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:

- Объявить один указатель на объект класса x.
- Объявить переменную строкового типа.
- Ввести значение строковой переменной. Вводимое значение является идентификатором.
- Создать объект класса 8 посредством параметризированного конструктора, передав в качестве аргумента строковую переменную.
- Адрес созданного объекта присвоить указателю на объект класса x.
- 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

Объекты стандартного потока ввода/вывода cin, cout;

Объект obj класса cl_8;

Класс cl_1:

- Свойства/поля:
  - Поле, отвечающее за наименование объекта:
    - Наименование - name;
    - Тип данных - string;
    - Модификатор доступа - private;
- Функционал:
  - Параметризированный конструктор cl_1(), который задаёт значение закрытому полю;
  - Метод getName() возвращает значение закрытого поля;

Класс cl_2:

- Свойства/поля:
  - Поле, отвечающее за наименование объекта:
    - Наименование - name;
    - Тип данных - string;
    - Модификатор доступа - private;
- Функционал:
  - Параметризированный конструктор cl_2(), который задаёт значение закрытому полю;
  - Метод getName() возвращает значение закрытого поля;

Класс cl_3:

- Свойства/поля:
  - Поле, отвечающее за наименование объекта:

        - Наименование - name;
        - Тип данных - string;
        - Модификатор доступа - private;
- Функционал:
    - Параметризированный конструктор cl_3(), который задаёт значение закрытому полю;
    - Метод getName() возвращает значение закрытого поля;

Класс cl_4:

- Свойства/поля:
    - Поле, отвечающее за наименование объекта:
        - Наименование - name;
        - Тип данных - string;
        - Модификатор доступа - private;
- Функционал:
    - Параметризированный конструктор cl_4(), который задаёт значение закрытому полю;
    - Метод getName() возвращает значение закрытого поля;

Класс cl_5:

- Свойства/поля:
    - Поле, отвечающее за наименование объекта:
        - Наименование - name;
        - Тип данных - string;
        - Модификатор доступа - private;
- Функционал:
    - Параметризированный конструктор cl_5(), который задаёт значение закрытому полю;
    - Метод getName() возвращает значение закрытого поля;

Класс cl_6:

- Свойства/поля:
  - Поле, отвечающее за наименование объекта:
    - Наименование - name;
    - Тип данных - string;
    - Модификатор доступа - private;
- Функционал:
  - Параметризированный конструктор cl_6(), который задаёт значение закрытому полю;
  - Метод getName() возвращает значение закрытого поля;

Класс cl_7:

- Свойства/поля:
  - Поле, отвечающее за наименование объекта:
    - Наименование - name;
    - Тип данных - string;
    - Модификатор доступа - private;
- Функционал:
  - Параметризированный конструктор cl_7(), который задаёт значение закрытому полю;
  - Метод getName() возвращает значение закрытого поля;

Класс cl_8:

- Свойства/поля:
  - Поле, отвечающее за наименование объекта:
    - Наименование - name;
    - Тип данных - string;
    - Модификатор доступа - private;
- Функционал:

- Параметризированный конструктор cl_8(), который задаёт значение закрытому полю;
- Метод getName() возвращает значение закрытого поля.

Таблица 1 – Иерархия наследования классов

№	Имя класса	Классы наследники	Модификатор доступа при наследовании	Описание	Номер	Комментарий
1	cl_1			Базовый класс		
		cl_2	public		2	
		cl_3	public		3	
		cl_4	public		4	Виртуальное наследование
		cl_5	public		5	Виртуальное наследование
2	cl_2			Производный класс от класса cl_1		
		cl_6	public		6	
3	cl_3			Производный		

				класс от класса cl_1		
		cl_6	public		6	
4	cl_4			Производный класс от класса cl_1		
		cl_7	public		7	
5	cl_5			Производный класс от класса cl_1		
		cl_7	public		7	
6	cl_6			Производный класс от классов cl_2 и cl_3		
		cl_8	public		8	
7	cl_7			Производный класс от классов cl_4 и cl_5		
		cl_8	public		8	

8	cl_8			Производный класс от классов cl_6 и cl_7		

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: Присвоение значения закрытому полю.

Параметры: string name.

Алгоритм конструктора представлен в таблице 2.

Таблица 2 – Алгоритм конструктора класса cl_1

№	Предикат	Действия	№ перехода
1		Присвоение закрытому полю name значение name + "_1"	∅

3.2 Алгоритм конструктора класса cl_2

Функционал: Присвоение значения закрытому полю.

Параметры: string name.

Алгоритм конструктора представлен в таблице 3.

Таблица 3 – Алгоритм конструктора класса cl_2

№	Предикат	Действия	№ перехода
1		Присвоение закрытому полю name значение name + "_2"	∅

3.3 Алгоритм конструктора класса cl_3

Функционал: Присвоение значения закрытому полю.

Параметры: string name.

Алгоритм конструктора представлен в таблице 4.

Таблица 4 – Алгоритм конструктора класса cl_3

№	Предикат	Действия	№ перехода
1		Присвоение закрытому полю name значение name + "_3"	∅

3.4 Алгоритм конструктора класса cl_4

Функционал: Присвоение значения закрытому полю.

Параметры: string name.

Алгоритм конструктора представлен в таблице 5.

Таблица 5 – Алгоритм конструктора класса cl_4

№	Предикат	Действия	№ перехода
1		Присвоение закрытому полю name значение name + "_4"	∅

3.5 Алгоритм конструктора класса cl_5

Функционал: Присвоение значения закрытому полю.

Параметры: string name.

Алгоритм конструктора представлен в таблице 6.

Таблица 6 – Алгоритм конструктора класса cl_5

№	Предикат	Действия	№ перехода
1		Присвоение закрытому полю name значение name + "_5"	∅

3.6 Алгоритм конструктора класса cl_6

Функционал: Присвоение значения закрытому полю.

Параметры: string name.

Алгоритм конструктора представлен в таблице 7.

Таблица 7 – Алгоритм конструктора класса cl_6

№	Предикат	Действия	№ перехода
1		Присвоение закрытому полю name значение name + "_6"	∅

3.7 Алгоритм конструктора класса cl_7

Функционал: Присвоение значения закрытому полю.

Параметры: string name.

Алгоритм конструктора представлен в таблице 8.

Таблица 8 – Алгоритм конструктора класса cl_7

№	Предикат	Действия	№ перехода
1		Присвоение закрытому полю name значение name + "_7"	∅

3.8 Алгоритм конструктора класса cl_8

Функционал: Присвоение значения закрытому полю.

Параметры: string name.

Алгоритм конструктора представлен в таблице 9.

Таблица 9 – Алгоритм конструктора класса cl_8

№	Предикат	Действия	№ перехода
1		Присвоение закрытому полю name значение name + "_8"	∅

3.9 Алгоритм метода getName класса cl_1

Функционал: Возврат значения скрытого свойства.

Параметры: .

Возвращаемое значение: string.

Алгоритм метода представлен в таблице 10.

Таблица 10 – Алгоритм метода getName класса cl_1

№	Предикат	Действия	№ перехода
1		Возврат значения закрытого поля name	∅

3.10 Алгоритм метода getName класса cl_2

Функционал: Возврат значения скрытого свойства.

Параметры: .

Возвращаемое значение: string.

Алгоритм метода представлен в таблице 11.

Таблица 11 – Алгоритм метода getName класса cl_2

№	Предикат	Действия	№ перехода
1		Возврат значения закрытого поля name	∅

3.11 Алгоритм метода getName класса cl_3

Функционал: Возврат значения скрытого свойства.

Параметры: .

Возвращаемое значение: string.

Алгоритм метода представлен в таблице 12.

Таблица 12 – Алгоритм метода getName класса cl_3

№	Предикат	Действия	№ перехода
1		Возврат значения закрытого поля name	∅

3.12 Алгоритм метода getName класса cl_4

Функционал: Возврат значения скрытого свойства.

Параметры: .

Возвращаемое значение: string.

Алгоритм метода представлен в таблице 13.

Таблица 13 – Алгоритм метода getName класса cl_4

№	Предикат	Действия	№ перехода
1		Возврат значения закрытого поля name	∅

3.13 Алгоритм метода getName класса cl_5

Функционал: Возврат значения скрытого свойства.

Параметры: .

Возвращаемое значение: string.

Алгоритм метода представлен в таблице 14.

Таблица 14 – Алгоритм метода getName класса cl_5

№	Предикат	Действия	№ перехода
1		Возврат значения закрытого поля name	∅

3.14 Алгоритм метода getName класса cl_6

Функционал: Возврат значения скрытого свойства.

Параметры: .

Возвращаемое значение: string.

Алгоритм метода представлен в таблице 15.

Таблица 15 – Алгоритм метода getName класса cl_6

№	Предикат	Действия	№ перехода
1		Возврат значения закрытого поля name	∅

3.15 Алгоритм метода getName класса cl_7

Функционал: Возврат значения скрытого свойства.

Параметры: .

Возвращаемое значение: string.

Алгоритм метода представлен в таблице 16.

Таблица 16 – Алгоритм метода getName класса cl_7

№	Предикат	Действия	№ перехода
1		Возврат значения закрытого поля name	∅

3.16 Алгоритм метода getName класса cl_8

Функционал: Возврат значения скрытого свойства.

Параметры: .

Возвращаемое значение: string.

Алгоритм метода представлен в таблице 17.

Таблица 17 – Алгоритм метода getName класса cl_8

№	Предикат	Действия	№ перехода
1		Возврат значения закрытого поля name	∅

3.17 Алгоритм функции main

Функционал: Основная программа.

Параметры: .

Возвращаемое значение: int - код возврата.

Алгоритм функции представлен в таблице 18.

Таблица 18 – Алгоритм функции main

№	Предикат	Действия	№ перехода
1		Объявление указателя pObj на cl_8	2
2		Объявление строковой переменной name	3
3		Ввод значения переменной name	4
4		Создание объекта obj класса cl_8 с помощью параметризированного конструктора с параметром name	5
5		Присвоение указателю pObj ссылки на объект obj	6
6		Вызов метода getName() через указатель pObj для объекта класса cl_1, созданного объектом класса cl_2, и вывод его значения на экран	7
7		Вызов метода getName() через указатель pObj для объекта класса cl_1, созданного объектом класса cl_3, и вывод его значения на экран	8
8		Вызов метода getName() через указатель pObj для объекта класса cl_1, созданного объектом класса cl_4, и вывод его значения на экран	9
9		Вызов метода getName() через указатель pObj для объекта класса cl_1, созданного объектом класса cl_5, и вывод его значения на экран	10
10		Вызов метода getName() через указатель pObj для объекта класса cl_2 и вывод его значения на экран	11
11		Вызов метода getName() через указатель pObj для объекта класса cl_3 и вывод его значения на экран	12
12		Вызов метода getName() через указатель pObj для объекта класса cl_4 и вывод его значения на экран	13
13		Вызов метода getName() через указатель pObj для объекта класса cl_5 и вывод его значения на экран	14
14		Вызов метода getName() через указатель pObj для объекта класса cl_6 и вывод его значения на экран	15
1		Вызов метода getName() через указатель pObj для объекта класса cl_7	16

№	Предикат	Действия	№ перехода
5		и вывод его значения на экран	
16		Вызов метода getName() через указатель pObj и вывод его значения на экран	17
17		Возвратить 0	∅

4 БЛОК-СХЕМЫ АЛГОРИТМОВ

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

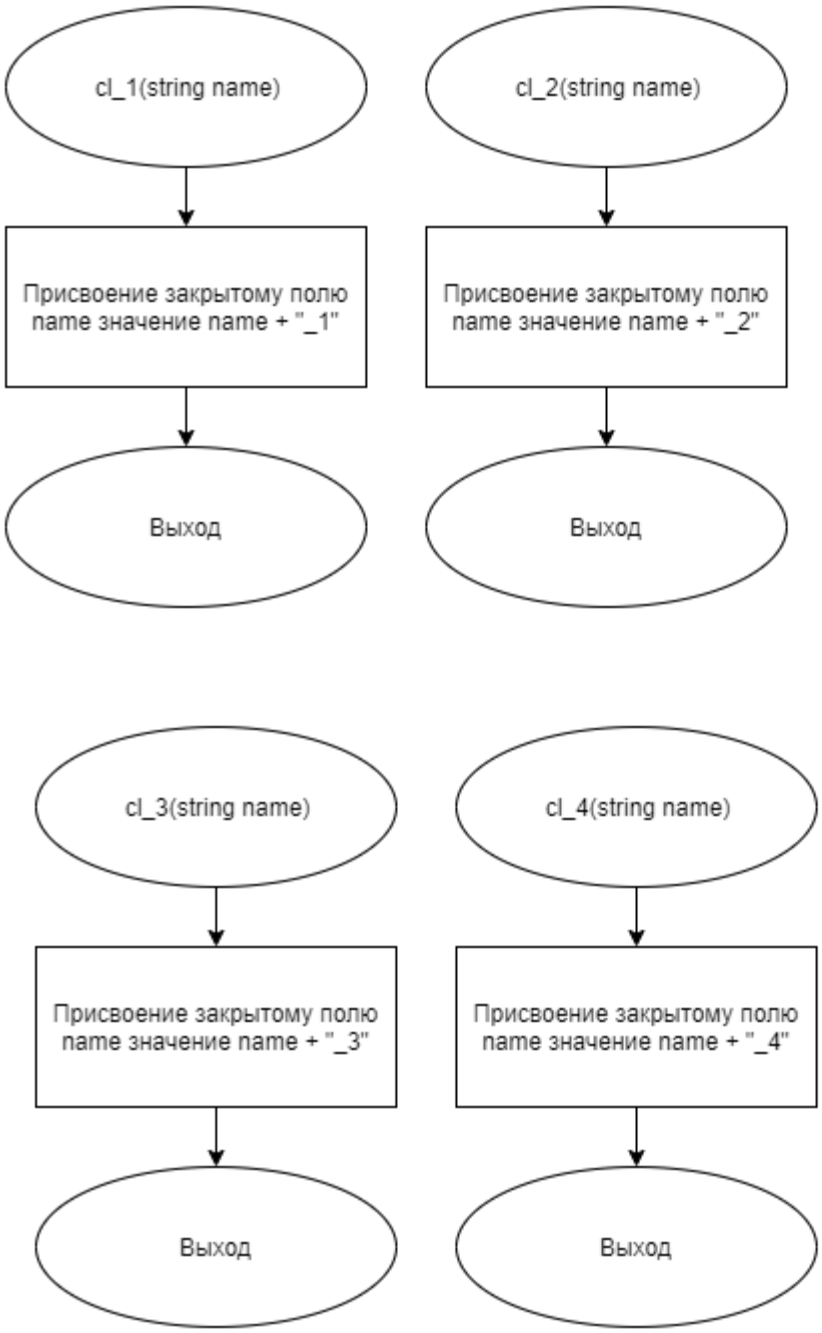

Рисунок 1 – Блок-схема алгоритма

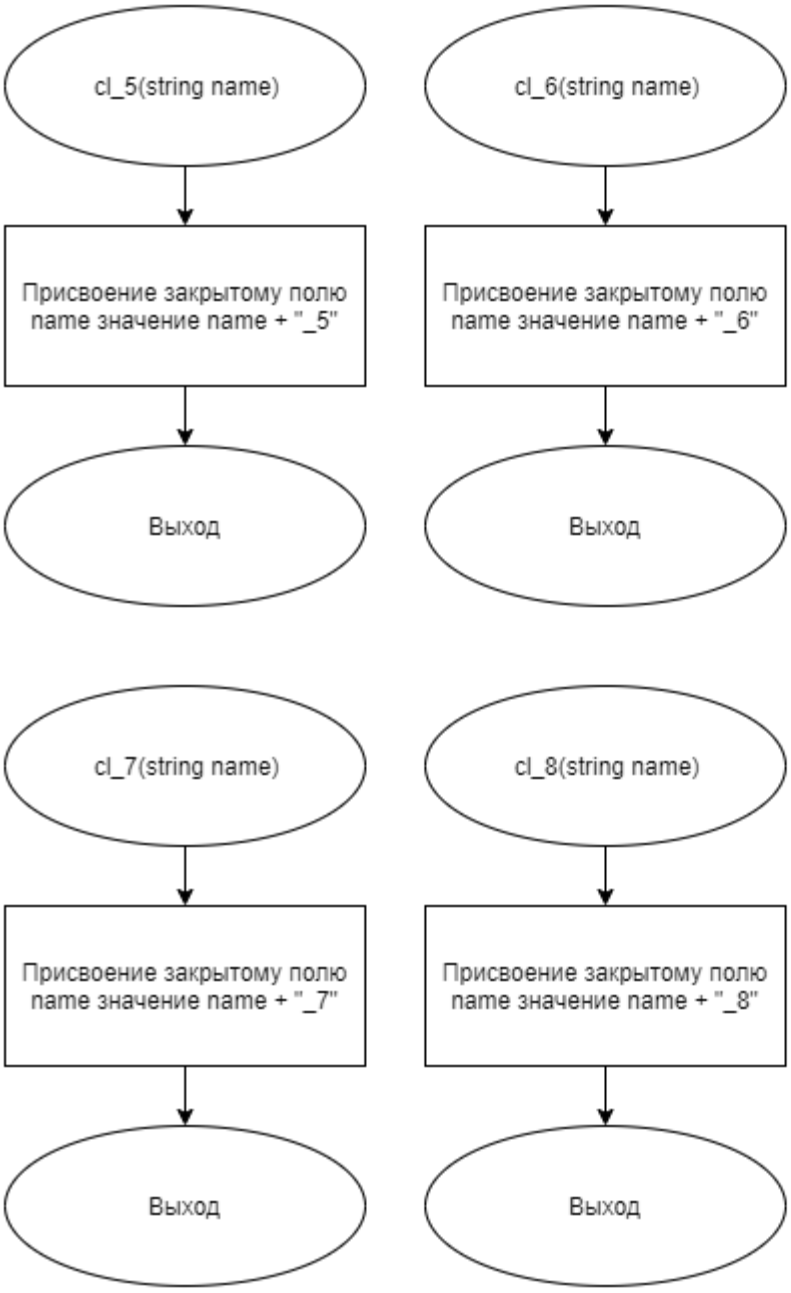

Рисунок 2 – Блок-схема алгоритма

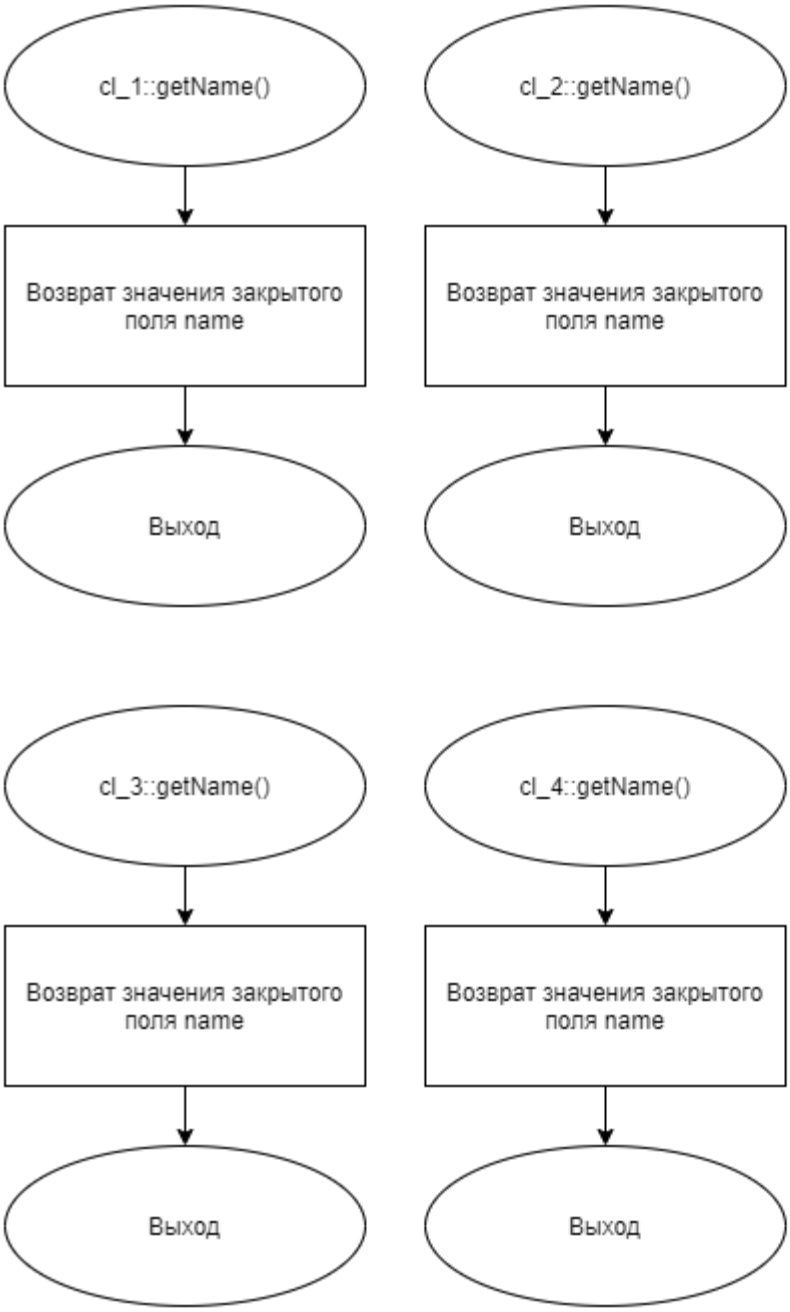

Рисунок 3 – Блок-схема алгоритма

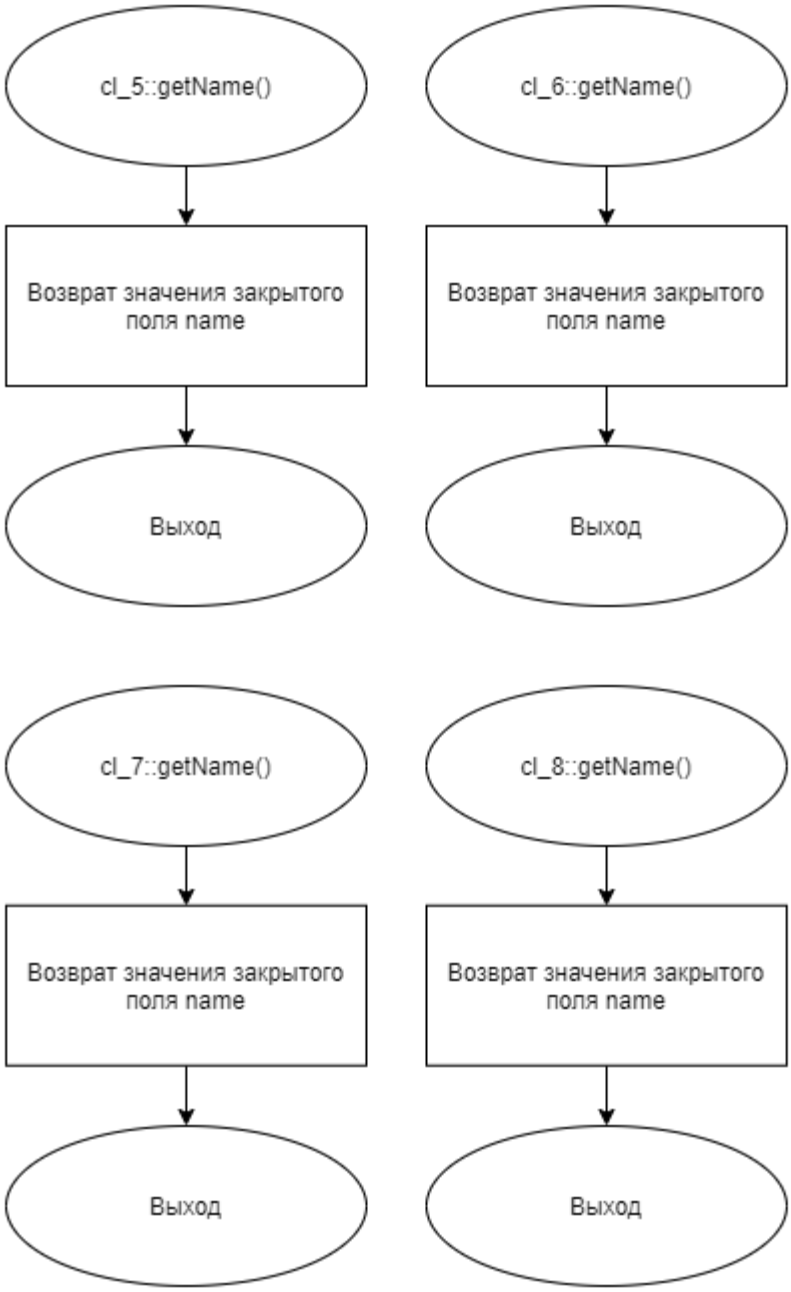

Рисунок 4 – Блок-схема алгоритма

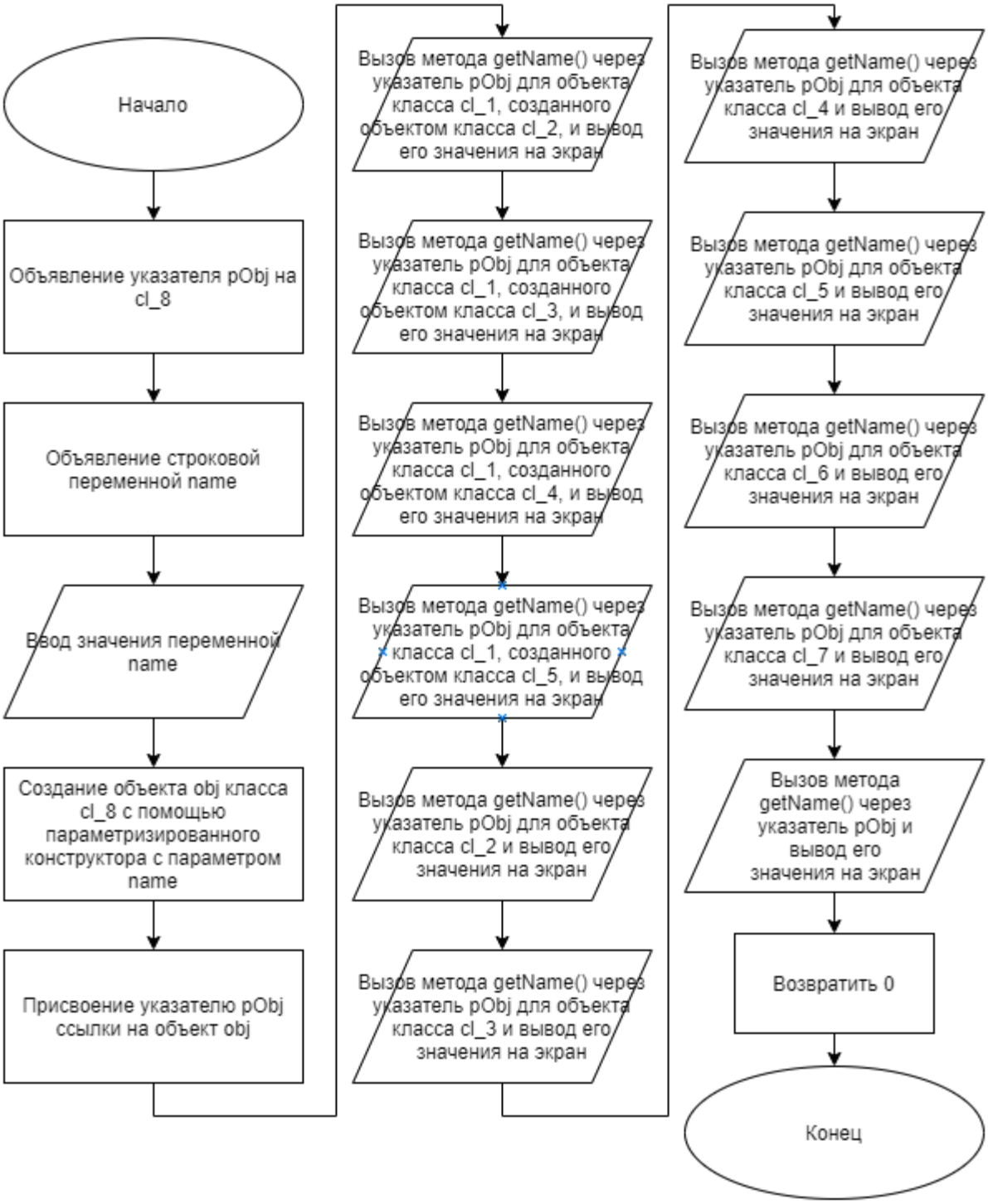

Рисунок 5 – Блок-схема алгоритма

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::getName() {
	return 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 getName();
};


#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::getName() {
	return 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 getName();
};
#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::getName() {
	return 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 getName();
};
#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::getName() {
	return 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 : virtual public cl_1 {
private:
	string name;
public:
	cl_4(string name);
	string getName();
};
#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::getName() {
	return 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 : virtual public cl_1 {
private:
	string name;
public:
	cl_5(string name);
	string getName();
};
#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::getName() {
	return 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 getName();
};

#endif
```

5.13 Файл cl_7.cpp

Листинг 13 – cl_7.cpp

```
#include "cl_7.h"
using namespace std;
cl_7::cl_7(string name):cl_1(name + "_7"), cl_4(name + "_7"), cl_5(name + "_7") {
	this->name = name + "_7";
}
string cl_7::getName() {
	return 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 getName();
};
```

```
#endif
```

5.15 Файл cl_8.cpp

Листинг 15 – cl_8.cpp

```
#include "cl_8.h"
cl_8::cl_8(string name):cl_7::cl_1(name + "_8"), cl_6(name + "_8"), cl_7(name +
"_8"){
	this->name = name + "_8";
}
string cl_8::getName() {
	return 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 getName();
};
#endif
```

5.17 Файл main.cpp

Листинг 17 – main.cpp

```
#include <iostream>
#include "cl_8.h"
using namespace std;

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

```
	cl_8 obj(name);
	pObj = &obj;
	cout << ((cl_1*)(cl_2*)pObj)->getName() << '\n';
	cout << ((cl_1*)(cl_3*)pObj)->getName() << '\n';
	cout << ((cl_1*)(cl_4*)pObj)->getName() << '\n';
	cout << ((cl_1*)(cl_5*)pObj)->getName() << '\n';
	cout << ((cl_2*)pObj)->getName() << '\n';
	cout << ((cl_3*)pObj)->getName() << '\n';
	cout << ((cl_4*)pObj)->getName() << '\n';
	cout << ((cl_5*)pObj)->getName() << '\n';
	cout << ((cl_6*)pObj)->getName() << '\n';
	cout << ((cl_7*)pObj)->getName() << '\n';
	cout << pObj->getName();
	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. Васильев А.Н. Объектно-ориентированное программирование на C++. Издательство: Наука и Техника. Санкт-Петербург, 2016г. 543 стр.

2. Шилдт Г. C++: базовый курс. 3-е изд. Пер. с англ.. — М.: Вильямс, 2017. — 624 с.

3. Методическое пособие для проведения практических заданий, контрольных и курсовых работ по дисциплине «Объектно-ориентированное программирование» [Электронный ресурс] – URL: https://mirea.aco-avrora.ru/student/files/methodichescoe_posobie_dlya_laboratorny h_rabot_3.pdf (дата обращения 05.05.2021).

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

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

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