

Здесь будет титульник, листай ниже

СОДЕРЖАНИЕ

| | |
|---|----|
| 1 ПОСТАНОВКА ЗАДАЧИ..... | 6 |
| 1.1 Описание входных данных..... | 7 |
| 1.2 Описание выходных данных..... | 8 |
| 2 МЕТОД РЕШЕНИЯ..... | 9 |
| 3 ОПИСАНИЕ АЛГОРИТМОВ..... | 14 |
| 3.1 Алгоритм конструктора класса cl_1..... | 14 |
| 3.2 Алгоритм метода getName класса cl_1..... | 14 |
| 3.3 Алгоритм конструктора класса cl_2..... | 15 |
| 3.4 Алгоритм метода getName класса cl_2..... | 15 |
| 3.5 Алгоритм конструктора класса cl_3..... | 15 |
| 3.6 Алгоритм метода getName класса cl_3..... | 16 |
| 3.7 Алгоритм конструктора класса cl_4..... | 16 |
| 3.8 Алгоритм метода getName класса cl_4..... | 16 |
| 3.9 Алгоритм конструктора класса cl_5..... | 17 |
| 3.10 Алгоритм метода getName класса cl_5..... | 17 |
| 3.11 Алгоритм конструктора класса cl_6..... | 18 |
| 3.12 Алгоритм метода getName класса cl_6..... | 18 |
| 3.13 Алгоритм конструктора класса cl_7..... | 18 |
| 3.14 Алгоритм метода getName класса cl_7..... | 19 |
| 3.15 Алгоритм конструктора класса cl_8..... | 19 |
| 3.16 Алгоритм метода getName класса cl_8..... | 19 |
| 3.17 Алгоритм функции main..... | 20 |
| 4 БЛОК-СХЕМЫ АЛГОРИТМОВ..... | 22 |
| 5 КОД ПРОГРАММЫ..... | 27 |
| 5.1 Файл cl_1.cpp..... | 27 |
| 5.2 Файл cl_1.h..... | 27 |

| | | |
|------|---------------------------------------|----|
| 5.3 | Файл cl_2.cpp..... | 28 |
| 5.4 | Файл cl_2.h..... | 28 |
| 5.5 | Файл cl_3.cpp..... | 28 |
| 5.6 | Файл cl_3.h..... | 29 |
| 5.7 | Файл cl_4.cpp..... | 29 |
| 5.8 | Файл cl_4.h..... | 30 |
| 5.9 | Файл cl_5.cpp..... | 30 |
| 5.10 | Файл cl_5.h..... | 30 |
| 5.11 | Файл cl_6.cpp..... | 31 |
| 5.12 | Файл cl_6.h..... | 31 |
| 5.13 | Файл cl_7.cpp..... | 32 |
| 5.14 | Файл cl_7.h..... | 32 |
| 5.15 | Файл cl_8.cpp..... | 33 |
| 5.16 | Файл cl_8.h..... | 33 |
| 5.17 | Файл main.cpp..... | 33 |
| 6 | ТЕСТИРОВАНИЕ..... | 35 |
| | СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ..... | 36 |

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 параметризованного конструктора, вызвать параметризованный 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 параметризованного 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_n` предназначен для демонстрации множественного наследования;
- библиотека `iostream`;
- объекты стандартного потока ввода/вывода данных `cin/cout`;
- пространство имен `std`.

Класс `cl_1`:

- свойства/поля:
 - поле наименование объекта:
 - наименование — `name`;
 - тип — `string`;
 - модификатор доступа — `private`;
- функционал:
 - метод `cl_1` — параметризованный конструктор, задает значение скрытому полю;
 - метод `getName` — возвращает значения закрытого поля.

Класс `cl_2`:

- свойства/поля:
 - поле наименование объекта:
 - наименование — `name`;
 - тип — `string`;
 - модификатор доступа — `private`;
- функционал:
 - метод `cl_2` — параметризованный конструктор, задает значение скрытому полю;

- о метод getName — возвращает значения закрытого поля.

Класс cl_3:

- свойства/поля:
 - о поле наименование объекта:
 - наименование — name;
 - тип — string;
 - модификатор доступа — private;
- функционал:
 - о метод cl_3 — параметризованный конструктор, задает значение скрытому полю;
 - о метод getName — возвращает значения закрытого поля.

Класс cl_4:

- свойства/поля:
 - о поле наименование объекта:
 - наименование — name;
 - тип — string;
 - модификатор доступа — private;
- функционал:
 - о метод cl_4 — параметризованный конструктор, задает значение скрытому полю;
 - о метод getName — возвращает значения закрытого поля.

Класс cl_5:

- свойства/поля:
 - о поле наименование объекта:
 - наименование — name;
 - тип — string;
 - модификатор доступа — private;

- функционал:
 - метод cl_5 — параметризированный конструктор, задает значение скрытому полю;
 - метод getName — возвращает значения закрытого поля.

Класс cl_6:

- свойства/поля:
 - поле наименование объекта:
 - наименование — name;
 - тип — string;
 - модификатор доступа — private;
- функционал:
 - метод cl_6 — параметризированный конструктор, задает значение скрытому полю;
 - метод getName — возвращает значения закрытого поля.

Класс cl_7:

- свойства/поля:
 - поле наименование объекта:
 - наименование — name;
 - тип — string;
 - модификатор доступа — private;
- функционал:
 - метод cl_7 — параметризированный конструктор, задает значение скрытому полю;
 - метод getName — возвращает значения закрытого поля.

Класс cl_8:

- свойства/поля:
 - поле наименование объекта:

- наименование — name;
- тип — string;
- модификатор доступа — private;
- функционал:
 - метод cl_8 — параметризированный конструктор, задает значение скрытому полю;
 - метод getName — возвращает значения закрытого поля.

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 |
| 7 | cl_7 | | | | |
| | | cl_8 | public | | 8 |

| № | Имя класса | Классы-наследники | Модификатор доступа при наследовании | Описание | Номер |
|----------|-------------------|--------------------------|---|-----------------|--------------|
| 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: параметризированный конструктор, задает значение скрытому полю.

Параметры: string name.

Алгоритм конструктора представлен в таблице 2.

Таблица 2 – Алгоритм конструктора класса cl_1

| № | Предикат | Действия | № перехода |
|---|----------|--|---------------|
| 1 | | присвоение значению закрытого поля name значения name + "_1" | Ø |

3.2 Алгоритм метода getName класса cl_1

Функционал: возвращает значения закрытого поля.

Параметры: нет.

Возвращаемое значение: string.

Алгоритм метода представлен в таблице 3.

Таблица 3 – Алгоритм метода getName класса cl_1

| № | Предикат | Действия | № перехода |
|---|----------|---------------------------------|---------------|
| 1 | | возврат значения закрытого поля | Ø |

3.3 Алгоритм конструктора класса cl_2

Функционал: параметризированный конструктор, задает значение скрытому полю.

Параметры: string name.

Алгоритм конструктора представлен в таблице 4.

Таблица 4 – Алгоритм конструктора класса cl_2

| № | Предикат | Действия | № перехода |
|---|----------|--|---------------|
| 1 | | присвоение значению закрытого поля name значения name + "_2" | Ø |

3.4 Алгоритм метода getName класса cl_2

Функционал: возвращает значения закрытого поля.

Параметры: нет.

Возвращаемое значение: string.

Алгоритм метода представлен в таблице 5.

Таблица 5 – Алгоритм метода getName класса cl_2

| № | Предикат | Действия | № перехода |
|---|----------|---------------------------------|---------------|
| 1 | | возврат значения закрытого поля | Ø |

3.5 Алгоритм конструктора класса cl_3

Функционал: параметризированный конструктор, задает значение скрытому полю.

Параметры: string name.

Алгоритм конструктора представлен в таблице 6.

Таблица 6 – Алгоритм конструктора класса *cl_3*

| № | Предикат | Действия | № перехода |
|---|----------|--|---------------|
| 1 | | присвоение значению закрытого поля name значения name + "_3" | Ø |

3.6 Алгоритм метода getName класса *cl_3*

Функционал: возвращает значения закрытого поля.

Параметры: нет.

Возвращаемое значение: string.

Алгоритм метода представлен в таблице 7.

Таблица 7 – Алгоритм метода *getName* класса *cl_3*

| № | Предикат | Действия | № перехода |
|---|----------|---------------------------------|---------------|
| 1 | | возврат значения закрытого поля | Ø |

3.7 Алгоритм конструктора класса *cl_4*

Функционал: параметризированный конструктор, задает значение скрытому полю.

Параметры: string name.

Алгоритм конструктора представлен в таблице 8.

Таблица 8 – Алгоритм конструктора класса *cl_4*

| № | Предикат | Действия | № перехода |
|---|----------|--|---------------|
| 1 | | присвоение значению закрытого поля name значения name + "_4" | Ø |

3.8 Алгоритм метода getName класса *cl_4*

Функционал: возвращает значения закрытого поля.

Параметры: нет.

Возвращаемое значение: string.

Алгоритм метода представлен в таблице 9.

Таблица 9 – Алгоритм метода getName класса cl_4

| № | Предикат | Действия | № перехода |
|---|----------|---------------------------------|---------------|
| 1 | | возврат значения закрытого поля | Ø |

3.9 Алгоритм конструктора класса cl_5

Функционал: параметризированный конструктор, задает значение скрытому полю.

Параметры: string name.

Алгоритм конструктора представлен в таблице 10.

Таблица 10 – Алгоритм конструктора класса cl_5

| № | Предикат | Действия | № перехода |
|---|----------|--|---------------|
| 1 | | присвоение значению закрытого поля name значения name + "_5" | Ø |

3.10 Алгоритм метода getName класса cl_5

Функционал: возвращает значения закрытого поля.

Параметры: нет.

Возвращаемое значение: string.

Алгоритм метода представлен в таблице 11.

Таблица 11 – Алгоритм метода getName класса cl_5

| № | Предикат | Действия | № перехода |
|---|----------|---------------------------------|---------------|
| 1 | | возврат значения закрытого поля | Ø |

3.11 Алгоритм конструктора класса cl_6

Функционал: параметризированный конструктор, задает значение скрытому полю.

Параметры: string name.

Алгоритм конструктора представлен в таблице 12.

Таблица 12 – Алгоритм конструктора класса cl_6

| № | Предикат | Действия | № перехода |
|---|----------|--|---------------|
| 1 | | присвоение значению закрытого поля name значения name + "_6" | Ø |

3.12 Алгоритм метода getName класса cl_6

Функционал: возвращает значения закрытого поля.

Параметры: нет.

Возвращаемое значение: string.

Алгоритм метода представлен в таблице 13.

Таблица 13 – Алгоритм метода getName класса cl_6

| № | Предикат | Действия | № перехода |
|---|----------|---------------------------------|---------------|
| 1 | | возврат значения закрытого поля | Ø |

3.13 Алгоритм конструктора класса cl_7

Функционал: параметризированный конструктор, задает значение скрытому полю.

Параметры: string name.

Алгоритм конструктора представлен в таблице 14.

Таблица 14 – Алгоритм конструктора класса cl_7

| № | Предикат | Действия | № перехода |
|---|----------|--|---------------|
| 1 | | присвоение значению закрытого поля name значения name + "_7" | Ø |

3.14 Алгоритм метода getName класса cl_7

Функционал: возвращает значения закрытого поля.

Параметры: нет.

Возвращаемое значение: string.

Алгоритм метода представлен в таблице 15.

Таблица 15 – Алгоритм метода getName класса cl_7

| № | Предикат | Действия | № перехода |
|---|----------|---------------------------------|---------------|
| 1 | | возврат значения закрытого поля | Ø |

3.15 Алгоритм конструктора класса cl_8

Функционал: параметризированный конструктор, задает значение скрытому полю.

Параметры: string name.

Алгоритм конструктора представлен в таблице 16.

Таблица 16 – Алгоритм конструктора класса cl_8

| № | Предикат | Действия | № перехода |
|---|----------|--|---------------|
| 1 | | присвоение значению закрытого поля name значения name + "_8" | Ø |

3.16 Алгоритм метода getName класса cl_8

Функционал: возвращает значения закрытого поля.

Параметры: нет.

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 | Ø |

3.17 Алгоритм функции main

Функционал: главная функция программы.

Параметры: нет.

Возвращаемое значение: целое, индикатор работоспособности программы.

Алгоритм функции представлен в таблице 18.

Таблица 18 – Алгоритм функции main

| № | Предикат | Действия | № перехода |
|---|----------|---|---------------|
| 1 | | объявление указателя sobj на cl_8 | 2 |
| 2 | | объявление переменной name типа string | 3 |
| 3 | | вывод значения переменной name | 4 |
| 4 | | создание объекта obj класса cl_8 с помощью параметризованного конструктора с параметром name | 5 |
| 5 | | присвоение указателю sobj ссылки на объект obj | 6 |
| 6 | | вызов метода getName() через указатель sobj для объекта класса cl_1, созданного объектом класса cl_2, и вывод его значения на экран | 7 |
| 7 | | вызов метода getName() через указатель sobj для объекта класса cl_1, созданного объектом класса cl_3, и вывод его значения на экран | 8 |
| 8 | | вызов метода getName() через указатель sobj для объекта класса cl_1, созданного объектом класса cl_4 и вывод его значения на экран | 9 |
| 9 | | вызов метода getName() через указатель sobj для объекта класса cl_1, созданного объектом класса cl_5 и вывод его значения на экран | 10 |

| № | Предикат | Действия | № перехода |
|----|----------|---|---------------|
| 10 | | вызов метода getName() через указатель sobj для объекта класса cl_2 и вывод его значения на экран | 11 |
| 11 | | вызов метода getName() через указатель sobj для объекта класса cl_3 и вывод его значения на экран | 12 |
| 12 | | вызов метода getName() через указатель sobj для объекта класса cl_4 и вывод его значения на экран | 13 |
| 13 | | вызов метода getName() через указатель sobj для объекта класса cl_5 и вывод его значения на экран | 14 |
| 14 | | вызов метода getName() через указатель sobj для объекта класса cl_6 и вывод его значения на экран | 15 |
| 15 | | вызов метода getName() через указатель sobj для объекта класса cl_7 и вывод его значения на экран | 16 |
| 16 | | вызов метода getName() через указатель sobj и вывод его значения на экран | 17 |
| 17 | | возврат значения 0 | ∅ |

4 БЛОК-СХЕМЫ АЛГОРИТМОВ

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

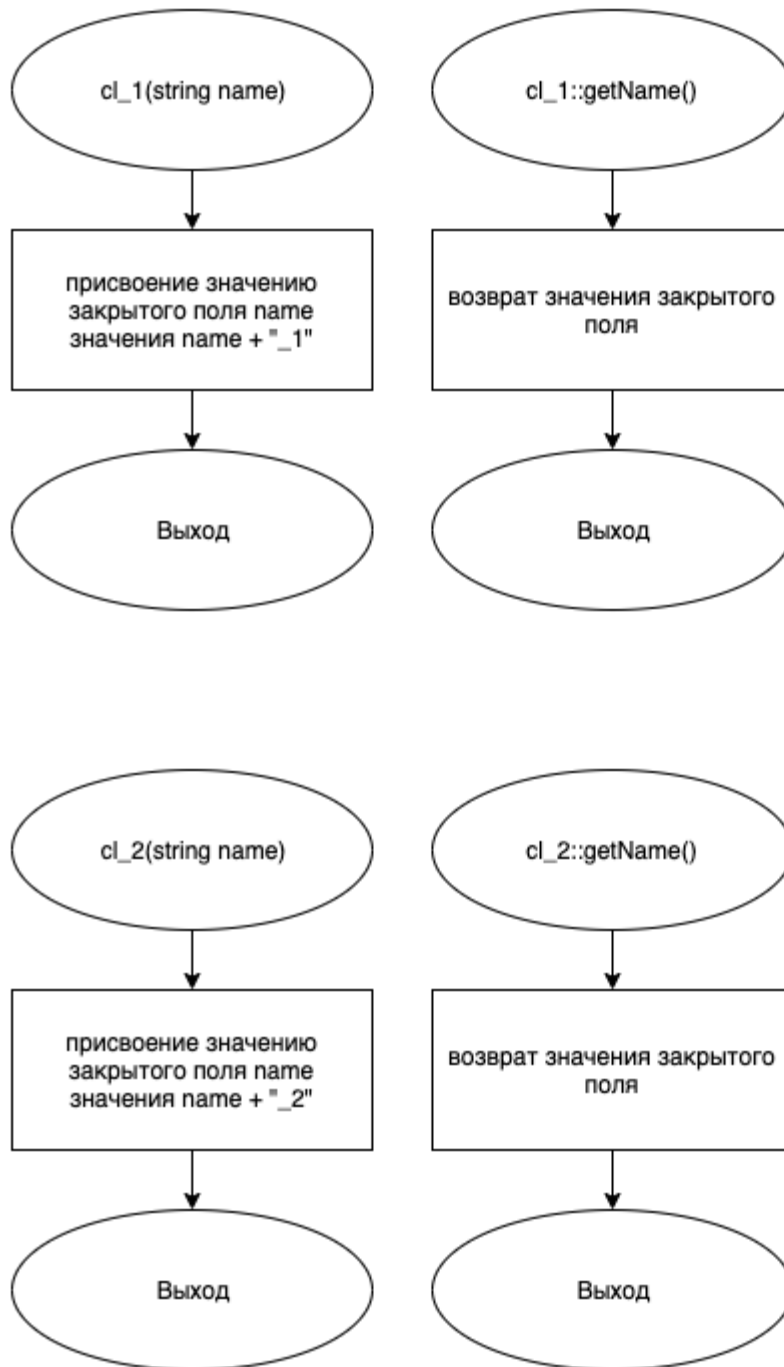


Рисунок 1 – Блок-схема алгоритма

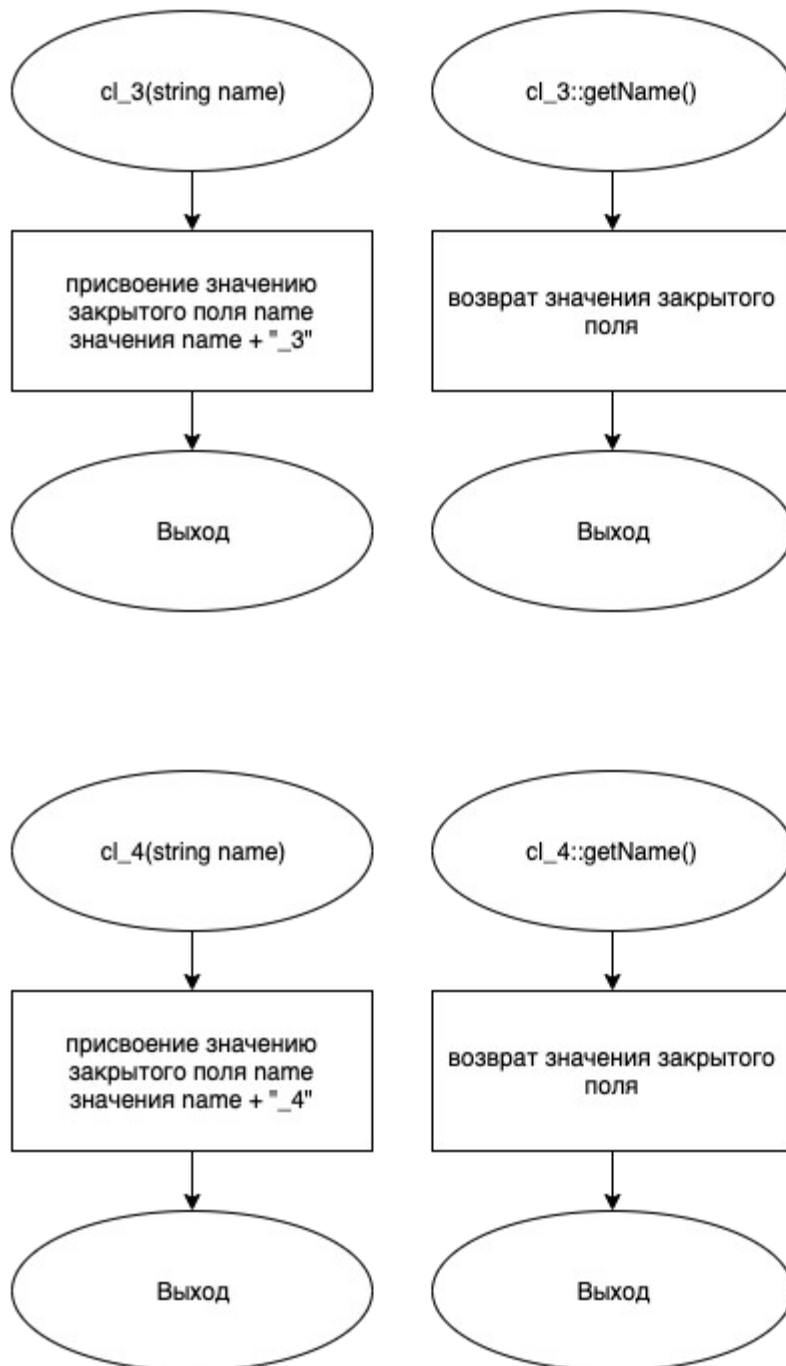


Рисунок 2 – Блок-схема алгоритма

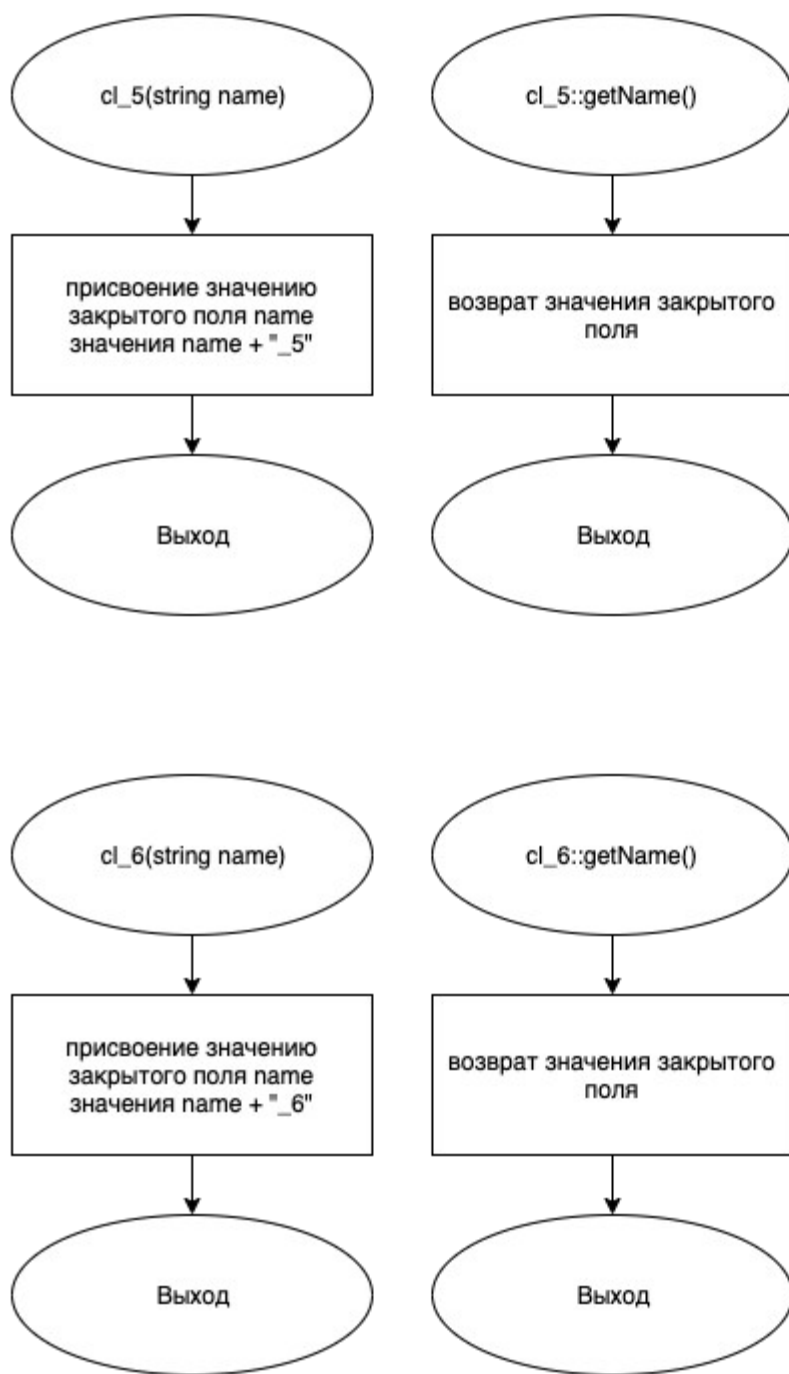


Рисунок 3 – Блок-схема алгоритма

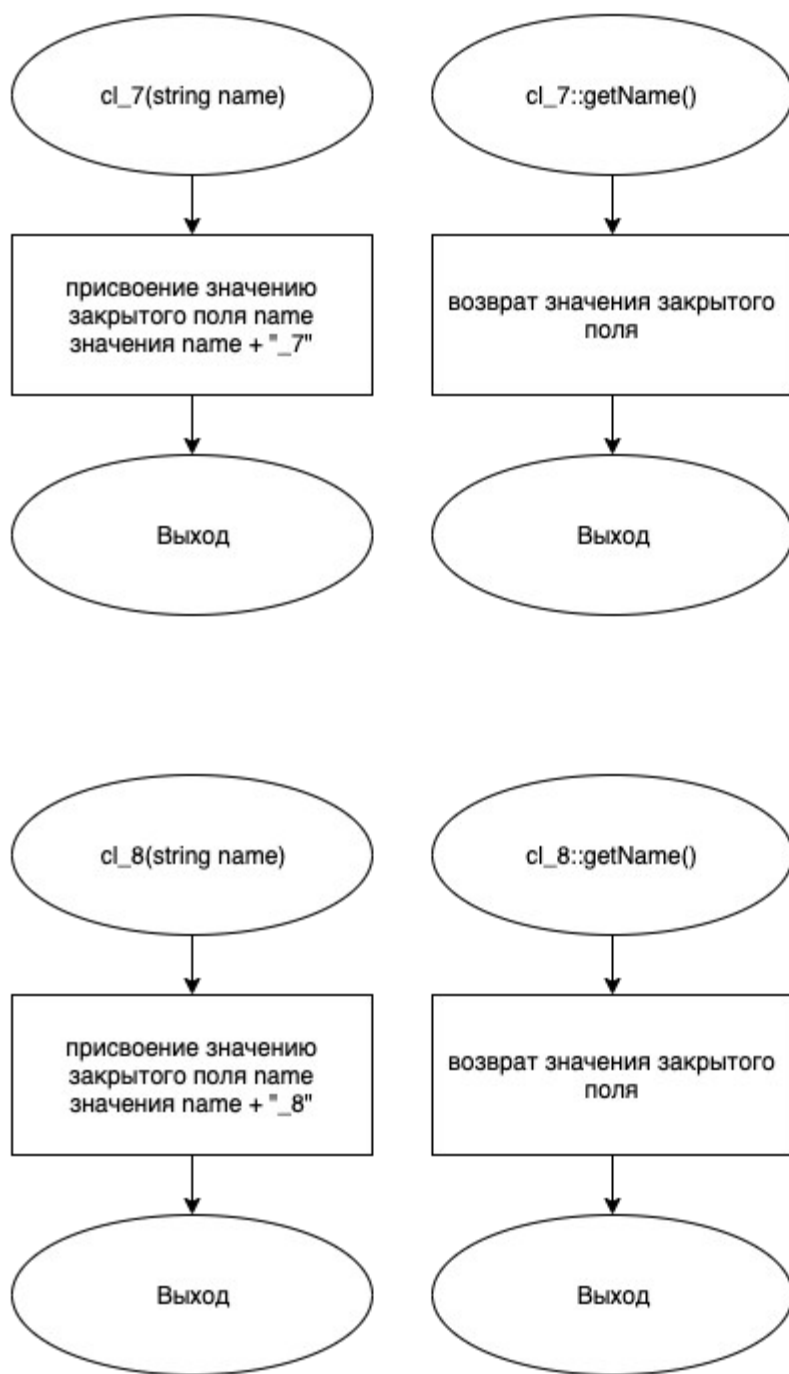


Рисунок 4 – Блок-схема алгоритма

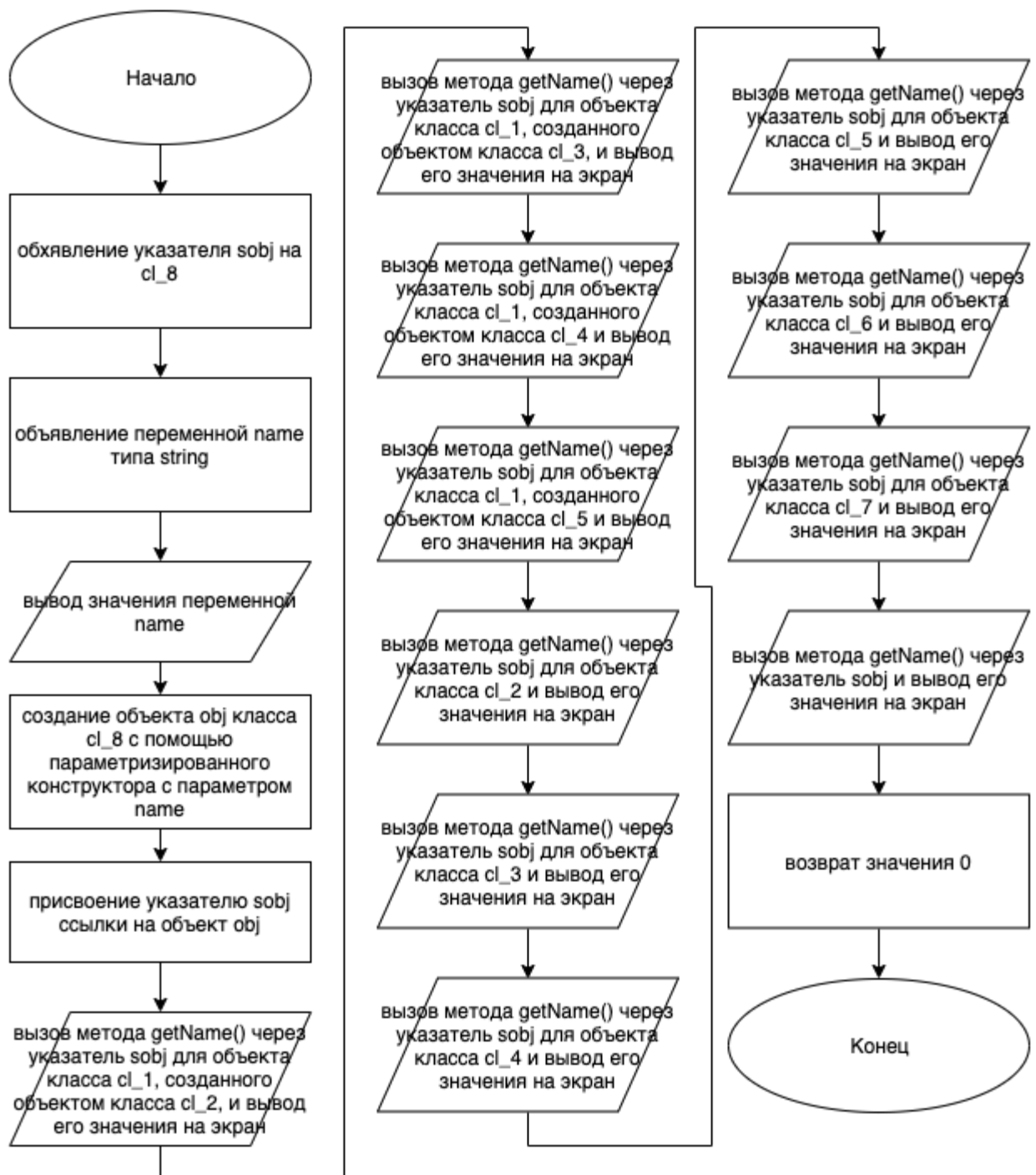


Рисунок 5 – Блок-схема алгоритма

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"
using namespace std;

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"
using namespace std;

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"
#include <iostream>
#include <string>
using namespace std;

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"
using namespace std;
```

```

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"
#include <iostream>
#include <string>
using namespace std;

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"
using namespace std;

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"
#include <iostream>
#include <string>
using namespace std;

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"
using namespace std;

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"
```

```

#include <iostream>
#include <string>
using namespace std;

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"
using namespace std;

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"
#include <iostream>
#include <string>
using namespace std;

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"
#include <iostream>
#include <string>
using namespace std;

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"
using namespace std;

cl_8::cl_8(string name):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 <iostream>
#include <string>

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

using namespace std;

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*sobj;
    string name;
    cin >> name;

    cl_8 obj(name);
    sobj=&obj;

    cout << ((cl_1*)(cl_2*)sobj)->getName() << endl;
    cout << ((cl_1*)(cl_3*)sobj)->getName() << endl;
    cout << ((cl_1*)(cl_4*)sobj)->getName() << endl;
    cout << ((cl_1*)(cl_5*)sobj)->getName() << endl;

    cout << ((cl_2*)sobj)->getName() << endl;
    cout << ((cl_3*)sobj)->getName() << endl;
    cout << ((cl_4*)sobj)->getName() << endl;
    cout << ((cl_5*)sobj)->getName() << endl;
    cout << ((cl_6*)sobj)->getName() << endl;
    cout << ((cl_7*)sobj)->getName() << endl;

    cout << sobj->getName() << endl;

    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-avvora.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-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).