МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**« МИРЭА Российский технологический университет»**

**РТУ МИРЭА**

Институт Информационных технологий

Кафедра Вычислительной техники

# УЧЕБНОЕ ЗАДАНИЕ

по дисциплине

« Объектно-ориентированное программирование»

Наименование задачи:

## « Задание 4_2_1 »

| | | |
|---|---|---|
| С тудент группы | ИКБО-13-21 | Черномуров С.А. |
| Руководитель практики | Ассистент | Асадова Ю.С. |
| Работа представлена | «___»_____ 2022 г. | _____ |
| | | *(подпись студента)* |
| Оценка | | _____ |
| | | *(подпись руководителя)* |

Москва 2022

**СОДЕРЖАНИЕ**

# ВВЕДЕНИЕ

## 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 наименования объекта класса.
Значение данного свойства определяется в параметризированном конструкторе согласно шаблону:

«значение строкового параметра»_«номер класса»

В основной функции реализовать алгоритм:

1. Объявить один указатель на объект класса x (где: x - номер класса, его надо определить).
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. Вывод реализовать в основной функции.

Наследственность реализовать так, чтобы всего объектов было 10.

**Описание входных данных**

**Первая строка:**

«идентификатор»

**Пример ввода**

Ident

**Описание выходных данных**

**Построчно (десять строк)**:

«идентификатор»_«номер класса»

**Пример вывода:**

Ident_1

Ident_1

Ident_1

Ident_2

Ident_3

Ident_4

Ident_5

Ident_6

Ident_7

Ident_8

## Метод решения

Для решения задачи используются:

- Объекты стандартных потоков ввода и вывода cin и cout соответственно. Используются для ввода с клавиатуры и вывода на экран.
- Объект ob класса Cl8. Используется для передачи ссылки в указатель на объект класса Cl8.
- Объекты классов Cl1, Cl2, Cl3, Cl4, Cl4, Cl5, Cl6, Cl7.
- **Класс Cl1:**
    - Свойства/поля:
        - Свойство:
            - Наименование - name;
            - Тип - строковый;
            - Модификатор доступа - закрытый.
    - Методы:
        - Метод Cl1:
            - Функционал - параметризированный конструктор, содержащий параметр строкового типа.
        - Метод GetName:
            - Функционал - параметризированный метод, возвращающий значение свойства name.
- **Класс Cl2:**
    - Свойства/поля:
        - Свойство:
            - Наименование - name;
            - Тип - строковый;
            - Модификатор доступа - закрытый.

- Методы:
    - Метод Cl2:
        - Функционал - параметризированный конструктор, содержащий параметр строкового типа.
    - Метод GetName:
        - Функционал - параметризированный метод, возвращающий значение свойства name.
- **Класс Cl3:**
    - Свойства/поля:
        - Свойство:
            - Наименование - name;
            - Тип - строковый;
            - Модификатор доступа - закрытый.
    - Методы:
        - Метод Cl2:
            - Функционал - параметризированный конструктор, содержащий параметр строкового типа.
        - Метод GetName:
            - Функционал - параметризированный метод, возвращающий значение свойства name.
- **Класс Cl4:**
    - Свойства/поля:
        - Свойство:
            - Наименование - name;
            - Тип - строковый;
            - Модификатор доступа - закрытый.

- Методы:
    - Метод Cl4:
        - Функционал - параметризированный конструктор, содержащий параметр строкового типа.
    - Метод GetName:
        - Функционал - параметризированный метод, возвращающий значение свойства name.
- **Класс Cl5:**
    - Свойства/поля:
        - Свойство:
            - Наименование - name;
            - Тип - строковый;
            - Модификатор доступа - закрытый.
    - Методы:
        - Метод Cl5:
            - Функционал - параметризированный конструктор, содержащий параметр строкового типа.
        - Метод GetName:
            - Функционал - параметризированный метод, возвращающий значение свойства name.
- **Класс Cl6:**
    - Свойства/поля:
        - Свойство:
            - Наименование - name;
            - Тип - строковый;
            - Модификатор доступа - закрытый.
    - Методы:
        - Метод Cl6:

            - Функционал - параметризированный конструктор, содержащий параметр строкового типа.
        - Метод GetName:
            - Функционал - параметризированный метод, возвращающий значение свойства name.
- **Класс Cl7:**
    - Свойства/поля:
        - Свойство:
            - Наименование - name;
            - Тип - строковый;
            - Модификатор доступа - закрытый.
    - Методы:
        - Метод Cl7:
            - Функционал - параметризированный конструктор, содержащий параметр строкового типа.
        - Метод GetName:
            - Функционал - параметризированный метод, возвращающий значение свойства name.
- **Класс Cl8:**
    - Свойства/поля:
        - Свойство:
            - Наименование - name;
            - Тип - строковый;
            - Модификатор доступа - закрытый.
    - Методы:
        - Метод Cl8:

- Функционал - параметризированный конструктор, содержащий параметр строкового типа.

- Метод GetName:
  - Функционал - параметризированный метод, возвращающий значение свойства name.

Иерархия наследования:

| № | Имя класса | Классы-наследники | Модификатор доступа при наследовании | Описание | Номер | Комментарий |
|---|---|---|---|---|---|---|
| 1 | Cl1 | | | Класс 1, является головным для классов 2, 3, 4, 5 | | |
| | | Cl2 | virtual public | | 2 | |
| | | Cl3 | virtual public | | 3 | |
| | | Cl4 | public | | 4 | |
| | | Cl5 | public | | 5 | |
| 2 | Cl2 | | | Класс 2, является головным для класса 6 | | |
| | | Cl6 | public | | 6 | |
| 3 | Cl3 | | | Класс 3, является головным для класса 6 | | |
| | | Cl6 | public | | 6 | |
| 4 | Cl4 | | | Класс 4, является головным для класса | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | 7 | | |
| | | Cl7 | public | | 7 | |
| 5 | Cl5 | | | Класс 5, является головным для класса 7 | | |
| | | Cl7 | public | | 7 | |
| 6 | Cl6 | | | Класс 6, является головным для класса 8 | | |
| | | Cl8 | public | | 8 | |
| 7 | Cl7 | | | Класс 7, является головным для класса 8 | | |
| | | Cl8 | public | | 8 | |
| 8 | Cl8 | | | Класс 8, является дочерним для классов 6 и 7 | | |

## Описание алгоритма

Согласно этапам разработки, после определения необходимого инструментария в разделе «Метод», составляются подробные описания алгоритмов для методов классов и функций.

Функция: main

Функционал: Основной алгоритм программы

Параметры: Отсутствуют

Возвращаемое значение: Целочисленный тип данных - код возврата

Алгоритм функции представлен в таблице 2.

Таблица 2. Алгоритм функции main

| № | Предикат | Действия | № перехода | Комментарий |
|---|---|---|---|---|
| 1 | | Объявление указателя ob8 на объект класса Cl8 | 2 | |
| 2 | | Объявление строковой переменной ident | 3 | |
| 3 | | Считывание с клавиатуры значения переменной ident | 4 | |
| 4 | | Создание объекта ob класса Cl8 путем вызова параметризированного конструктора со строковым параметром name | 5 | |
| 5 | | Присвоение указателю ob8 ссылки на объект ob класса Cl8 | 6 | |
| 6 | | Вывод на экран значения, возвращенного методом GetName объекта класса Cl1 с последующим переносом на новую строку | 7 | |
| 7 | | Вывод на экран значения, | 8 | |

| | | | | |
|---|---|---|---|---|
| | | возвращенного методом GetName объекта класса Cl1 с последующим переносом на новую строку | | |
| 8 | | Вывод на экран значения, возвращенного методом GetName объекта класса Cl1 с последующим переносом на новую строку | 9 | |
| 9 | | Вывод на экран значения, возвращенного методом GetName объекта класса Cl2 с последующим переносом на новую строку | 10 | |
| 10 | | Вывод на экран значения, возвращенного методом GetName объекта класса Cl3 с последующим переносом на новую строку | 11 | |
| 11 | | Вывод на экран значения, возвращенного методом GetName объекта класса Cl4 с последующим переносом на новую строку | 12 | |
| 12 | | Вывод на экран значения, возвращенного методом GetName объекта класса Cl5 с последующим переносом на новую строку | 13 | |
| 13 | | Вывод на экран значения, возвращенного методом GetName объекта класса Cl6 с последующим переносом на новую строку | 14 | |
| 14 | | Вывод на экран значения, возвращенного методом GetName объекта класса Cl7 с последующим переносом на | 15 | |

| | | | | |
|---|---|---|---|---|
| | | новую строку | | |
| 15 | | Вывод на экран значения, возвращенного методом GetName объекта ob класса Cl8 | ∅ | |

Конструктор класса: Cl1

Модификатор доступа: public

Функционал: Параметризированный конструктор, содержащий параметр строкового типа

Параметры: Строковый параметр name

Алгоритм конструктора представлен в таблице 3.

Таблица 3. Алгоритм конструктора класса Cl1

| № | Предикат | Действия | № перехода | Комментарий |
|---|---|---|---|---|
| 1 | | Присвоение свойству name текущего объекта значения name+"_1" | ∅ | |

Конструктор класса: Cl2

Модификатор доступа: public

Функционал: Параметризированный конструктор, содержащий параметр строкового типа

Параметры: Строковый параметр name

Алгоритм конструктора представлен в таблице 4.

Таблица 4. Алгоритм конструктора класса Cl2

| № | Предикат | Действия | № перехода | Комментарий |
|---|---|---|---|---|
| 1 | | Присвоение свойству name текущего объекта значения | ∅ | |

| | | | | |
|---|---|---|---|---|
| | | name+"_2" | | |

Конструктор класса: Cl3

Модификатор доступа: public

Функционал: Параметризированный конструктор, содержащий параметр строкового типа

Параметры: Строковый параметр name

Алгоритм конструктора представлен в таблице 5.

Таблица 5. Алгоритм конструктора класса Cl3

| № | Предикат | Действия | № перехода | Комментарий |
|---|---|---|---|---|
| 1 | | Присвоение свойству name текущего объекта значения name+"_3" | ∅ | |

Конструктор класса: Cl4

Модификатор доступа: public

Функционал: Параметризированный конструктор, содержащий параметр строкового типа

Параметры: Строковый параметр name

Алгоритм конструктора представлен в таблице 6.

Таблица 6. Алгоритм конструктора класса Cl4

| № | Предикат | Действия | № перехода | Комментарий |
|---|---|---|---|---|
| 1 | | Присвоение свойству name текущего объекта значения | ∅ | |

| | | | | |
|---|---|---|---|---|
| | | name+"_4" | | |

Конструктор класса: Cl5

Модификатор доступа: public

Функционал: Параметризированный конструктор, содержащий параметр строкового типа

Параметры: Строковый параметр name

Алгоритм конструктора представлен в таблице 7.

Таблица 7. Алгоритм конструктора класса Cl5

| № | Предикат | Действия | № перехода | Комментарий |
|---|---|---|---|---|
| 1 | | Присвоение свойству name текущего объекта значения name+"_5" | ∅ | |

Конструктор класса: Cl6

Модификатор доступа: public

Функционал: Параметризированный конструктор, содержащий параметр строкового типа

Параметры: Строковый параметр name

Алгоритм конструктора представлен в таблице 8.

Таблица 8. Алгоритм конструктора класса Cl6

| № | Предикат | Действия | № перехода | Комментарий |
|---|---|---|---|---|
| 1 | | Присвоение свойству name текущего объекта значения name+"_6" | ∅ | |

Конструктор класса: Cl7

Модификатор доступа: public

Функционал: Параметризированный конструктор, содержащий параметр строкового типа

Параметры: Строковый параметр name

Алгоритм конструктора представлен в таблице 9.

Таблица 9. Алгоритм конструктора класса Cl7

| № | Предикат | Действия | № перехода | Комментарий |
|---|---|---|---|---|
| 1 | | Присвоение свойству name текущего объекта значения name+"_7" | ∅ | |

Конструктор класса: Cl8

Модификатор доступа: public

Функционал: Параметризированный конструктор, содержащий параметр строкового типа

Параметры: Строковый параметр name

Алгоритм конструктора представлен в таблице 10.

Таблица 10. Алгоритм конструктора класса Cl8

| № | Предикат | Действия | № перехода | Комментарий |
|---|---|---|---|---|
| 1 | | Присвоение свойству name текущего объекта значения name+"_8" | ∅ | |

Класс объекта: Cl1

Модификатор доступа: public

Метод: GetName

Функционал: Параметризированный метод, возвращающий значение свойства name

Параметры: Отсутствуют

Возвращаемое значение: Строковый тип данных - значение свойства name

Алгоритм метода представлен в таблице 11.

Таблица 11. Алгоритм метода GetName класса Cl1

| № | Предикат | Действия | № перехода | Комментарий |
|---|---|---|---|---|
| 1 | | Возврат методом значения свойства name | ∅ | |

Класс объекта: Cl2

Модификатор доступа: public

Метод: GetName

Функционал: Параметризированный метод, возвращающий значение свойства name

Параметры: Отсутствуют

Возвращаемое значение: Строковый тип данных - значение свойства name

Алгоритм метода представлен в таблице 12.

Таблица 12. Алгоритм метода GetName класса Cl2

| № | Предикат | Действия | № перехода | Комментарий |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| 1 | | Возврат методом значения свойства name | ∅ | |

Класс объекта: Cl3

Модификатор доступа: public

Метод: GetName

Функционал: Параметризированный метод, возвращающий значение свойства name

Параметры: Отсутствуют

Возвращаемое значение: Строковый тип данных - значение свойства name

Алгоритм метода представлен в таблице 13.

Таблица 13. Алгоритм метода GetName класса Cl3

| № | Предикат | Действия | № перехода | Комментарий |
|---|---|---|---|---|
| 1 | | Возврат методом значения свойства name | ∅ | |

Класс объекта: Cl4

Модификатор доступа: public

Метод: GetName

Функционал: Параметризированный метод, возвращающий значение свойства name

Параметры: Отсутствуют

Возвращаемое значение: Строковый тип данных - значение свойства name

Алгоритм метода представлен в таблице 14.

Таблица 14. Алгоритм метода GetName класса Cl4

| № | Предикат | Действия | № перехода | Комментарий |
|---|---|---|---|---|
| 1 | | Возврат методом значения свойства name | ∅ | |

Класс объекта: Cl5

Модификатор доступа: public

Метод: GetName

Функционал: Параметризированный метод, возвращающий значение свойства name

Параметры: Отсутствуют

Возвращаемое значение: Строковый тип данных - значение свойства name

Алгоритм метода представлен в таблице 15.

Таблица 15. Алгоритм метода GetName класса Cl5

| № | Предикат | Действия | № перехода | Комментарий |
|---|---|---|---|---|
| 1 | | Возврат методом значения свойства name | ∅ | |

Класс объекта: Cl6

Модификатор доступа: public

Метод: GetName

Функционал: Параметризированный метод, возвращающий значение свойства name

Параметры: Отсутствуют

Возвращаемое значение: Строковый тип данных - значение свойства name

Алгоритм метода представлен в таблице 16.

Таблица 16. Алгоритм метода GetName класса Cl6

| № | Предикат | Действия | № перехода | Комментарий |
|---|---|---|---|---|
| 1 | | Возврат методом значения свойства name | ∅ | |

Класс объекта: Cl7

Модификатор доступа: public

Метод: GetName

Функционал: Параметризированный метод, возвращающий значение свойства name

Параметры: Отсутствуют

Возвращаемое значение: Строковый тип данных - значение свойства name

Алгоритм метода представлен в таблице 17.

Таблица 17. Алгоритм метода GetName класса Cl7

| № | Предикат | Действия | № перехода | Комментарий |
|---|---|---|---|---|
| 1 | | Возврат методом значения свойства name | ∅ | |

Класс объекта: Cl8

Модификатор доступа: public

Метод: GetName

Функционал: Параметризированный метод, возвращающий значение свойства name

Параметры: Отсутствуют

Возвращаемое значение: Строковый тип данных - значение свойства name

Алгоритм метода представлен в таблице 18.

Таблица 18. Алгоритм метода GetName класса Cl8

| № | Предикат | Действия | № перехода | Комментарий |
|---|---|---|---|---|
| 1 | | Возврат методом значения свойства name | ∅ | |

## Блок-схема алгоритма

Представим описание алгоритмов в графическом виде на рисунках ниже.

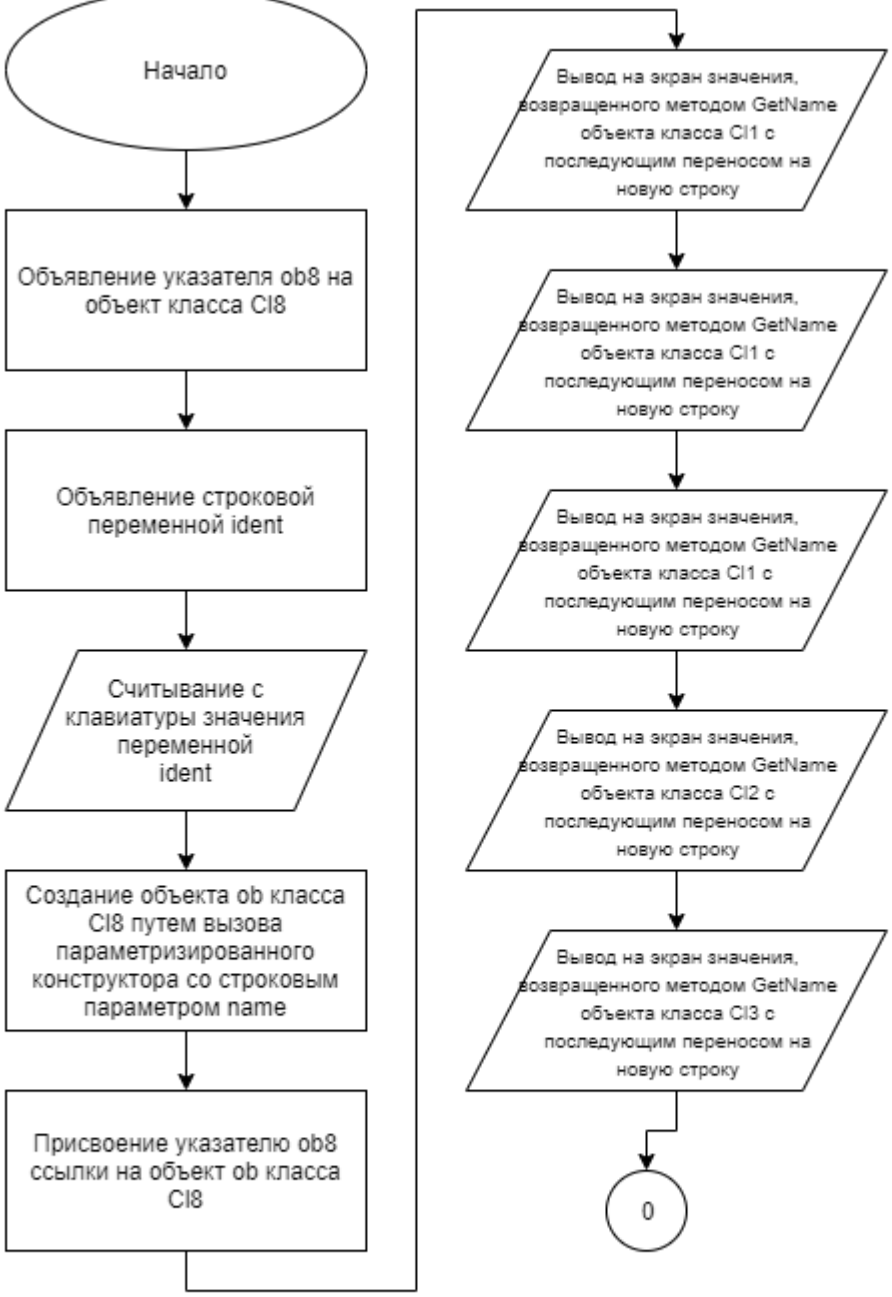


Рис. 1. Блок-схема алгоритма.

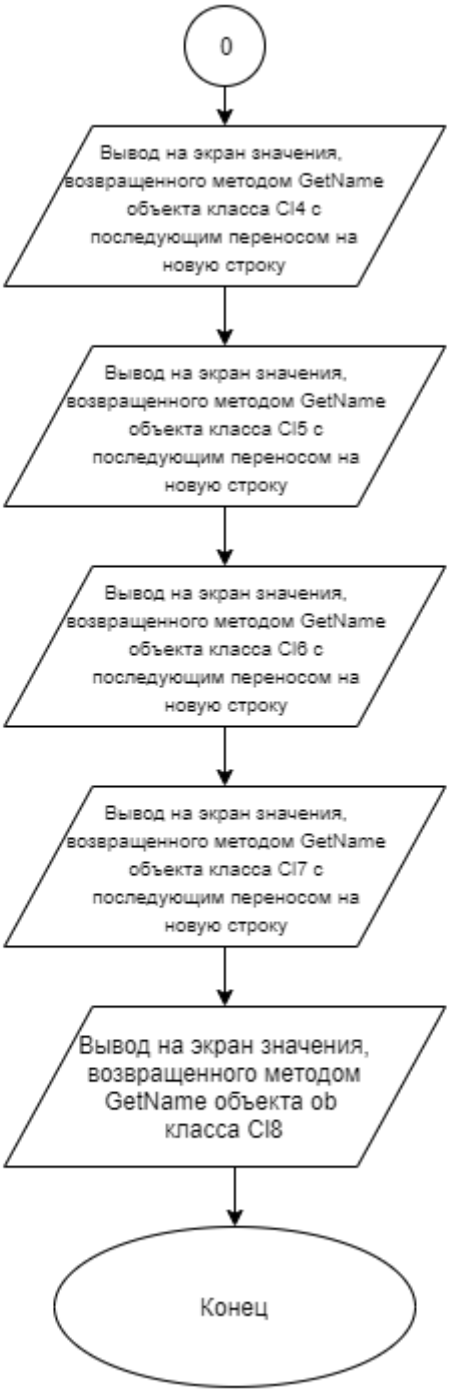


Рис. 2. Блок-схема алгоритма.

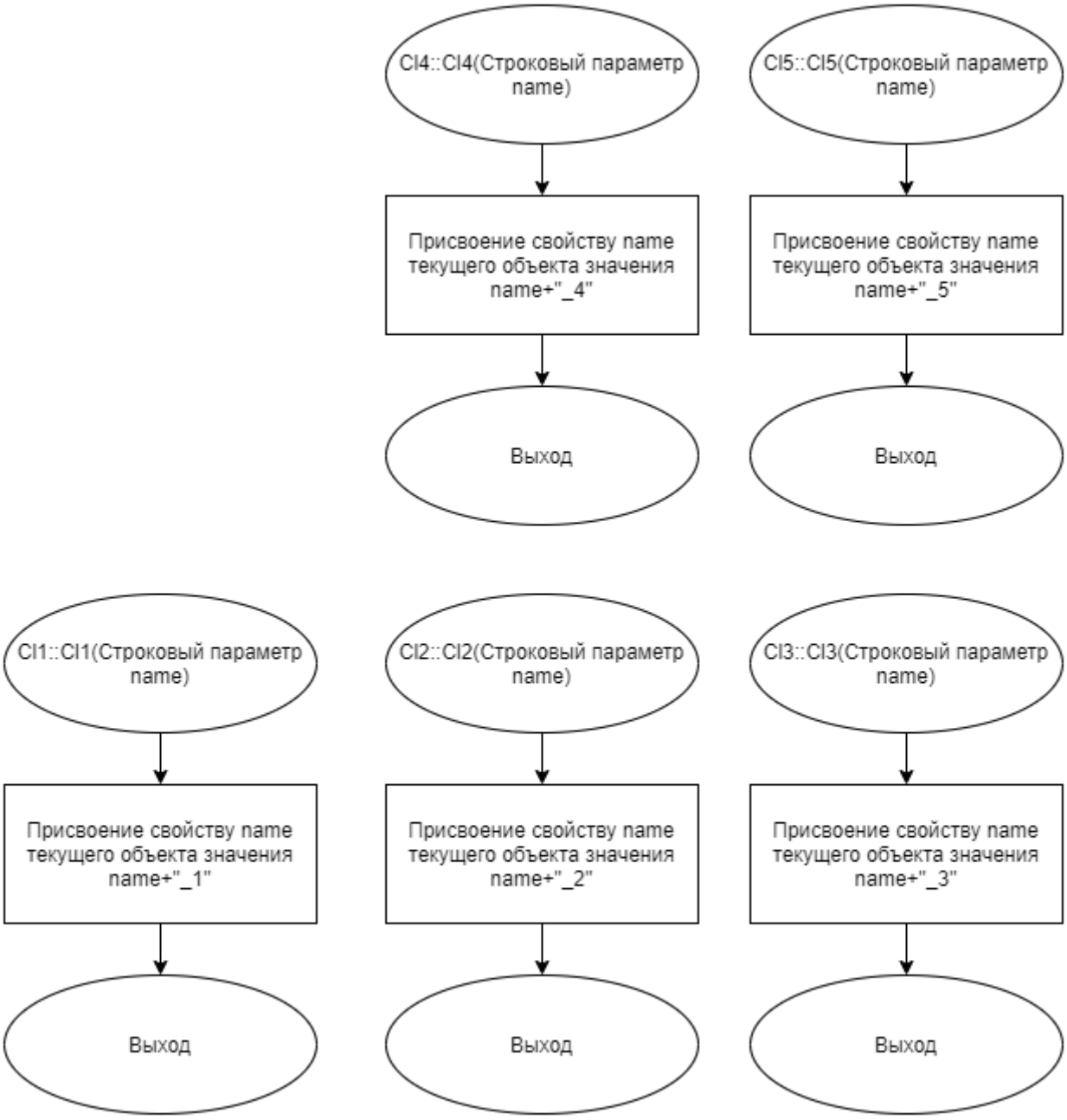


Рис. 3. Блок-схема алгоритма.

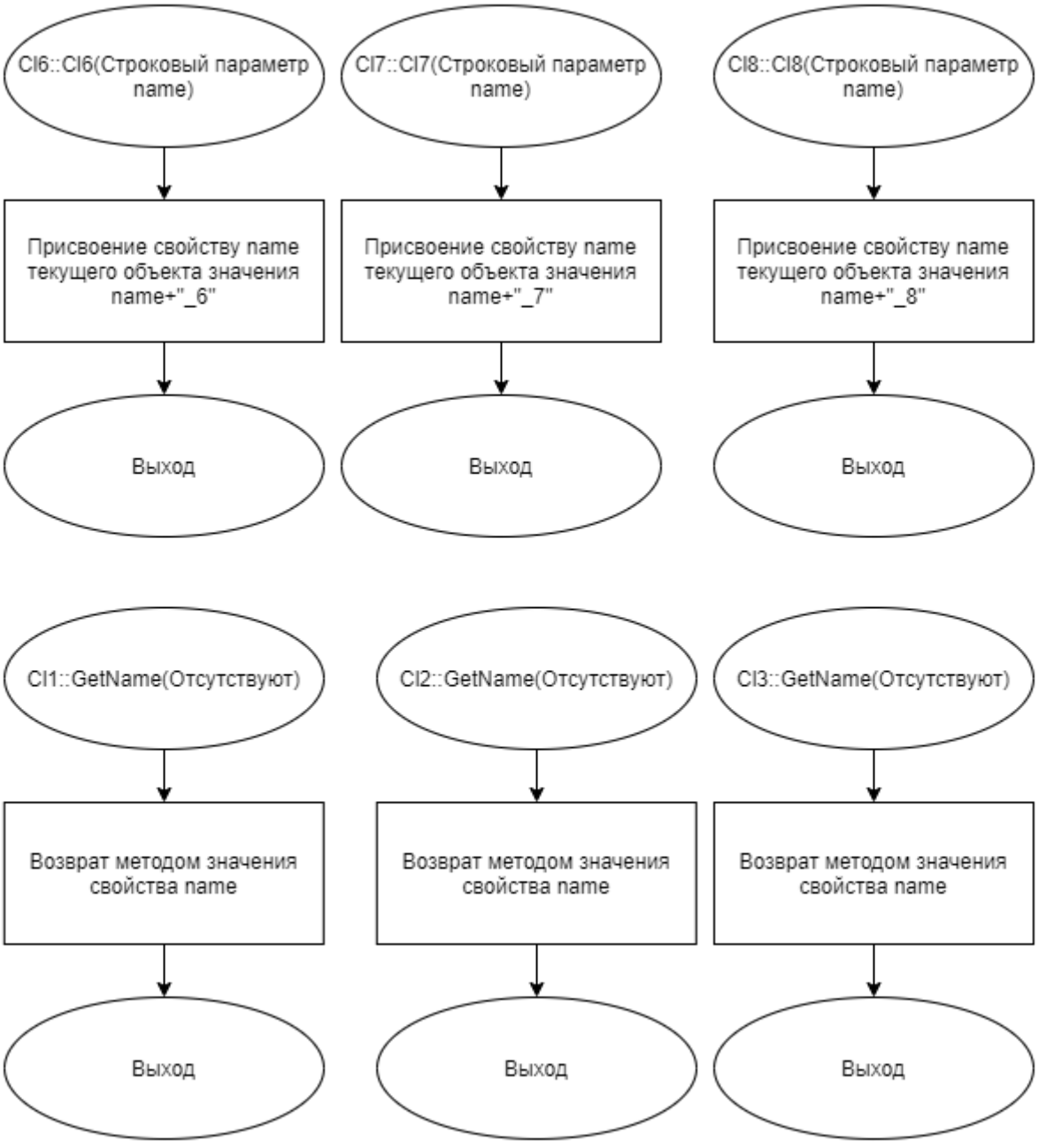


Рис. 4. Блок-схема алгоритма.

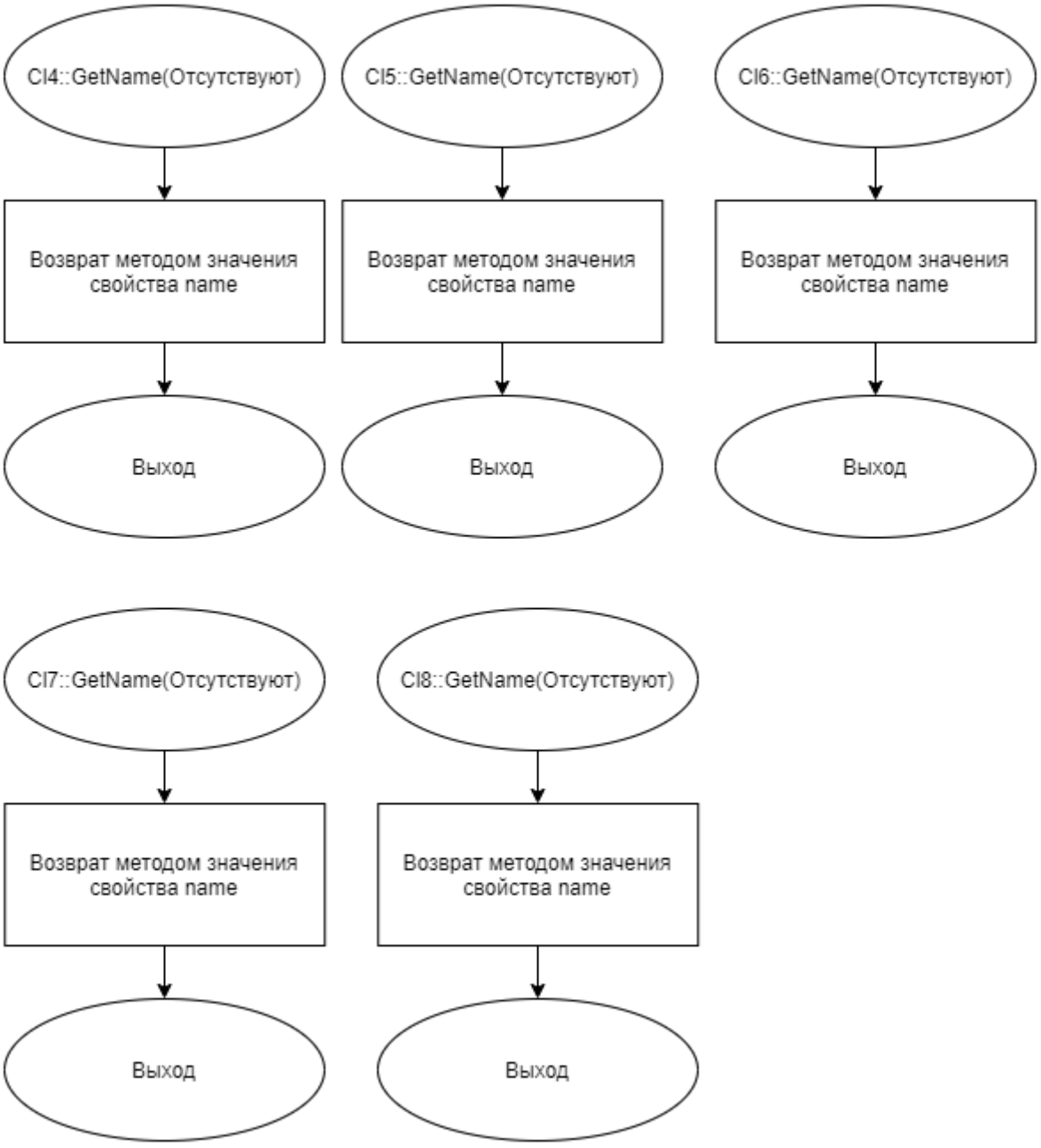


Рис. 5. Блок-схема алгоритма.

## Код программы

Программная реализация алгоритмов для решения задачи представлена ниже.

### Файл Cl1.cpp

```
#include "Cl1.h"
#include <string>
#include <iostream>
using namespace std;

Cl1 :: Cl1(string name){
        this->name=name+"_1";
}


string Cl1 :: GetName(){
        return name;
}
```

### Файл Cl1.h

```
#ifndef CL1_H
#define CL1_H

#include <string>
using namespace std;
class Cl1
{
        private:
                string name;
        public:

                Cl1(string name);
                string GetName();
};

#endif
```

### Файл Cl2.cpp

```
#include "Cl2.h"
#include <iostream>
#include <string>
using namespace std;
```

```
Cl2 :: Cl2(string name): Cl1(name){
        this->name=name+"_2";
}



string Cl2 :: GetName(){
        return name;
}
```

## Файл Cl2.h

```
#ifndef CL2_H
#define CL2_H
#include "Cl1.h"

#include <string>
using namespace std;
class Cl2 : virtual public  Cl1
{
        private:
                string name;
        public:
                Cl2(string name);
                string GetName();
};

#endif
```

## Файл Cl3.cpp

```
#include "Cl3.h"
#include <iostream>
#include <string>
using namespace std;

Cl3 :: Cl3(string name): Cl1(name){
        this->name=name+"_3";

}



string Cl3 :: GetName(){
        return name;
}
```

## Файл Cl3.h

```
#ifndef CL3_H
#define CL3_H
#include "Cl1.h"

#include <string>
using namespace std;
class Cl3 : virtual public Cl1
{
        private:
                string name;
        public:
                Cl3(string name);
                string GetName();
};

#endif
```

## Файл Cl4.cpp

```
#include "Cl4.h"
#include <iostream>
#include <string>
using namespace std;

Cl4 :: Cl4(string name): Cl1(name){
        this->name=name+"_4";

}



string Cl4 :: GetName(){
        return name;
}
```

## Файл Cl4.h

```
#ifndef CL4_H
#define CL4_H
#include "Cl1.h"

#include <string>
using namespace std;
class Cl4 : public Cl1
{
        private:
```

```
                string name;
        public:
                Cl4(string name);
                string GetName();
};

#endif
```

## Файл Cl5.cpp

```
#include "Cl5.h"
#include <iostream>
#include <string>
using namespace std;

Cl5 :: Cl5(string name): Cl1(name){
        this->name=name+"_5";

}



string Cl5 :: GetName(){
        return name;
}
```

## Файл Cl5.h

```
#ifndef CL5_H
#define CL5_H
#include "Cl1.h"

#include <string>
using namespace std;
class Cl5 : public Cl1
{
        private:
                string name;
        public:
                Cl5(string name);
                string GetName();
};

#endif
```

## Файл Cl6.cpp

```
#include "Cl6.h"
#include <iostream>
#include <string>
using namespace std;

Cl6 :: Cl6(string name):Cl1(name),Cl2(name), Cl3(name){
        this->name=name+"_6";

}



string Cl6 :: GetName(){
        return name;
}
```

## Файл Cl6.h

```
#ifndef CL6_H
#define CL6_H

#include "Cl2.h"
#include "Cl3.h"

#include <string>
using namespace std;
class Cl6 : public Cl2, public Cl3
{
        private:
                string name;
        public:
                Cl6(string name);
                string GetName();
};

#endif
```

## Файл Cl7.cpp

```
#include "Cl7.h"
#include <iostream>
#include <string>
using namespace std;

Cl7 :: Cl7(string name):Cl4(name),Cl5(name){
        this->name=name+"_7";
```

```
}


string Cl7 :: GetName(){
        return name;
}
```

## Файл Cl7.h

```
#ifndef CL7_H
#define CL7_H

#include "Cl4.h"
#include "Cl5.h"

#include <string>
using namespace std;
class Cl7 : public Cl4, public Cl5
{
        private:
                string name;
        public:
                Cl7(string name);
                string GetName();
};

#endif
```

## Файл Cl8.cpp

```
#include "Cl8.h"
#include <string>
using namespace std;

Cl8 :: Cl8(string name):Cl1(name),Cl6(name),Cl7(name){
        this->name=name+"_8";
}



string Cl8 :: GetName(){
        return name;
}
```

## Файл Cl8.h

```
#ifndef CL8_H
#define CL8_H

#include "Cl6.h"
#include "Cl7.h"

#include <string>
using namespace std;
class Cl8 : public Cl6, public Cl7
{
        private:
                string name;
        public:
                Cl8(string name);
                string GetName();
};

#endif
```

## Файл main.cpp

```
#include "Cl1.h"
#include "Cl2.h"
#include "Cl3.h"
#include "Cl4.h"
#include "Cl5.h"
#include "Cl6.h"
#include "Cl7.h"
#include "Cl8.h"

#include <iostream>
using namespace std;

int main(){
        Cl8* ob8;

        string ident;
        cin>>ident;

        Cl8 ob(ident);
        ob8=&ob;
        cout<<((Cl2*)((Cl6*)ob8))->Cl1 :: GetName()<<"\n"; //1

        cout<<((Cl2*)((Cl6*)ob8))->Cl1 :: GetName()<<"\n"; //1

        cout<<((Cl2*)((Cl6*)ob8))->Cl1 :: GetName()<<"\n"; //1

        cout<<((Cl6*)ob8)->Cl2 :: GetName()<<"\n"; //2

        cout<<((Cl6*)ob8)->Cl3 :: GetName()<<"\n"; //3
```

```
        cout<<((Cl7*)ob8)->Cl4 :: GetName()<<"\n"; //4

        cout<<((Cl7*)ob8)->Cl5 :: GetName()<<"\n"; //5

        cout<<ob8->Cl6 :: GetName()<<"\n"; //6

        cout<<ob8->Cl7 :: GetName()<<"\n"; //7

        cout<<ob8->GetName(); //8
        return 0;
}
```

## Тестирование

Результат тестирования программы представлен в следующей таблице.

| Входные данные | Ожидаемые выходные данные | Фактические выходные данные |
|---|---|---|
| ident | ident_1 ident_1 ident_1 ident_2 ident_3 ident_4 ident_5 ident_6 ident_7 ident_8 | ident_1 ident_1 ident_1 ident_2 ident_3 ident_4 ident_5 ident_6 ident_7 ident_8 |
| object | object_1 object_1 object_1 object_2 object_3 object_4 object_5 object_6 object_7 object_8 | object_1 object_1 object_1 object_2 object_3 object_4 object_5 object_6 object_7 object_8 |

# ЗАКЛЮЧЕНИЕ

## СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ (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_laboratornyh_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).