cl_2 по cl_6	7
7		Вывод значения вызова метода get_name у объекта по указателю x с приведённым типом на cl_1 по cl_3 по cl_6	8
8		Вывод значения вызова метода get_name у объекта по указателю x с приведённым типом на cl_1 по cl_4 по cl_7	9
9		Вывод значения вызова метода get_name у объекта по указателю x с приведённым типом на cl_1 по cl_5 по cl_7	10
10		Вывод значения вызова метода get_name у объекта по указателю x с приведённым типом на cl_2 по cl_6	11
11		Вывод значения вызова метода get_name у объекта по указателю x с приведённым типом на cl_3 по cl_6	12
12		Вывод значения вызова метода get_name у объекта по указателю x с приведённым типом на cl_4 по cl_7	13
13		Вывод значения вызова метода get_name у объекта по указателю x с приведённым типом на cl_5 по cl_7	14
14		Вывод значения вызова метода get_name у объекта по указателю x с приведённым типом на cl_6	15
15		Вывод значения вызова метода get_name у объекта по указателю x с приведённым типом на cl_7	16
16		Вывод значения вызова метода get_name у объекта по указателю x	17
17		Возврат значения 0	∅

4 БЛОК-СХЕМЫ АЛГОРИТМОВ

Представим описание алгоритмов в графическом виде на рисунках 1-5.

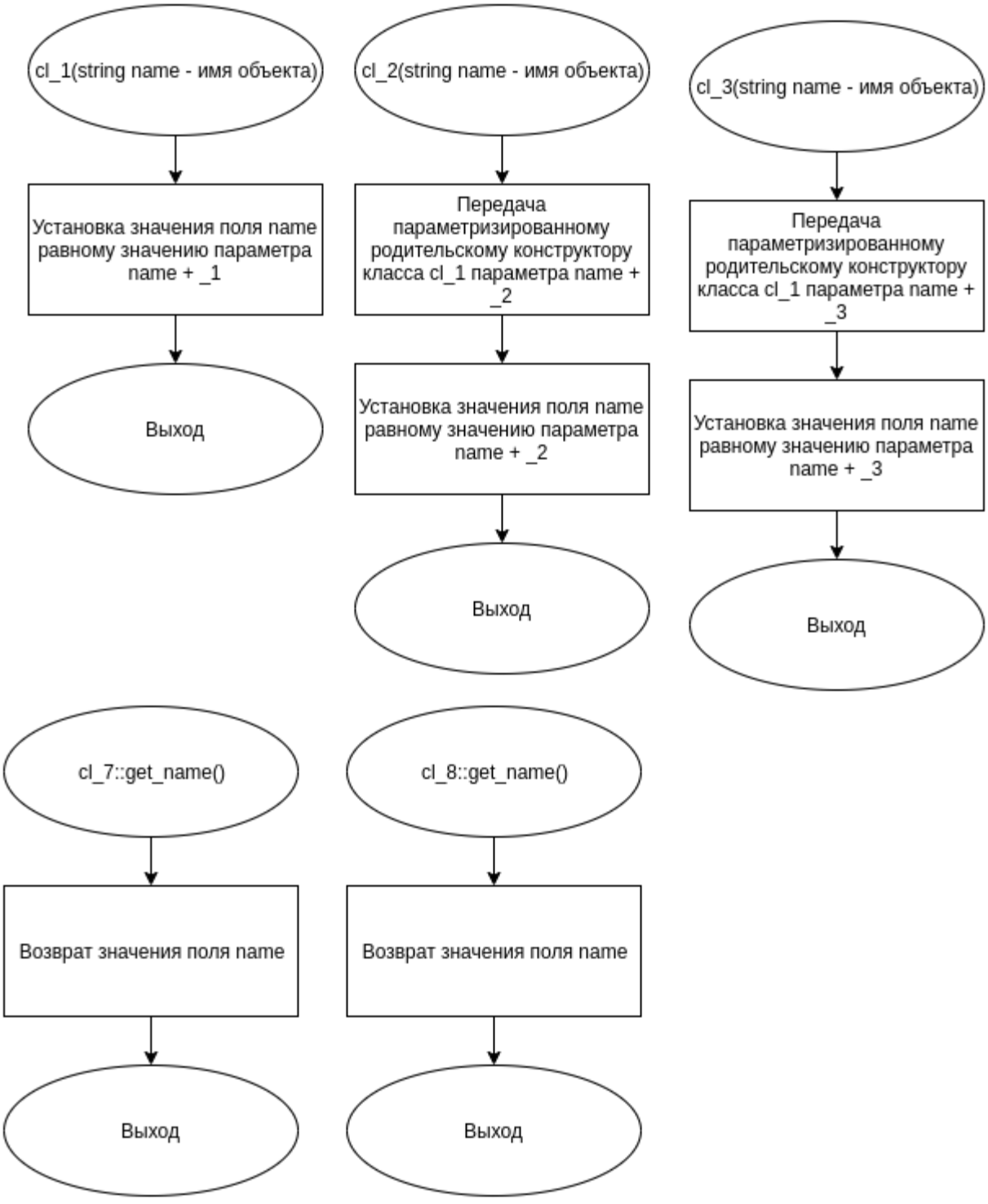

Рисунок 1 – Блок-схема алгоритма

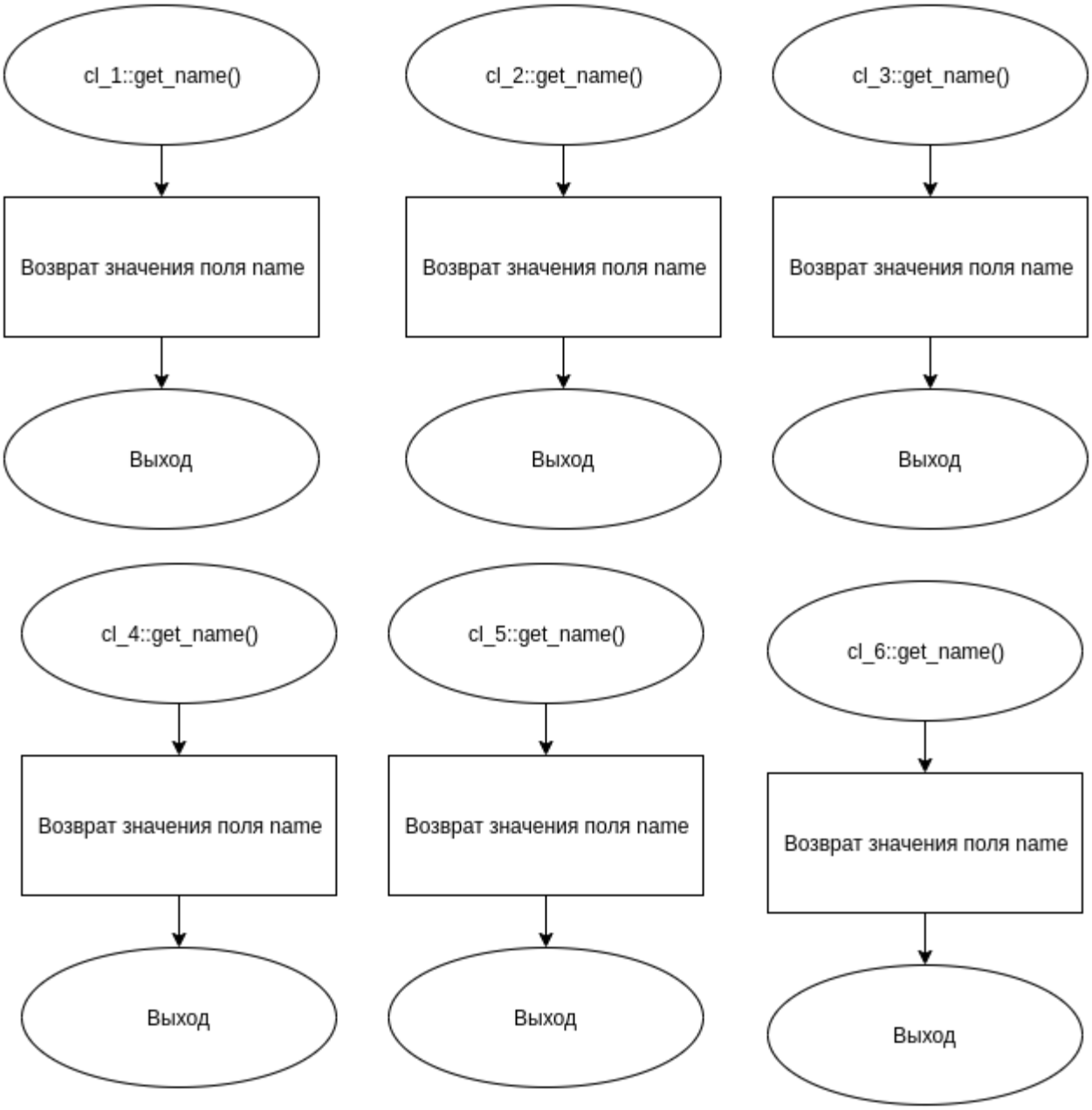

Рисунок 2 – Блок-схема алгоритма

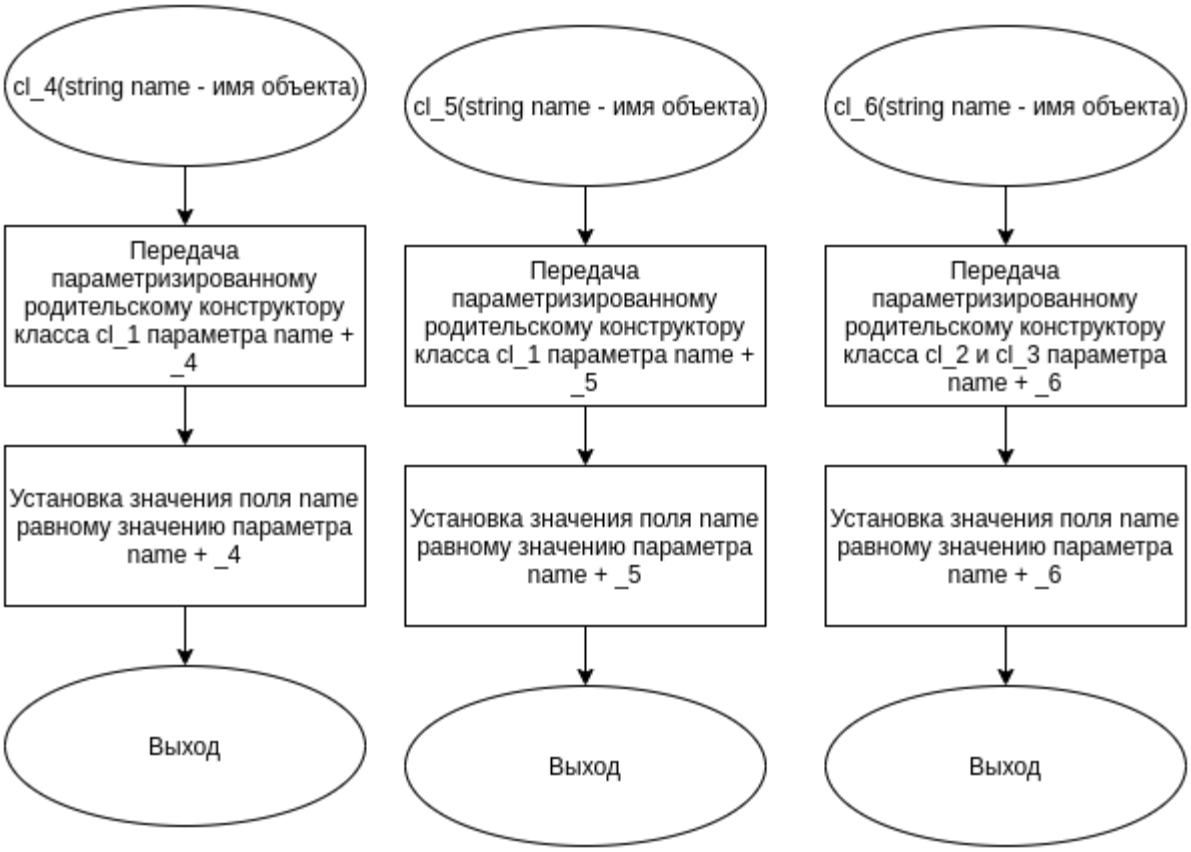

Рисунок 3 – Блок-схема алгоритма

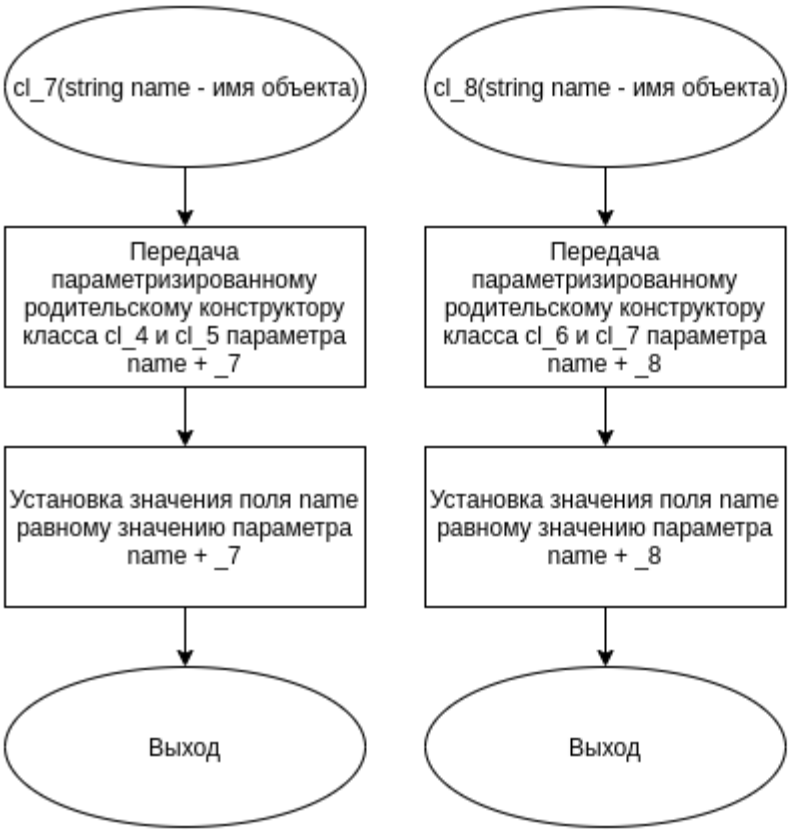

Рисунок 4 – Блок-схема алгоритма

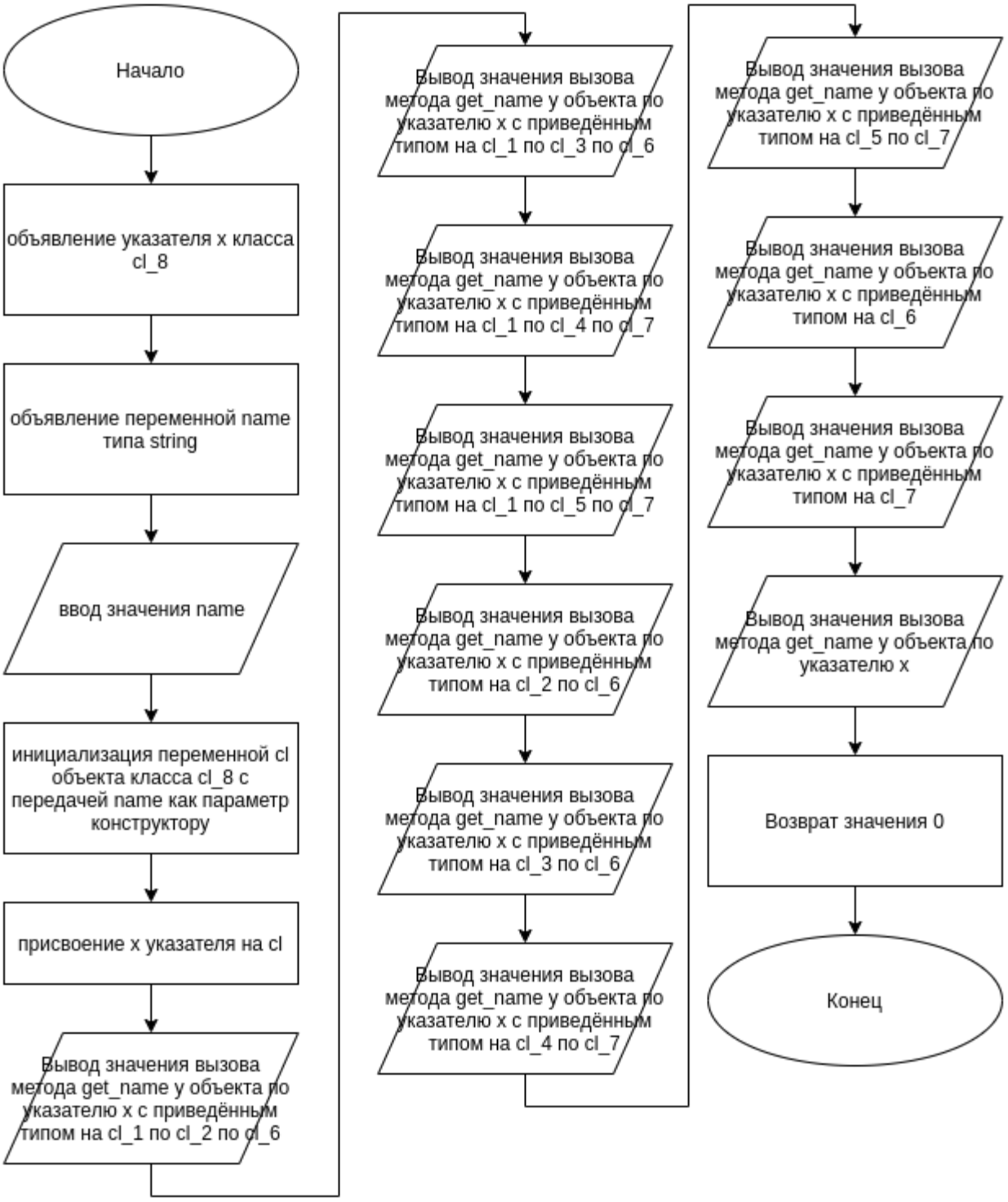

Рисунок 5 – Блок-схема алгоритма

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(string name) {
   this->name = name+"_1";
}

string cl_1::get_name() {
   return this->name;
}
```

5.2 Файл cl_1.h

Листинг 2 – cl_1.h

```
#ifndef __CL_1__H
#define __CL_1__H
#include <iostream>
#include <string>

using namespace std;

class cl_1 {
private:
   string name;
public:
   cl_1(string name);
   string get_name();
};


#endif
```

5.3 Файл cl_2.cpp

Листинг 3 – cl_2.cpp

```
#include "cl_2.h"


cl_2::cl_2(string name) : cl_1 (name + "_2"){
   this->name = name+"_2";
}

string cl_2::get_name() {
   return this->name;
}
```

5.4 Файл cl_2.h

Листинг 4 – cl_2.h

```
#ifndef __CL_2__H
#define __CL_2__H

#include "cl_1.h"
class cl_2 : public cl_1 {
private:
   string name;
public:
   cl_2(string name);
   string get_name();
};

#endif
```

5.5 Файл cl_3.cpp

Листинг 5 – cl_3.cpp

```
#include "cl_3.h"


cl_3::cl_3(string name) : cl_1 (name + "_3"){
   this->name = name+"_3";
}
```

```
string cl_3::get_name() {
   return this->name;
}
```

5.6 Файл cl_3.h

Листинг 6 – cl_3.h

```
#ifndef __CL_3__H
#define __CL_3__H

#include "cl_1.h"
class cl_3 : public cl_1 {
private:
   string name;
public:
   cl_3(string name);
   string get_name();
};

#endif
```

5.7 Файл cl_4.cpp

Листинг 7 – cl_4.cpp

```
#include "cl_4.h"

cl_4::cl_4(string name) : cl_1 (name + "_4") {
   this->name = name+"_4";
}

string cl_4::get_name() {
   return this->name;
}
```

5.8 Файл cl_4.h

Листинг 8 – cl_4.h

```
#ifndef __CL_4__H
#define __CL_4__H

#include "cl_1.h"
class cl_4 : public virtual cl_1 {
private:
   string name;
public:
   cl_4(string name);
   string get_name();
};

#endif
```

5.9 Файл cl_5.cpp

Листинг 9 – cl_5.cpp

```
#include "cl_5.h"

cl_5::cl_5(string name) : cl_1 (name + "_5") {
   this->name = name+"_5";
}

string cl_5::get_name() {
   return this->name;
}
```

5.10 Файл cl_5.h

Листинг 10 – cl_5.h

```
#ifndef __CL_5__H
#define __CL_5__H

#include "cl_1.h"
class cl_5 : public virtual cl_1 {
private:
   string name;
```

```
public:
   cl_5(string name);
   string get_name();
};

#endif
```

5.11 Файл cl_6.cpp

Листинг 11 – cl_6.cpp

```
#include "cl_6.h"

cl_6::cl_6(string name) : cl_2 (name + "_6"), cl_3 (name + "_6"){
   this->name = name+"_6";
}

string cl_6::get_name() {
   return this->name;
}
```

5.12 Файл cl_6.h

Листинг 12 – cl_6.h

```
#ifndef __CL_6__H
#define __CL_6__H

#include "cl_2.h"
#include "cl_3.h"
class cl_6 : public cl_2, public cl_3 {
private:
   string name;
public:
   cl_6(string name);
   string get_name();
};

#endif
```

5.13 Файл cl_7.cpp

Листинг 13 – cl_7.cpp

```
#include "cl_7.h"

cl_7::cl_7(string name) : cl_4 (name + "_7"), cl_5 (name + "_7"), cl_1(name
+ "_7"){
   this->name = name+"_7";
}

string cl_7::get_name() {
   return this->name;
}
```

5.14 Файл cl_7.h

Листинг 14 – cl_7.h

```
#ifndef __CL_7__H
#define __CL_7__H

#include "cl_4.h"
#include "cl_5.h"
class cl_7 : public cl_4, public cl_5 {
private:
   string name;
public:
   cl_7(string name);
   string get_name();
};

#endif
```

5.15 Файл cl_8.cpp

Листинг 15 – cl_8.cpp

```
#include "cl_8.h"

cl_8::cl_8(string name) : cl_6 (name + "_8"), cl_7 (name + "_8"), cl_1(name
+ "_8"){
   this->name = name+"_8";
```

```
}

string cl_8::get_name() {
   return this->name;
}
```

5.16 Файл cl_8.h

Листинг 16 – cl_8.h

```
#ifndef __CL_8__H
#define __CL_8__H

#include "cl_6.h"
#include "cl_7.h"
class cl_8 : public cl_6, public cl_7 {
private:
   string name;
public:
   cl_8(string name);
   string get_name();
};

#endif
```

5.17 Файл main.cpp

Листинг 17 – main.cpp

```
#include <stdlib.h>
#include <stdio.h>

#include "cl_8.h"

int main()
{
   cl_8 *x;
   string name;
   cin >> name;

   cl_8 cl(name);
   x = &cl;

   cout << ((cl_1*)(cl_2*)(cl_6*)x)->get_name() << endl;
   cout << ((cl_1*)(cl_3*)(cl_6*)x)->get_name() << endl;
```

```
        cout << ((cl_1*)(cl_4*)(cl_7*)x)->get_name() << endl;
        cout << ((cl_1*)(cl_5*)(cl_7*)x)->get_name() << endl;
        cout << ((cl_2*)(cl_6*)x)->get_name() << endl;
        cout << ((cl_3*)(cl_6*)x)->get_name() << endl;
        cout << ((cl_4*)(cl_7*)x)->get_name() << endl;
        cout << ((cl_5*)(cl_7*)x)->get_name() << endl;
        cout << ((cl_6*)x)->get_name() << endl;
        cout << ((cl_7*)x)->get_name() << endl;
        cout << x->get_name();
        return(0);
    }
```

6 ТЕСТИРОВАНИЕ

Результат тестирования программы представлен в таблице 19.

Таблица 19 – Результат тестирования программы

Входные данные	**Ожидаемые выходные данные**	**Фактические выходные данные**

`Object`	`Object_8_6_2_1`
	`Object_8_6_3_1`
	`Object_8_1`
	`Object_8_1`
	`Object_8_6_2`
	`Object_8_6_3`
	`Object_8_7_4`
	`Object_8_7_5`
	`Object_8_6`
	`Object_8_7`
	`Object_8`
		`Object_8_6_2_1`
		`Object_8_6_3_1`
		`Object_8_1`
		`Object_8_1`
		`Object_8_6_2`
		`Object_8_6_3`
		`Object_8_7_4`
		`Object_8_7_5`
		`Object_8_6`
		`Object_8_7`
		`Object_8`

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ГОСТ 19 Единая система программной документации.

2. Методическое пособие студента для выполнения практических заданий, контрольных и курсовых работ по дисциплине «Объектно-ориентированное программирование» [Электронный ресурс] – URL: https://mirea.aco-avrora.ru/student/files/methodichescoe_posobie_dlya_laboratornyh_rabot_3.pdf (дата обращения 05.05.2021).

3. Приложение к методическому пособию студента по выполнению заданий в рамках курса «Объектно-ориентированное программирование» [Электронный ресурс]. URL: https://mirea.aco-avrora.ru/student/files/Prilozheniye_k_methodichke.pdf (дата обращения 05.05.2021).

4. Шилдт Г. C++: базовый курс. 3-е изд. Пер. с англ.. — М.: Вильямс, 2019. — 624 с.

5. Видео лекции по курсу «Объектно-ориентированное программирование» [Электронный ресурс]. АСО «Аврора».

6. Антик М.И. Дискретная математика [Электронный ресурс]: Учебное пособие /Антик М.И., Казанцева Л.В. — М.: МИРЭА — Российский технологический университет, 2018 — 1 электрон. опт. диск (CD-ROM).