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 стандартного потока ввода с клавиатуры cin;
- Объект стандартного потока вывода на экран cout.

Класс Class_1:

- свойства/поля:
    - поле Наименование объекта:
        - наименование — name;
        - тип — string;
        - модификатор доступа — private;
- функционал:
    - метод Class_1 — Конструктор параметризированный;
    - метод get — Возврат наименования объекта класса.

Класс Class_2:

- свойства/поля:
    - поле Наименование объекта:
        - наименование — name;
        - тип — string;
        - модификатор доступа — private;
- функционал:
    - метод Class_2 — Конструктор параметризированный;
    - метод get — Возврат наименования объекта класса.

Класс Class_3:

- свойства/поля:
    - поле Наименование объекта:
        - наименование — name;

        - тип — string;
        - модификатор доступа — private;
- функционал:
    - метод Class_3 — Конструктор параметризированный;
    - метод get — Возврат наименования объекта класса.

Класс Class_4:

- свойства/поля:
    - поле Наименование объекта:
        - наименование — name;
        - тип — string;
        - модификатор доступа — private;
- функционал:
    - метод Class_4 — Конструктор параметризированный;
    - метод get — Возврат наименования объекта класса.

Класс Class_5:

- свойства/поля:
    - поле Наименование объекта:
        - наименование — name;
        - тип — string;
        - модификатор доступа — private;
- функционал:
    - метод Class_5 — Конструктор параметризированный;
    - метод get — Возврат наименования объекта класса.

Класс Class_6:

- свойства/поля:
    - поле Наименование объекта:
        - наименование — name;

        - тип — string;
        - модификатор доступа — private;
- функционал:
    - метод Class_6 — Конструктор параметризированный;
    - метод get — Возврат наименования объекта класса.

Класс Class_7:

- свойства/поля:
    - поле Наименование объекта:
        - наименование — name;
        - тип — string;
        - модификатор доступа — private;
- функционал:
    - метод Class_7 — Конструктор параметризированный;
    - метод get — Возврат наименования объекта класса.

Класс Class_8:

- свойства/поля:
    - поле Наименование объекта:
        - наименование — name;
        - тип — string;
        - модификатор доступа — private;
- функционал:
    - метод Class_8 — Конструктор параметризированный;
    - метод get — Возврат наименования объекта класса.

Таблица 1 – Иерархия наследования классов

№	Имя класса	Классы-наследники	Модификатор доступа при наследовании	Описание	Номер
1	Class_1			Базовый класс	

№	Имя класса	Классы-наследники	Модификатор доступа при наследовании	Описание	Номер
		Class_2	public		2
		Class_3	public		3
		Class_4	virtual public		4
		Class_5	virtual public		5
2	Class_2			Производный класс	
		Class_6	public		6
3	Class_3			Производный класс	
		Class_6	public		6
4	Class_4			Производный класс	
		Class_7	public		7
5	Class_5			Производный класс	
		Class_7	public		7
6	Class_6			Производный класс	
		Class_8	public		8
7	Class_7			Производный класс	
		Class_8	public		8
8	Class_8			Производный 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 Class_1

Функционал: Конструктор параметризированный.

Параметры: Строка s для наименования объекта.

Алгоритм конструктора представлен в таблице 2.

Таблица 2 – Алгоритм конструктора класса Class_1

№	Предикат	Действия	№ перехода
1		Свойство name инициализируется значением параметра s + "_1"	∅

3.2 Алгоритм метода get класса Class_1

Функционал: Возврат наименования объекта класса.

Параметры: нет.

Возвращаемое значение: Строка с наименованием объекта.

Алгоритм метода представлен в таблице 3.

Таблица 3 – Алгоритм метода get класса Class_1

№	Предикат	Действия	№ перехода
1		Возврат скрытого свойства name	∅

3.3 Алгоритм конструктора класса Class_2

Функционал: Конструктор параметризированный.

Параметры: Строка s для наименования объекта.

Алгоритм конструктора представлен в таблице 4.

Таблица 4 – Алгоритм конструктора класса Class_2

№	Предикат	Действия	№ перехода
1		Свойство name инициализируется значением параметра s + "_2"	∅

3.4 Алгоритм метода get класса Class_2

Функционал: Возврат наименования объекта класса.

Параметры: нет.

Возвращаемое значение: Строка с наименованием объекта.

Алгоритм метода представлен в таблице 5.

Таблица 5 – Алгоритм метода get класса Class_2

№	Предикат	Действия	№ перехода
1		Возврат скрытого свойства name	∅

3.5 Алгоритм конструктора класса Class_3

Функционал: Конструктор параметризированный.

Параметры: Строка s для наименования объекта.

Алгоритм конструктора представлен в таблице 6.

Таблица 6 – Алгоритм конструктора класса Class_3

№	Предикат	Действия	№ перехода
1		Свойство name инициализируется значением параметра s + "_3"	∅

3.6 Алгоритм метода get класса Class_3

Функционал: Возврат наименования объекта класса.

Параметры: нет.

Возвращаемое значение: Строка с наименованием объекта.

Алгоритм метода представлен в таблице 7.

Таблица 7 – Алгоритм метода get класса Class_3

№	Предикат	Действия	№ перехода
1		Возврат скрытого свойства name	∅

3.7 Алгоритм конструктора класса Class_4

Функционал: Конструктор параметризированный.

Параметры: Строка s для наименования объекта.

Алгоритм конструктора представлен в таблице 8.

Таблица 8 – Алгоритм конструктора класса Class_4

№	Предикат	Действия	№ перехода
1		Свойство name инициализируется значением параметра s + "_4"	∅

3.8 Алгоритм метода get класса Class_4

Функционал: Возврат наименования объекта класса.

Параметры: нет.

Возвращаемое значение: Строка с наименованием объекта.

Алгоритм метода представлен в таблице 9.

Таблица 9 – Алгоритм метода get класса Class_4

№	Предикат	Действия	№ перехода
1		Возврат скрытого свойства name	∅

3.9 Алгоритм конструктора класса Class_5

Функционал: Конструктор параметризированный.

Параметры: Строка s для наименования объекта.

Алгоритм конструктора представлен в таблице 10.

Таблица 10 – Алгоритм конструктора класса Class_5

№	Предикат	Действия	№ перехода
1		Свойство name инициализируется значением параметра s + "_5"	∅

3.10 Алгоритм метода get класса Class_5

Функционал: Возврат наименования объекта класса.

Параметры: нет.

Возвращаемое значение: Строка с наименованием объекта.

Алгоритм метода представлен в таблице 11.

Таблица 11 – Алгоритм метода get класса Class_5

№	Предикат	Действия	№ перехода
1		Возврат скрытого свойства name	∅

3.11 Алгоритм конструктора класса Class_6

Функционал: Конструктор параметризированный.

Параметры: Строка s для наименования объекта.

Алгоритм конструктора представлен в таблице 12.

Таблица 12 – Алгоритм конструктора класса Class_6

№	Предикат	Действия	№ перехода
1		Свойство name инициализируется значением параметра s + "_6"	∅

3.12 Алгоритм метода get класса Class_6

Функционал: Возврат наименования объекта класса.

Параметры: нет.

Возвращаемое значение: Строка с наименованием объекта.

Алгоритм метода представлен в таблице 13.

Таблица 13 – Алгоритм метода get класса Class_6

№	Предикат	Действия	№ перехода
1		Возврат скрытого свойства name	∅

3.13 Алгоритм конструктора класса Class_7

Функционал: Конструктор параметризированный.

Параметры: Строка s для наименования объекта.

Алгоритм конструктора представлен в таблице 14.

Таблица 14 – Алгоритм конструктора класса Class_7

№	Предикат	Действия	№ перехода
1		Свойство name инициализируется значением параметра s + "_7"	∅

3.14 Алгоритм метода get класса Class_7

Функционал: Возврат наименования объекта класса.

Параметры: нет.

Возвращаемое значение: Строка с наименованием объекта.

Алгоритм метода представлен в таблице 15.

Таблица 15 – Алгоритм метода get класса Class_7

№	Предикат	Действия	№ перехода
1		Возврат скрытого свойства name	∅

3.15 Алгоритм конструктора класса Class_8

Функционал: Конструктор параметризированный.

Параметры: Строка s для наименования объекта.

Алгоритм конструктора представлен в таблице 16.

Таблица 16 – Алгоритм конструктора класса Class_8

№	Предикат	Действия	№ перехода
1		Свойство name инициализируется значением параметра s + "_8"	∅

3.16 Алгоритм метода get класса Class_8

Функционал: Возврат наименования объекта класса.

Параметры: нет.

Возвращаемое значение: Строка с наименованием объекта.

Алгоритм метода представлен в таблице 17.

Таблица 17 – Алгоритм метода get класса Class_8

№	Предикат	Действия	№ перехода
1		Возврат скрытого свойства name	∅

3.17 Алгоритм функции main

Функционал: Реализация множественного наследования.

Параметры: нет.

Возвращаемое значение: Целочисленное значение.

Алгоритм функции представлен в таблице 18.

Таблица 18 – Алгоритм функции main

№	Предикат	Действия	№ перехода
1		Объявление указателя obj на объект класса Class_8	2
2		Объявление строковой переменной s	3
3		Ввод значения s	4
4		Создание объекта класса Class_8 с параметром s и присвоения адреса этого объекта указателю obj	5
5		Приведение указателя obj к типу Class_2, затем приведение результата к типу Class_1, вывод значения результата метода get по указателю obj	6
6		Приведение указателя obj к типу Class_3, затем приведение результата к типу Class_1, вывод значения результата метода get по указателю obj	7
7		Приведение указателя obj к типу Class_4, затем приведение результата к типу Class_1, вывод значения результата метода get по указателю obj	8
8		Приведение указателя obj к типу Class_5, затем приведение результата к типу Class_1, вывод значения результата метода get по указателю obj	9
9		Приведение указателя obj к типу Class_2, вывод значения результата метода get по указателю obj	10
10		Приведение указателя obj к типу Class_3, вывод значения результата метода get по указателю obj	11
11		Приведение указателя obj к типу Class_4, вывод значения результата метода get по указателю obj	12
12		Приведение указателя obj к типу Class_5, вывод значения результата метода get по указателю obj	13

№	Предикат	Действия	№ перехода
13		Приведение указателя obj к типу Class_6, вывод значения результата метода get по указателю obj	14
14		Приведение указателя obj к типу Class_7, вывод значения результата метода get по указателю obj	15
15		Вывод значения результата метода get по указателю obj	∅

4 БЛОК-СХЕМЫ АЛГОРИТМОВ

Представим описание алгоритмов в графическом виде на рисунках 1-4.

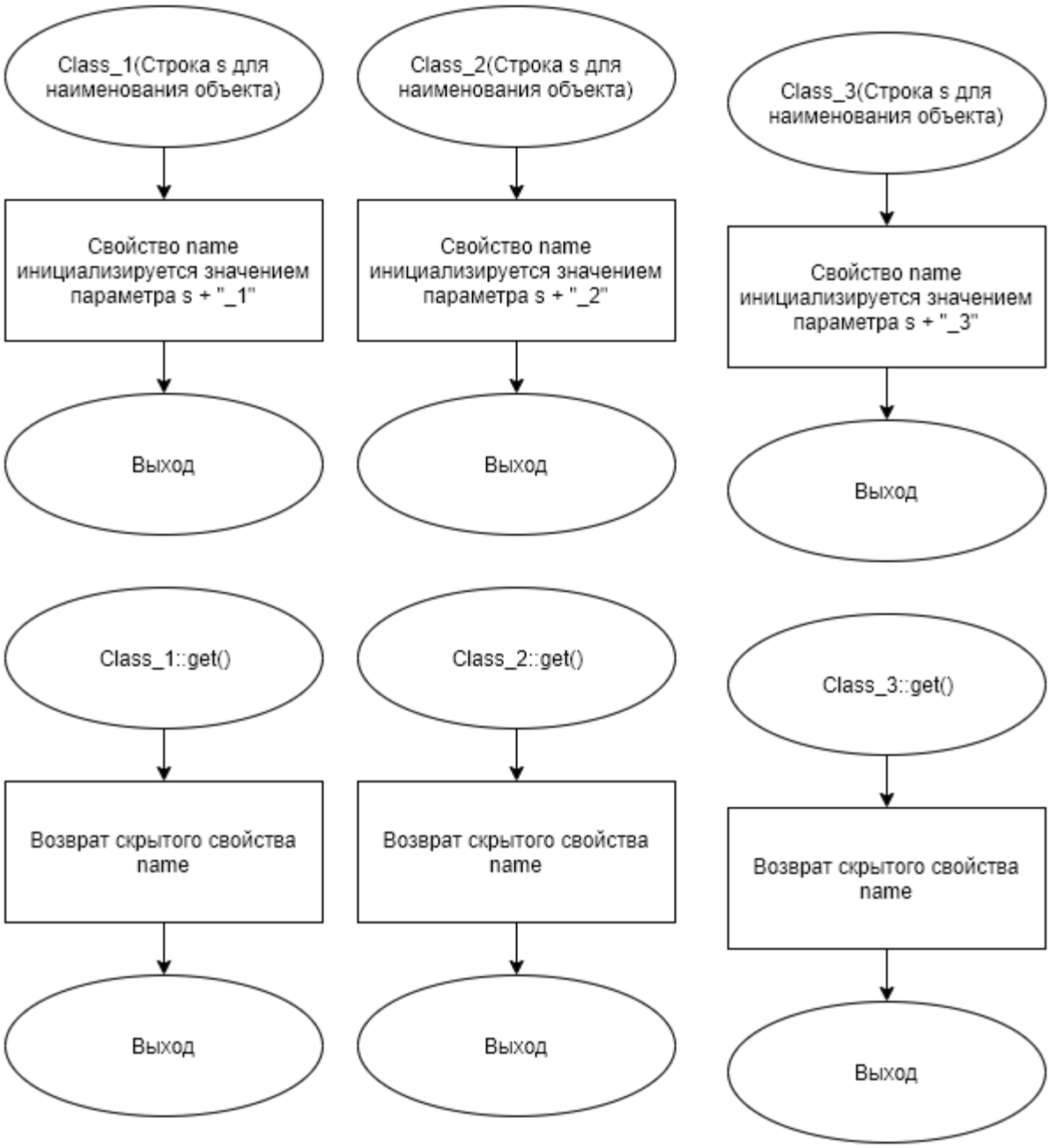

Рисунок 1 – Блок-схема алгоритма

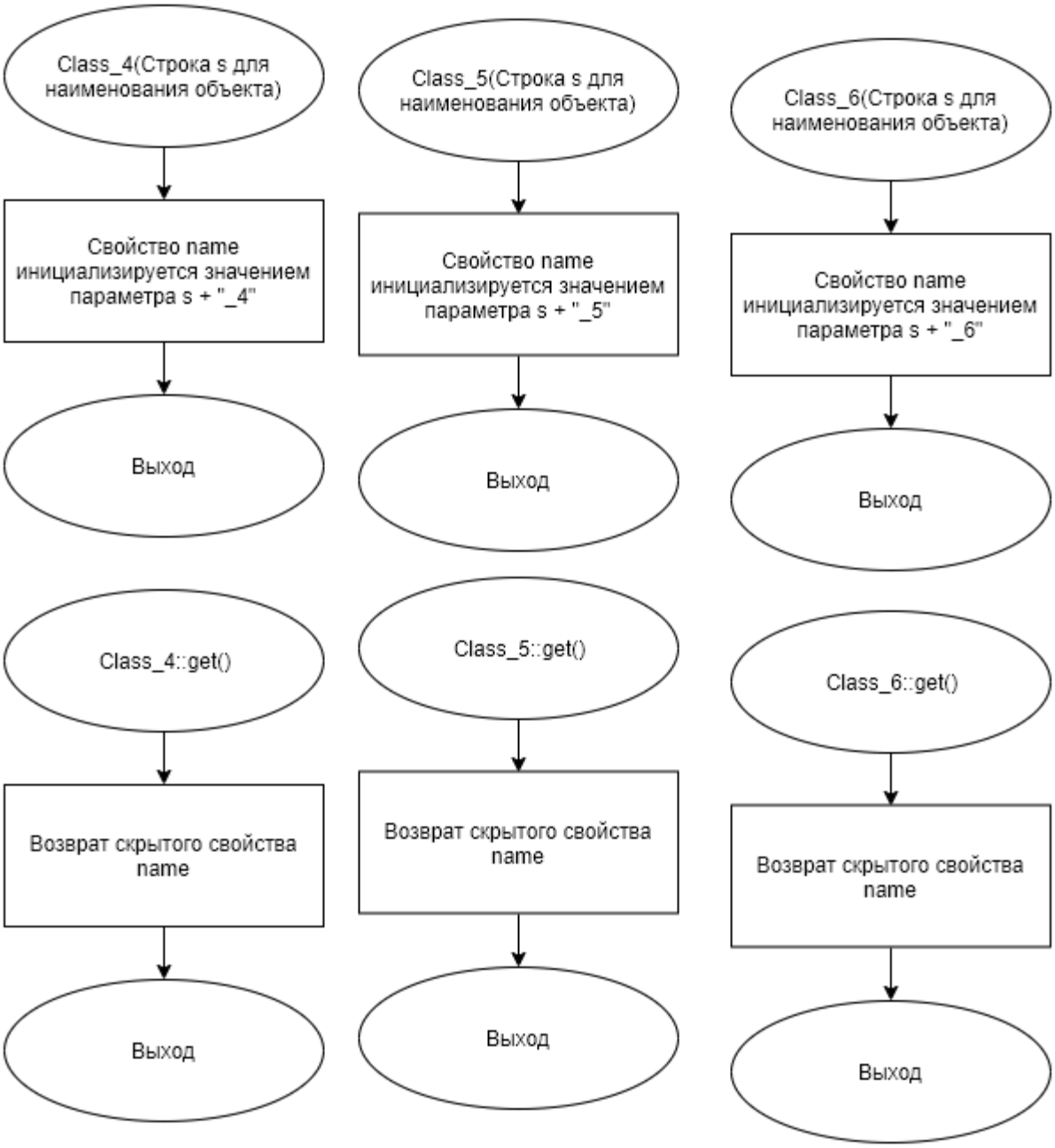

Рисунок 2 – Блок-схема алгоритма

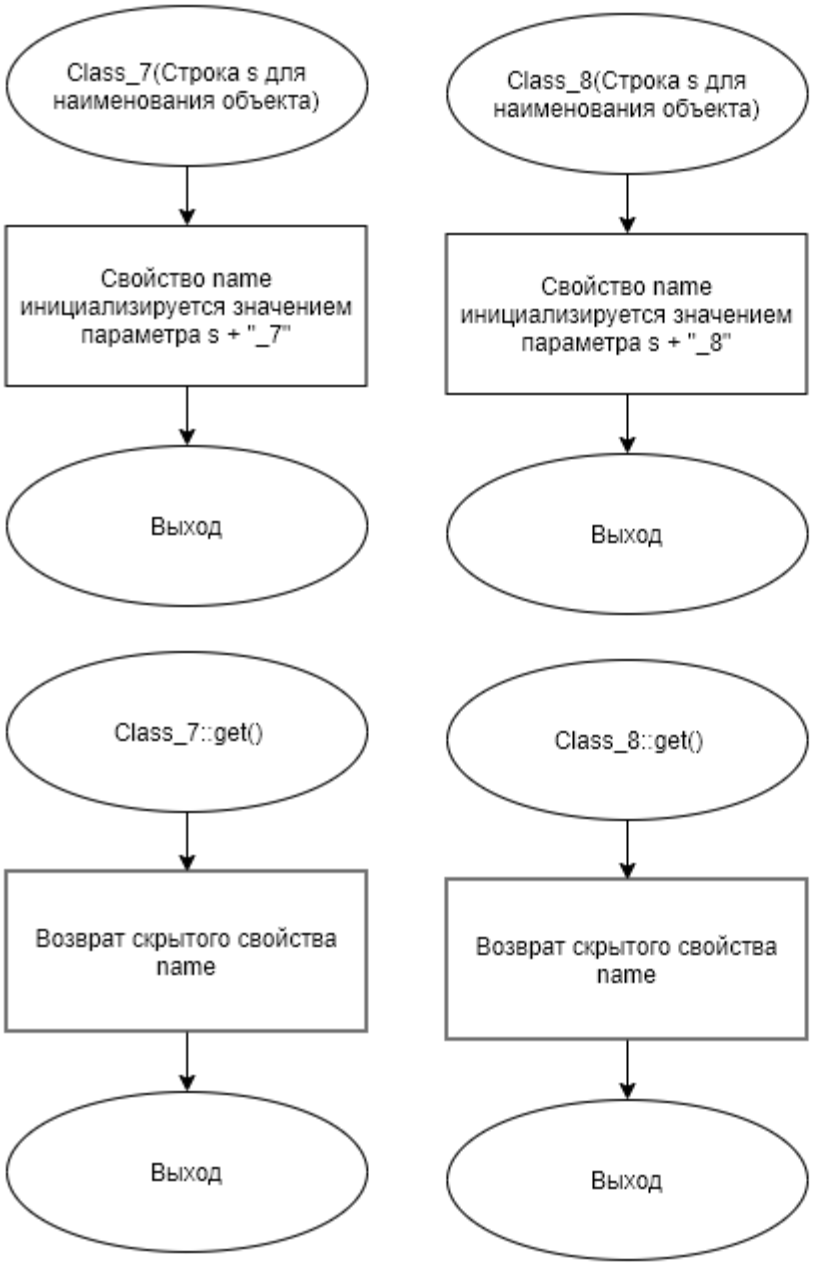

Рисунок 3 – Блок-схема алгоритма

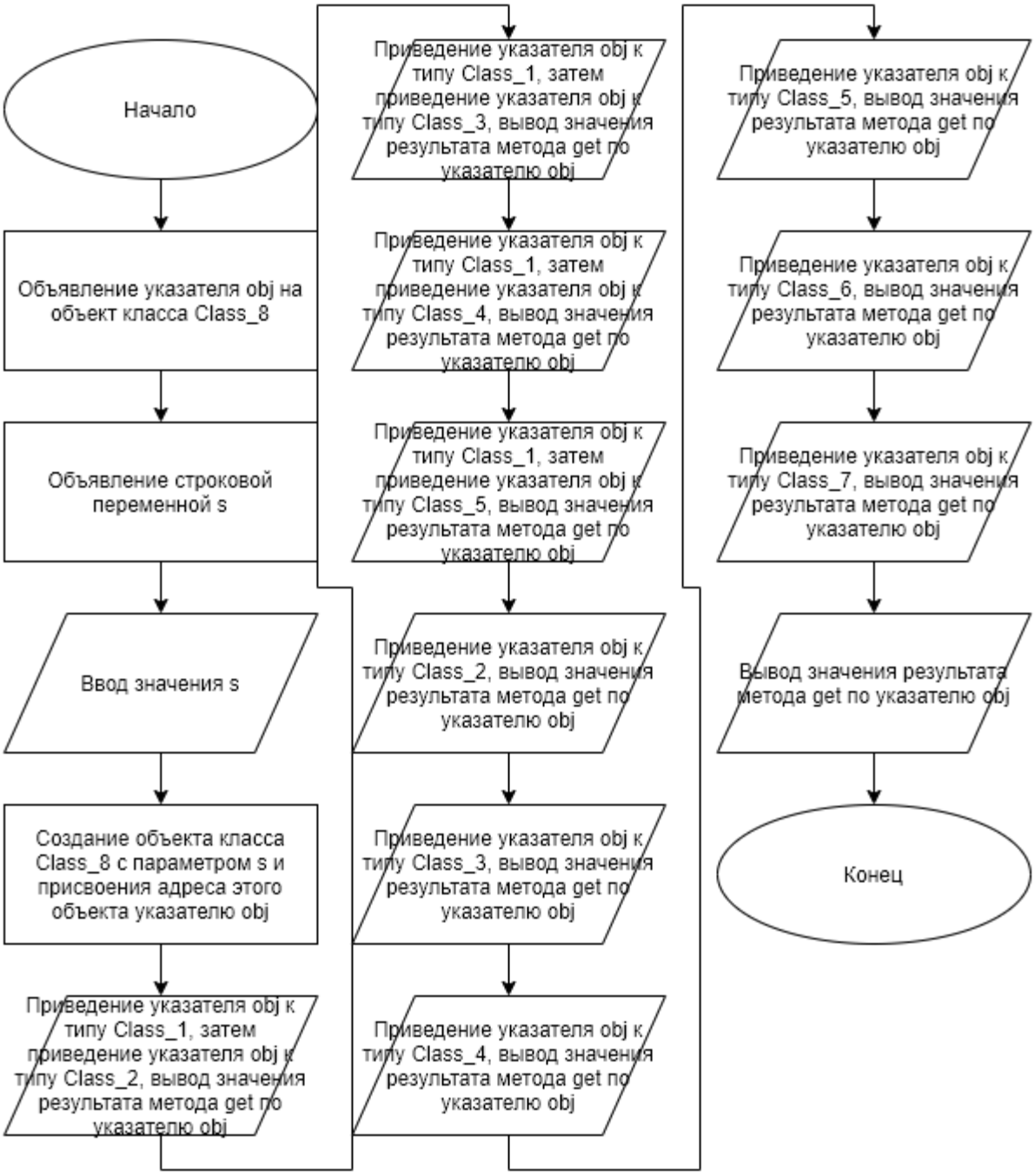

Рисунок 4 – Блок-схема алгоритма

5 КОД ПРОГРАММЫ

Программная реализация алгоритмов для решения задачи представлена ниже.

5.1 Файл Class_1.cpp

Листинг 1 – Class_1.cpp

```
#include "Class_1.h"

Class_1::Class_1(string s)
{
   name = s + "_1";
}

string Class_1::get()
{
   return name;
}
```

5.2 Файл Class_1.h

Листинг 2 – Class_1.h

```
#ifndef __CLASS_1__H
#define __CLASS_1__H
#include <iostream>
#include <string>

using namespace std;

class Class_1
{
private:
   string name;

public:
   Class_1(string s);
   string get();
};
```

```
#endif
```

5.3 Файл Class_2.cpp

Листинг 3 – Class_2.cpp

```
#include "Class_2.h"

Class_2::Class_2(string s):Class_1(s + "_2")
{
   name = s + "_2";
}

string Class_2::get()
{
   return name;
}
```

5.4 Файл Class_2.h

Листинг 4 – Class_2.h

```
#ifndef __CLASS_2__H
#define __CLASS_2__H
#include <iostream>
#include <string>
#include "Class_1.h"

using namespace std;

class Class_2: public Class_1
{
private:
   string name;

public:
   Class_2(string s);
   string get();
};


#endif
```

5.5 Файл Class_3.cpp

Листинг 5 – Class_3.cpp

```
#include "Class_3.h"

Class_3::Class_3(string s):Class_1(s + "_3")
{
   name = s + "_3";
}

string Class_3::get()
{
   return name;
}
```

5.6 Файл Class_3.h

Листинг 6 – Class_3.h

```
#ifndef __CLASS_3__H
#define __CLASS_3__H
#include <iostream>
#include <string>
#include "Class_1.h"

using namespace std;

class Class_3: public Class_1
{
private:
   string name;

public:
   Class_3(string s);
   string get();
};


#endif
```

5.7 Файл Class_4.cpp

Листинг 7 – Class_4.cpp

```
#include "Class_4.h"

Class_4::Class_4(string s):Class_1(s + "_4")
{
   name = s + "_4";
}

string Class_4::get()
{
   return name;
}
```

5.8 Файл Class_4.h

Листинг 8 – Class_4.h

```
#ifndef __CLASS_4__H
#define __CLASS_4__H
#include <iostream>
#include <string>
#include "Class_1.h"

using namespace std;

class Class_4: virtual public Class_1
{
private:
   string name;

public:
   Class_4(string s);
   string get();
};


#endif
```

5.9 Файл Class_5.cpp

Листинг 9 – Class_5.cpp

```
#include "Class_5.h"

Class_5::Class_5(string s):Class_1(s + "_5")
{
   name = s + "_5";
}

string Class_5::get()
{
   return name;
}
```

5.10 Файл Class_5.h

Листинг 10 – Class_5.h

```
#ifndef __CLASS_5__H
#define __CLASS_5__H
#include <iostream>
#include <string>
#include "Class_1.h"

using namespace std;

class Class_5: virtual public Class_1
{
private:
   string name;

public:
   Class_5(string s);
   string get();
};


#endif
```

5.11 Файл Class_6.cpp

Листинг 11 – Class_6.cpp

```
#include "Class_6.h"

Class_6::Class_6(string s):Class_2(s + "_6"), Class_3(s + "_6")
{
   name = s + "_6";
}

string Class_6::get()
{
   return name;
}
```

5.12 Файл Class_6.h

Листинг 12 – Class_6.h

```
#ifndef __CLASS_6__H
#define __CLASS_6__H
#include <iostream>
#include <string>
#include "Class_2.h"
#include "Class_3.h"

using namespace std;

class Class_6: public Class_2, public Class_3
{
private:
   string name;

public:
   Class_6(string s);
   string get();
};


#endif
```

5.13 Файл Class_7.cpp

Листинг 13 – Class_7.cpp

```
#include "Class_7.h"

Class_7::Class_7(string s):Class_1(s + "_7"), Class_4(s + "_7"), Class_5(s +
"_7")
{
   name = s + "_7";
}

string Class_7::get()
{
   return name;
}
```

5.14 Файл Class_7.h

Листинг 14 – Class_7.h

```
#ifndef __CLASS_7__H
#define __CLASS_7__H
#include <iostream>
#include <string>
#include "Class_4.h"
#include "Class_5.h"

using namespace std;

class Class_7: public Class_4, public Class_5
{
private:
   string name;

public:
   Class_7(string s);
   string get();
};


#endif
```

5.15 Файл Class_8.cpp

Листинг 15 – Class_8.cpp

```
#include "Class_8.h"

Class_8::Class_8(string s):Class_1(s + "_8"), Class_6(s + "_8"), Class_7(s +
"_8")
{
   name = s + "_8";
}

string Class_8::get()
{
   return name;
}
```

5.16 Файл Class_8.h

Листинг 16 – Class_8.h

```
#ifndef __CLASS_8__H
#define __CLASS_8__H
#include <iostream>
#include <string>
#include "Class_6.h"
#include "Class_7.h"

using namespace std;

class Class_8: public Class_6, public Class_7
{
private:
   string name;

public:
   Class_8(string s);
   string get();
};


#endif
```

5.17 Файл main.cpp

Листинг 17 – main.cpp

```
#include <stdlib.h>
#include <stdio.h>
#include "Class_8.h"

int main()
{
   Class_8* obj;
   string s;
   cin >> s;
   obj = new Class_8(s);
   cout << ((Class_1*)((Class_2*)(Class_6*)obj)) -> get() << endl;
   cout << ((Class_1*)((Class_3*)(Class_6*)obj)) -> get() << endl;
   cout << ((Class_1*)((Class_4*)(Class_7*)obj)) -> get() << endl;
   cout << ((Class_1*)((Class_5*)(Class_7*)obj)) -> get() << endl;
   cout << ((Class_2*)(Class_6*)obj) -> get() << endl;
   cout << ((Class_3*)(Class_6*)obj) -> get() << endl;
   cout << ((Class_4*)(Class_7*)obj) -> get() << endl;
   cout << ((Class_5*)(Class_7*)obj) -> get() << endl;
   cout << ((Class_6*)obj) -> get() << endl;
   cout << ((Class_7*)obj) -> get() << endl;
   cout << obj -> get();
}
```

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
qq	qq_8_6_2_1 qq_8_6_3_1 qq_8_1 qq_8_1 qq_8_6_2 qq_8_6_3 qq_8_7_4 qq_8_7_5 qq_8_6 qq_8_7 qq_8	qq_8_6_2_1 qq_8_6_3_1 qq_8_1 qq_8_1 qq_8_6_2 qq_8_6_3 qq_8_7_4 qq_8_7_5 qq_8_6 qq_8_7 qq_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).