Здесь будет титульник, листай ниже

СОДЕРЖАНИЕ

1 ПОСТАНОВКА ЗАДАЧИ

Разработать систему, которая демонстрирует возможность:

- конструирования производного объекта на базе нескольких поколений родительских объектов;
- переконфигурации расположения родительских объектов на определенном уровне иерархии;
- однозначного обращения (использования, доступа) к элементам производного и исходных объектов.

Спроектировать 8 разных объектов. Перенумеровать классы их принадлежности от 1 до 8. Каждый объект имеет свойство строкового типа в закрытом доступе. Это свойство хранит наименование объекта. Наименование объекта определяется в момент создания объекта посредством значения параметра строкового типа конструктора и номера принадлежности классу согласно шаблону:

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

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

Объекты 2, 3, 4 и 5 классов являются производными (содержат в своем составе) объект первого класса. Объект шестого класса содержит в своем составе объекты второго и третьего класса. Объект седьмого класса содержит в своем составе объекты четвертого и пятого класса. Объект восьмого класса содержит в своем составе объекты шестого и седьмого класса.

При создании объекта со второй по восьмой класс, до вызова их параметризированного конструктора, вызвать параметризированный конструктор или конструкторы входящих (родительских) объектов. При вызове в конструктор родительского объекта в качестве параметра передать выражение:

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

Например, для конструктора объекта второго класса

```
cl_2 :: cl_2 ( string s_name ) : cl_1 ( s_name + "_2" )
```

Алгоритм конструирования и отработки системы:

1. Объявляется один указатель на объект класса x.
2. Объявляется переменная строкового типа.
3. Вводится значение строковой переменной. Введенное значение является идентификатором.
4. Создается объект класса 8 посредством параметризированного конструктора, в качестве аргумента передается строковая переменная.
5. Адрес созданного объекта присваивается указателю на объект класса x.
6. Используя только указатель на объект класса x выводится имена всех объектов в составе объекта класса 8 и имя самого объекта класса 8. Вывод выполнить построчно, упорядоченная согласно возрастанию номеров классов. Наименования объектов первого класса выводится последовательно для производных объектов 2, 3. 4 и 5 класса.

Конструктивно выполнить построение так, чтобы всего объектов было 10. При сдаче задачи посредством вложенных прямоугольников нарисовать конструкцию объекта 8 класса и дать пояснения.

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 obj класса cl_8 предназначен для демонстрации работы программы;
- функция main для определения входной точки программы;
- заголовочные файлы;
- классы;
- указатель;
- виртуальное наследование.

Класс cl_1:

- свойства/поля:
  - поле хранит имя объекта:
    - наименование — m_objectName;
    - тип — string;
    - модификатор доступа — private;
- функционал:
  - метод cl_1 — инициализирует m_objectName;
  - метод getName — возвращает значение приватного поля m_objectName, которое содержит имя объекта.

Класс cl_2:

- свойства/поля:
  - поле хранит имя объекта:
    - наименование — m_objectName;
    - тип — string;
    - модификатор доступа — private;
- функционал:

- метод cl_2 — инициализирует m_objectName;
- метод getName — возвращает значение приватного поля m_objectName, которое содержит имя объекта.

Класс cl_3:

- свойства/поля:
  - поле хранит имя объекта:
    - наименование — m_objectName;
    - тип — string;
    - модификатор доступа — private;
- функционал:
  - метод cl_3 — инициализирует m_objectName;
  - метод getName — возвращает значение приватного поля m_objectName, которое содержит имя объекта.

Класс cl_4:

- свойства/поля:
  - поле хранит имя объекта:
    - наименование — m_objectName;
    - тип — string;
    - модификатор доступа — private;
- функционал:
  - метод cl_4 — инициализирует m_objectName;
  - метод getName — возвращает значение приватного поля m_objectName, которое содержит имя объекта.

Класс cl_5:

- свойства/поля:
  - поле хранит имя объекта:
    - наименование — m_objectName;

        - тип — string;
        - модификатор доступа — private;
- функционал:
    - метод cl_5 — инициализирует m_objectName;
    - метод getName — возвращает значение приватного поля m_objectName, которое содержит имя объекта.

Класс cl_6:

- свойства/поля:
    - поле хранит имя объекта:
        - наименование — m_objectName;
        - тип — string;
        - модификатор доступа — private;
- функционал:
    - метод cl_6 — инициализирует m_objectName;
    - метод getName — возвращает значение приватного поля m_objectName, которое содержит имя объекта.

Класс cl_7:

- свойства/поля:
    - поле хранит имя объекта:
        - наименование — m_objectName;
        - тип — string;
        - модификатор доступа — private;
- функционал:
    - метод cl_7 — инициализирует m_objectName;
    - метод getName — возвращает значение приватного поля m_objectName, которое содержит имя объекта.

Класс cl_8:

- свойства/поля:
    - поле хранит имя объекта:
        - наименование — m_objectName;
        - тип — string;
        - модификатор доступа — private;
- функционал:
    - метод cl_8 — инициализирует m_objectName;
    - метод getName — возвращает значение приватного поля m_objectName, которое содержит имя объекта.

Таблица 1 – Иерархия наследования классов

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

№	Имя класса	Классы-наследники	Модификатор доступа при наследовании	Описание	Номер
7	cl_7				
		cl_8	public		8
8	cl_8				

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: Инициализирует m_objectName.

Параметры: std::string name.

Алгоритм конструктора представлен в таблице 2.

Таблица 2 – Алгоритм конструктора класса cl_1

№	Предикат	Действия	№ перехода
1		Инициализация m_objectName = name + "_1"	∅

3.2 Алгоритм конструктора класса cl_2

Функционал: Инициализирует m_objectName.

Параметры: std::string name.

Алгоритм конструктора представлен в таблице 3.

Таблица 3 – Алгоритм конструктора класса cl_2

№	Предикат	Действия	№ перехода
1		Инициализация m_objectName = name + "_2"	∅

3.3 Алгоритм конструктора класса cl_3

Функционал: Инициализирует m_objectName.

Параметры: std::string name.

Алгоритм конструктора представлен в таблице 4.

Таблица 4 – Алгоритм конструктора класса cl_3

№	Предикат	Действия	№ перехода
1		Инициализация m_objectName = name + "_3"	∅

3.4 Алгоритм конструктора класса cl_4

Функционал: Инициализирует m_objectName.

Параметры: std::string name.

Алгоритм конструктора представлен в таблице 5.

Таблица 5 – Алгоритм конструктора класса cl_4

№	Предикат	Действия	№ перехода
1		Инициализация m_objectName = name + "_4"	∅

3.5 Алгоритм конструктора класса cl_5

Функционал: Инициализирует m_objectName.

Параметры: std::string name.

Алгоритм конструктора представлен в таблице 6.

Таблица 6 – Алгоритм конструктора класса cl_5

№	Предикат	Действия	№ перехода
1		Инициализация m_objectName = name + "_5"	∅

3.6 Алгоритм конструктора класса cl_6

Функционал: Инициализирует m_objectName.

Параметры: std::string name.

Алгоритм конструктора представлен в таблице 7.

Таблица 7 – Алгоритм конструктора класса cl_6

№	Предикат	Действия	№ перехода
1		Инициализация m_objectName = name + "_6"	∅

3.7 Алгоритм конструктора класса cl_7

Функционал: Инициализирует m_objectName.

Параметры: std::string name.

Алгоритм конструктора представлен в таблице 8.

Таблица 8 – Алгоритм конструктора класса cl_7

№	Предикат	Действия	№ перехода
1		Инициализация m_objectName = name + "_7"	∅

3.8 Алгоритм конструктора класса cl_8

Функционал: Инициализирует m_objectName.

Параметры: std::string name.

Алгоритм конструктора представлен в таблице 9.

Таблица 9 – Алгоритм конструктора класса cl_8

№	Предикат	Действия	№ перехода
1		Инициализация m_objectName = name + "_8"	∅

3.9 Алгоритм функции main

Функционал: Входная точка программы.

Параметры: нет.

Возвращаемое значение: int.

Алгоритм функции представлен в таблице 10.

Таблица 10 – Алгоритм функции main

№	Предикат	Действия	№ перехода
1		Объявление указателя на объект obj класса cl_8	2
2		Объявление строковой переменной name	3
3		Ввод name	4
4		Инициализация указателя на объект obj класса cl_8 с помощью оператора new	5
5		Вывод ((cl_1*)((cl_2*)((cl_6*)(obj))))->PrintName()	6
6		Вывод ((cl_1*)((cl_3*)((cl_6*)(obj))))->PrintName()	7
7		Вывод ((cl_1*)((cl_4*)(obj)))->PrintName()	8
8		Вывод ((cl_1*)((cl_5*)(obj)))->PrintName()	9
9		Вывод ((cl_2*)((cl_6*)(obj)))->PrintName()	10
10		Вывод ((cl_3*)((cl_6*)(obj)))->PrintName()	11
11		Вывод ((cl_4*)((cl_7*)(obj)))->PrintName()	12
12		Вывод ((cl_5*)((cl_7*)(obj)))->PrintName()	13
13		Вывод ((cl_6*)(obj))->PrintName()	14
14		Вывод ((cl_7*)(obj))->PrintName()	15
15		Вывод obj->PrintName()	16
16		Удаление объекта obj с помощью оператора delete с освобождение памяти указателя	∅

3.10 Алгоритм метода getName класса cl_1

Функционал: Возвращает значение приватного поля m_objectName, которое содержит имя объекта.

Параметры: нет.

Возвращаемое значение: std::string.

Алгоритм метода представлен в таблице 11.

Таблица 11 – Алгоритм метода getName класса cl_1

№	Предикат	Действия	№ перехода
1		Возврат приватного поля строкового типа m_objectName	∅

3.11 Алгоритм метода getName класса cl_2

Функционал: Возвращает значение приватного поля m_objectName, которое содержит имя объекта.

Параметры: нет.

Возвращаемое значение: std::string.

Алгоритм метода представлен в таблице 12.

Таблица 12 – Алгоритм метода getName класса cl_2

№	Предикат	Действия	№ перехода
1		Возврат приватного поля строкового типа m_objectName	∅

3.12 Алгоритм метода getName класса cl_3

Функционал: Возвращает значение приватного поля m_objectName, которое содержит имя объекта.

Параметры: нет.

Возвращаемое значение: std::string.

Алгоритм метода представлен в таблице 13.

Таблица 13 – Алгоритм метода getName класса cl_3

№	Предикат	Действия	№ перехода
1		Возврат приватного поля строкового типа m_objectName	∅

3.13 Алгоритм метода getName класса cl_4

Функционал: Возвращает значение приватного поля m_objectName, которое содержит имя объекта.

Параметры: нет.

Возвращаемое значение: std::string.

Алгоритм метода представлен в таблице 14.

Таблица 14 – Алгоритм метода getName класса cl_4

№	Предикат	Действия	№ перехода
1		Возврат приватного поля строкового типа m_objectName	∅

3.14 Алгоритм метода getName класса cl_5

Функционал: Возвращает значение приватного поля m_objectName, которое содержит имя объекта.

Параметры: нет.

Возвращаемое значение: std::string.

Алгоритм метода представлен в таблице 15.

Таблица 15 – Алгоритм метода getName класса cl_5

№	Предикат	Действия	№ перехода
1		Возврат приватного поля строкового типа m_objectName	∅

3.15 Алгоритм метода getName класса cl_6

Функционал: Возвращает значение приватного поля m_objectName, которое содержит имя объекта.

Параметры: нет.

Возвращаемое значение: std::string.

Алгоритм метода представлен в таблице 16.

Таблица 16 – Алгоритм метода getName класса cl_6

№	Предикат	Действия	№ перехода
1		Возврат приватного поля строкового типа m_objectName	∅

3.16 Алгоритм метода getName класса cl_7

Функционал: Возвращает значение приватного поля m_objectName, которое содержит имя объекта.

Параметры: нет.

Возвращаемое значение: std::string.

Алгоритм метода представлен в таблице 17.

Таблица 17 – Алгоритм метода getName класса cl_7

№	Предикат	Действия	№ перехода
1		Возврат приватного поля строкового типа m_objectName	∅

3.17 Алгоритм метода getName класса cl_8

Функционал: Возвращает значение приватного поля m_objectName, которое содержит имя объекта.

Параметры: нет.

Возвращаемое значение: std::string.

Алгоритм метода представлен в таблице 18.

Таблица 18 – Алгоритм метода getName класса cl_8

№	Предикат	Действия	№ перехода
1		Возврат приватного поля строкового типа m_objectName	∅

4 БЛОК-СХЕМЫ АЛГОРИТМОВ

Представим описание алгоритмов в графическом виде на рисунках 1-6.

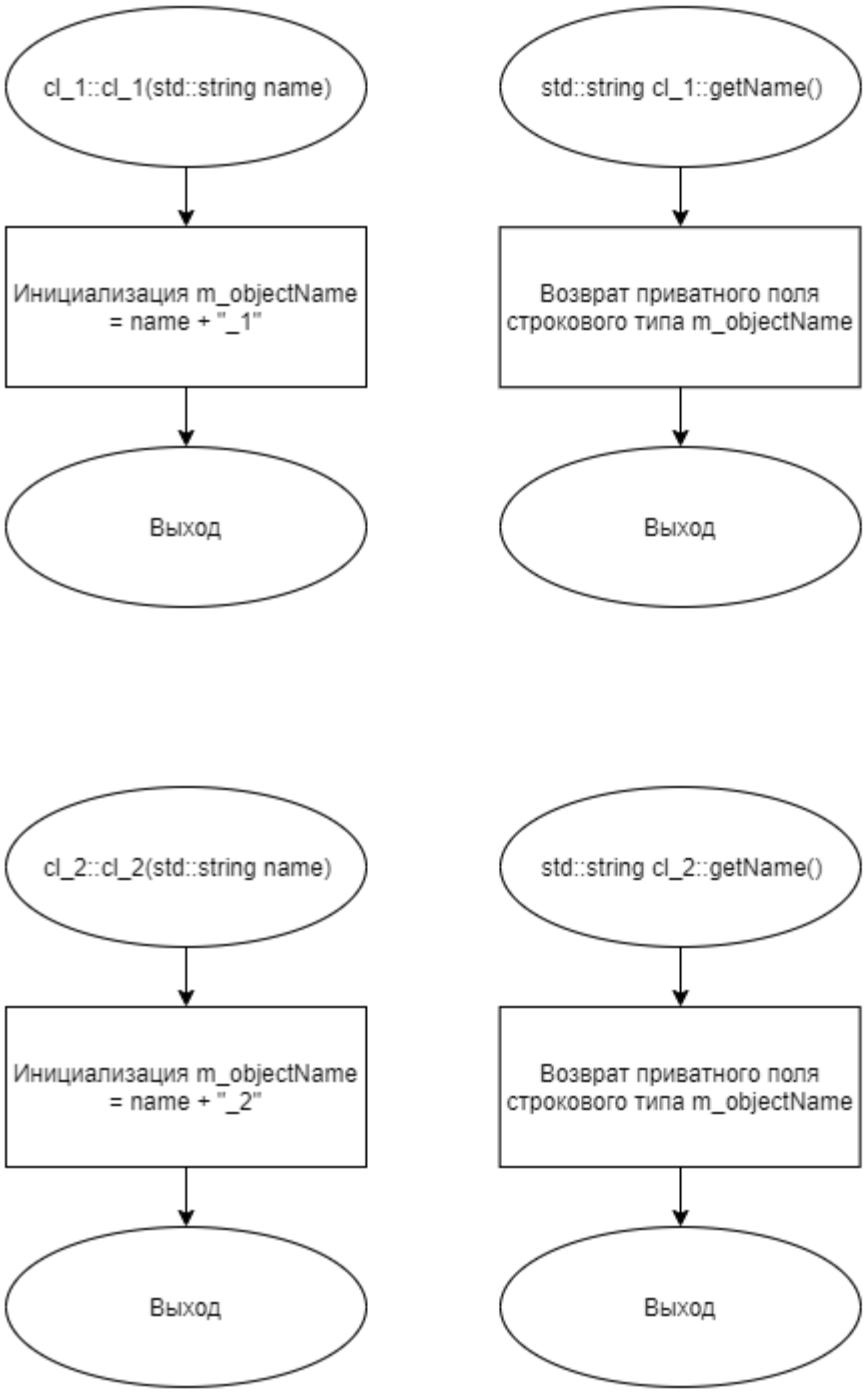

Рисунок 1 – Блок-схема алгоритма

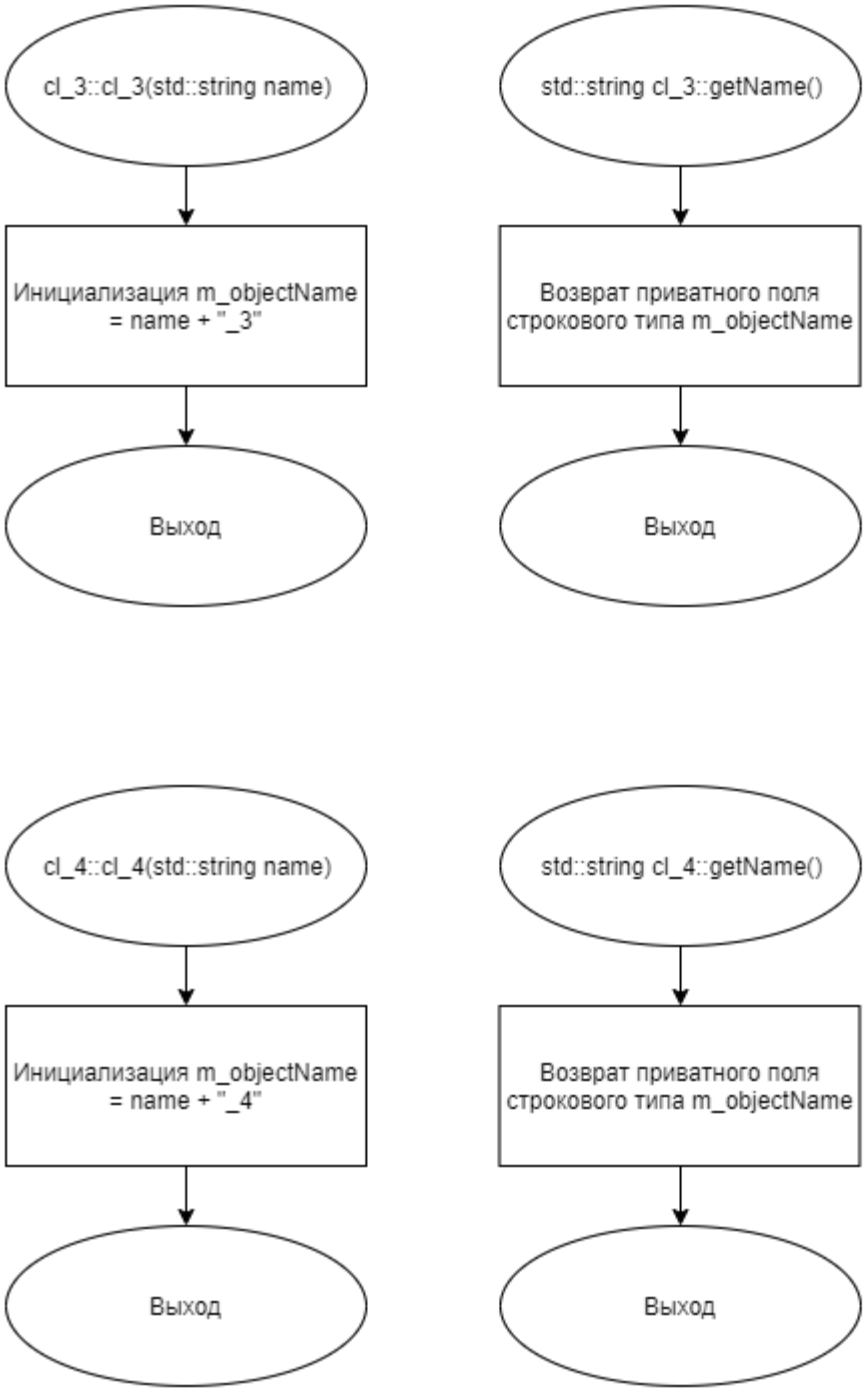

Рисунок 2 – Блок-схема алгоритма

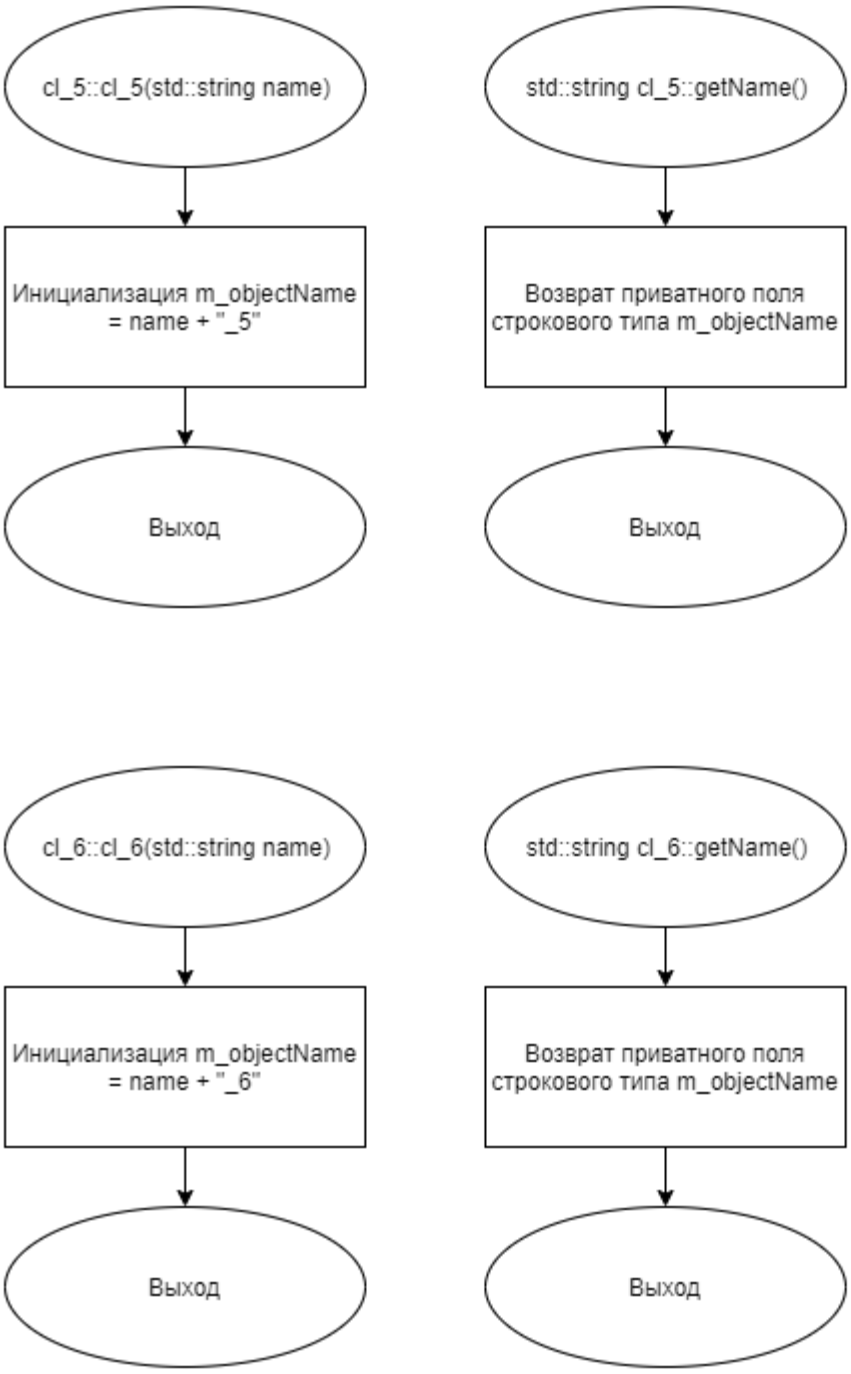

Рисунок 3 – Блок-схема алгоритма

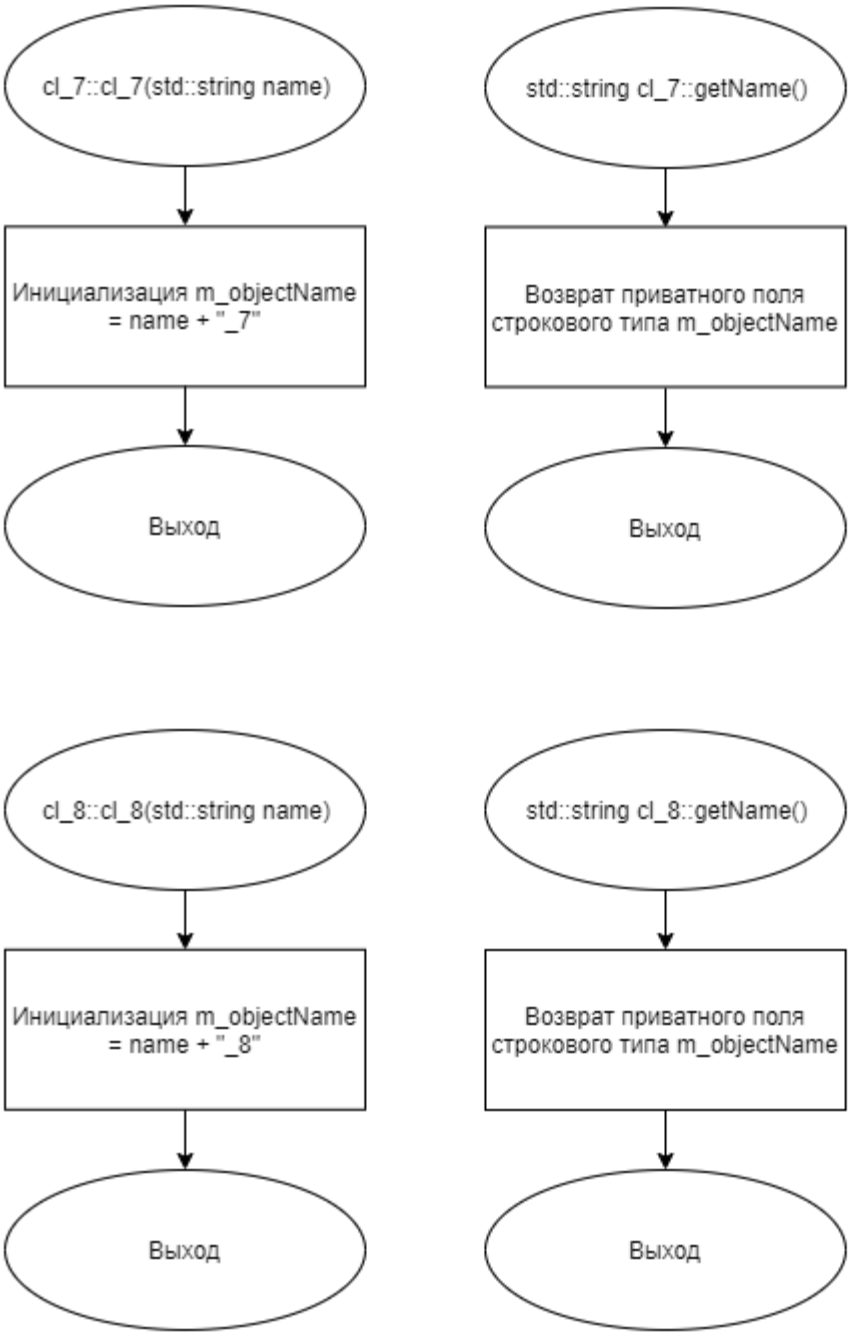

Рисунок 4 – Блок-схема алгоритма

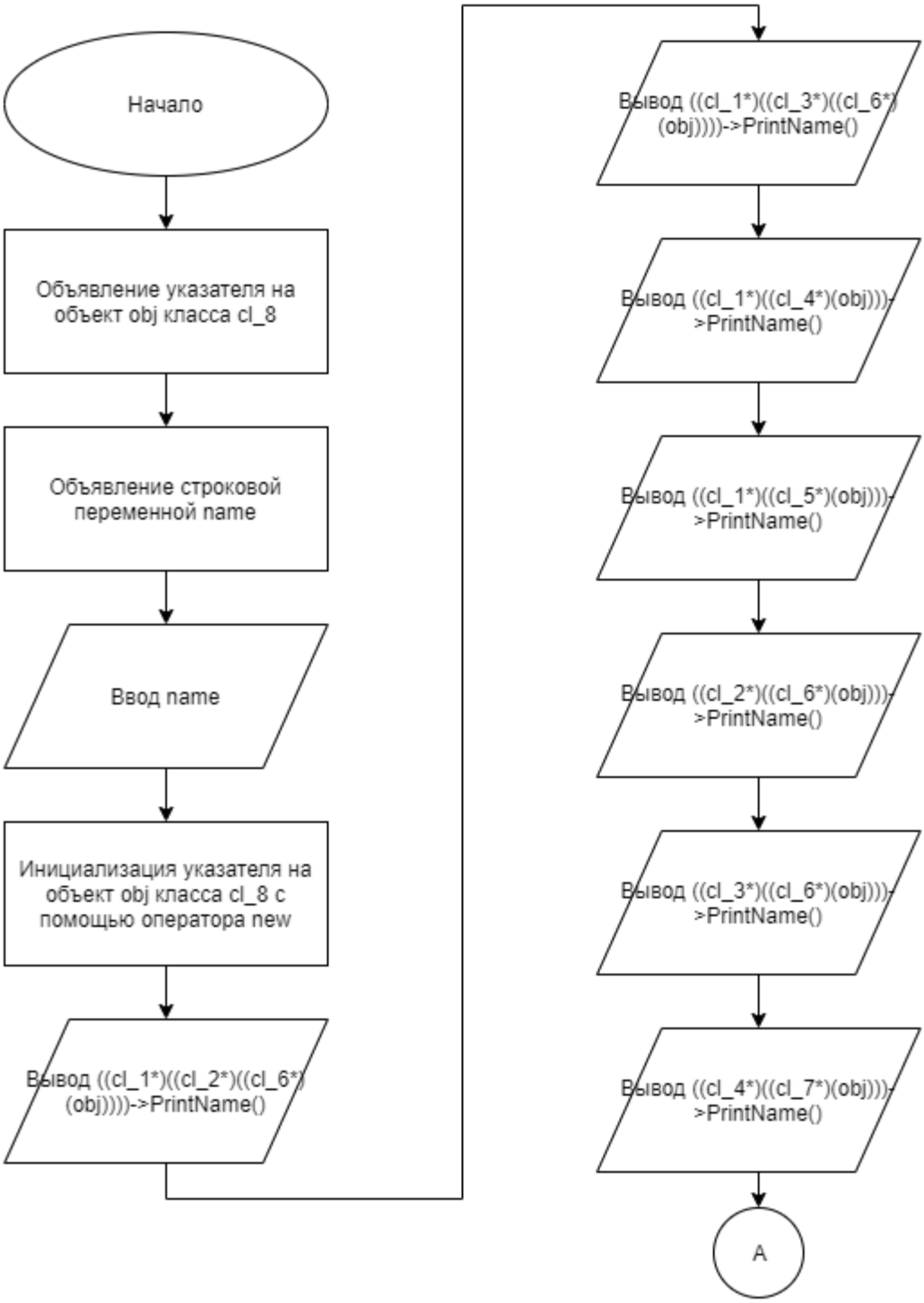

Рисунок 5 – Блок-схема алгоритма

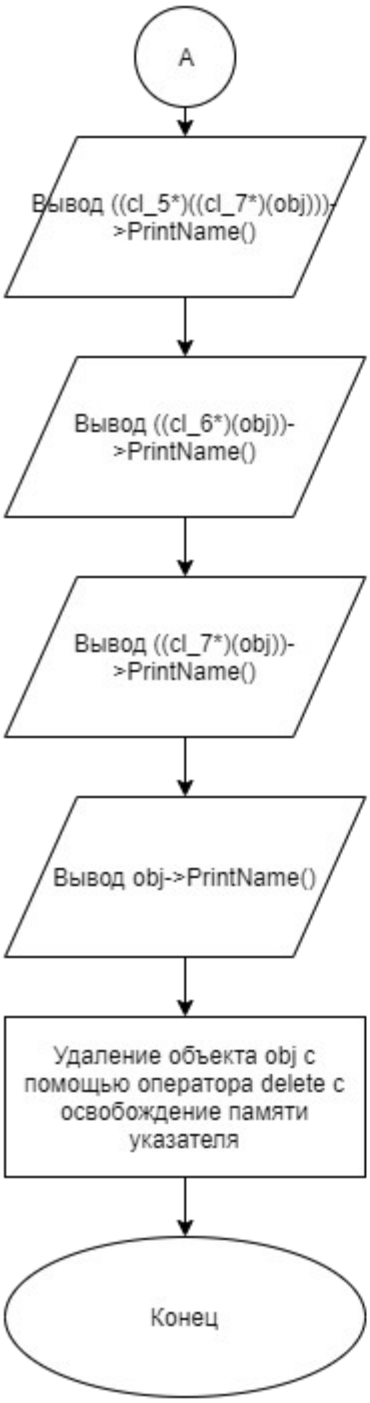

Рисунок 6 – Блок-схема алгоритма

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(std::string name)
{
   m_objectName = name + "_1";
}

std::string cl_1::getName()
{
   return m_objectName;
}
```

5.2 Файл cl_1.h

Листинг 2 – cl_1.h

```
#ifndef __CL_1__H
#define __CL_1__H

#include <string>
#include <iostream>

class cl_1 {
   std::string m_objectName;
public:
   cl_1(std::string name);
   std::string getName();
};

#endif
```

5.3 Файл cl_2.cpp

Листинг 3 – cl_2.cpp

```
#include "cl_2.h"

cl_2::cl_2(std::string name) : cl_1(name + "_2")
{
   m_objectName = name + "_2";
}

std::string cl_2::getName()
{
   return m_objectName;
}
```

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 {
   std::string m_objectName;
public:
   cl_2(std::string name);
   std::string getName();
};

#endif
```

5.5 Файл cl_3.cpp

Листинг 5 – cl_3.cpp

```
#include "cl_3.h"

cl_3::cl_3(std::string name) : cl_1(name + "_3")
{
   m_objectName = name + "_3";
```

```
}

std::string cl_3::getName()
{
   return m_objectName;
}
```

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 {
   std::string m_objectName;
public:
   cl_3(std::string name);
   std::string getName();
};

#endif
```

5.7 Файл cl_4.cpp

Листинг 7 – cl_4.cpp

```
#include "cl_4.h"

cl_4::cl_4(std::string name) : cl_1(name + "_4")
{
   m_objectName = name + "_4";
}

std::string cl_4::getName()
{
   return m_objectName;
}
```

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 {
   std::string m_objectName;
public:
   cl_4(std::string name);
   std::string getName();
};

#endif
```

5.9 Файл cl_5.cpp

Листинг 9 – cl_5.cpp

```
#include "cl_5.h"

cl_5::cl_5(std::string name) : cl_1(name + "_5")
{
   m_objectName = name + "_5";
}

std::string cl_5::getName()
{
   return m_objectName;
}
```

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 {
   std::string m_objectName;
public:
   cl_5(std::string name);
   std::string getName();
};

#endif
```

5.11 Файл cl_6.cpp

Листинг 11 – cl_6.cpp

```
#include "cl_6.h"

cl_6::cl_6(std::string name) : cl_2(name + "_6"), cl_3(name + "_6")
{
   m_objectName = name + "_6";
}

std::string cl_6::getName()
{
   return m_objectName;
}
```

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 {
   std::string m_objectName;
public:
   cl_6(std::string name);
   std::string getName();
};

#endif
```

5.13 Файл cl_7.cpp

Листинг 13 – cl_7.cpp

```
#include "cl_7.h"

cl_7::cl_7(std::string  name)  :  cl_4(name  +  "_7"),  cl_5(name  +  "_7"),
cl_1(name + "_7")
{
   m_objectName = name + "_7";
}

std::string cl_7::getName()
{
   return m_objectName;
}
```

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 {
   std::string m_objectName;
public:
   cl_7(std::string name);
   std::string getName();
};

#endif
```

5.15 Файл cl_8.cpp

Листинг 15 – cl_8.cpp

```
#include "cl_8.h"

cl_8::cl_8(std::string  name)  :  cl_6(name  +  "_8"),  cl_7(name  +  "_8"),
```

```
cl_1(name + "_8")
{
   m_objectName = name + "_8";
}

std::string cl_8::getName()
{
   return m_objectName;
}
```

5.16 Файл cl_8.h

Листинг 16 – cl_8.h

```
#ifndef __CL_8__H
#define __CL_8__H

#include "cl_1.h"
#include "cl_6.h"
#include "cl_7.h"

class cl_8 : public cl_6, public cl_7 {
   std::string m_objectName;
public:
   cl_8(std::string name);
   std::string getName();
};


#endif
```

5.17 Файл main.cpp

Листинг 17 – main.cpp

```
#include "cl_8.h"

int main()
{
   cl_8* obj;

   std::string name;
   std::cin >> name;

   obj = new cl_8(name);
```

```
        std::cout << ((cl_1*)((cl_2*)((cl_6*)(obj))))->getName() << std::endl;
        std::cout << ((cl_1*)((cl_3*)((cl_6*)(obj))))->getName() << std::endl;
        std::cout << ((cl_1*)((cl_4*)(obj)))->getName() << std::endl;
        std::cout << ((cl_1*)((cl_5*)(obj)))->getName() << std::endl;
        std::cout << ((cl_2*)((cl_6*)(obj)))->getName() << std::endl;
        std::cout << ((cl_3*)((cl_6*)(obj)))->getName() << std::endl;
        std::cout << ((cl_4*)((cl_7*)(obj)))->getName() << std::endl;
        std::cout << ((cl_5*)((cl_7*)(obj)))->getName() << std::endl;
        std::cout << ((cl_6*)(obj))->getName() << std::endl;
        std::cout << ((cl_7*)(obj))->getName() << std::endl;
        std::cout << obj->getName() << std::endl;

        delete obj;

        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).