Здесь будет титульник, листай ниже

СОДЕРЖАНИЕ

1 ПОСТАНОВКА ЗАДАЧИ.....6
1.1 Описание входных данных.....7
1.2 Описание выходных данных.....7
2 МЕТОД РЕШЕНИЯ.....9
3 ОПИСАНИЕ АЛГОРИТМОВ.....13
3.1 Алгоритм конструктора класса class_1.....13
3.2 Алгоритм метода get_name класса class_1.....13
3.3 Алгоритм конструктора класса class_2.....13
3.4 Алгоритм метода get_name класса class_2.....14
3.5 Алгоритм конструктора класса class_3.....14
3.6 Алгоритм метода get_name класса class_3.....15
3.7 Алгоритм конструктора класса class_4.....15
3.8 Алгоритм метода get_name класса class_4.....15
3.9 Алгоритм конструктора класса class_5.....16
3.10 Алгоритм метода get_name класса class_5.....16
3.11 Алгоритм конструктора класса class_6.....16
3.12 Алгоритм метода get_name класса class_6.....17
3.13 Алгоритм конструктора класса class_7.....17
3.14 Алгоритм метода get_name класса class_7.....17
3.15 Алгоритм конструктора класса class_8.....18
3.16 Алгоритм метода get_name класса class_8.....18
3.17 Алгоритм функции main.....18
4 БЛОК-СХЕМЫ АЛГОРИТМОВ.....21
5 КОД ПРОГРАММЫ.....31
5.1 Файл class_1.cpp.....31
5.2 Файл class_1.h.....31

5.3 Файл class_2.cpp.....32
5.4 Файл class_2.h.....32
5.5 Файл class_3.cpp.....32
5.6 Файл class_3.h.....33
5.7 Файл class_4.cpp.....33
5.8 Файл class_4.h.....33
5.9 Файл class_5.cpp.....34
5.10 Файл class_5.h.....34
5.11 Файл class_6.cpp.....35
5.12 Файл class_6.h.....35
5.13 Файл class_7.cpp.....35
5.14 Файл class_7.h.....36
5.15 Файл class_8.cpp.....36
5.16 Файл class_8.h.....37
5.17 Файл main.cpp.....37
6 ТЕСТИРОВАНИЕ.....38
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....39

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 класса class_8 предназначен для демонстрация наследования;
- cin - объект стандартного потока;
- cout - объект стандартного потока.

Класс class_1:

- свойства/поля:
    - поле хранит текстовое сообщение:
        - наименование — name;
        - тип — string;
        - модификатор доступа — private;
- функционал:
    - метод class_1 — конструктор;
    - метод get_name — возвращает значение поля name.

Класс class_2:

- свойства/поля:
    - поле хранит текстовое сообщение:
        - наименование — name;
        - тип — string;
        - модификатор доступа — private;
- функционал:
    - метод class_2 — конструктор;
    - метод get_name — возвращает значение поля name.

Класс class_3:

- свойства/поля:
    - поле хранит текстовое сообщение:

        - наименование — name;
        - тип — string;
        - модификатор доступа — private;
- функционал:
    - метод class_3 — конструктор;
    - метод get_name — возвращает значение поля name.

Класс class_4:

- свойства/поля:
    - поле хранит текстовое сообщение:
        - наименование — name;
        - тип — string;
        - модификатор доступа — private;
- функционал:
    - метод class_4 — конструктор;
    - метод get_name — возвращает значение поля name.

Класс class_5:

- свойства/поля:
    - поле хранит текстовое сообщение:
        - наименование — name;
        - тип — string;
        - модификатор доступа — private;
- функционал:
    - метод class_5 — конструктор;
    - метод get_name — возвращает значение поля name.

Класс class_6:

- свойства/поля:
    - поле хранит текстовое сообщение:

        - наименование — name;
        - тип — string;
        - модификатор доступа — private;
- функционал:
    - метод class_6 — конструктор;
    - метод get_name — возвращает значение поля name.

Класс class_7:

- свойства/поля:
    - поле хранит текстовое сообщение:
        - наименование — name;
        - тип — string;
        - модификатор доступа — private;
- функционал:
    - метод class_7 — конструктор;
    - метод get_name — возвращает значение поля name.

Класс class_8:

- свойства/поля:
    - поле хранит текстовое сообщение:
        - наименование — name;
        - тип — string;
        - модификатор доступа — private;
- функционал:
    - метод class_8 — конструктор;
    - метод get_name — возвращает значение поля name.

Таблица 1 – Иерархия наследования классов

№	Имя класса	Классы-наследники	Модификатор доступа при наследовании	Описание	Номер
1	class_1			базовый класс	
		class_2	public		2
		class_3	public		3
		class_4	virtual public		4
		class_5	virtual public		5
2	class_2				
		class_6	public		6
3	class_3				
		class_6	public		6
4	class_4				
		class_7	public		7
5	class_5				
		class_7	public		7
6	class_6				
		class_8	public		8
7	class_7				
		class_8	public		8
8	class_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 class_1

Функционал: конструктор.

Параметры: string, name, имя объекта.

Алгоритм конструктора представлен в таблице 2.

Таблица 2 – Алгоритм конструктора класса class_1

№	Предикат	Действия	№ перехода
1		значение поля name текущего объекта становиться равным name + _1	∅

3.2 Алгоритм метода get_name класса class_1

Функционал: вывод значения поля name.

Параметры: нет.

Возвращаемое значение: string, имя объекта.

Алгоритм метода представлен в таблице 3.

Таблица 3 – Алгоритм метода get_name класса class_1

№	Предикат	Действия	№ перехода
1		возврат значения поля name	∅

3.3 Алгоритм конструктора класса class_2

Функционал: конструктор.

Параметры: string, name, имя объекта.

Алгоритм конструктора представлен в таблице 4.

Таблица 4 – Алгоритм конструктора класса class_2

№	Предикат	Действия	№ перехода
1		значение поля name текущего объекта становиться равным name + _2	∅

3.4 Алгоритм метода get_name класса class_2

Функционал: вывод значения поля name.

Параметры: нет.

Возвращаемое значение: string, имя объекта.

Алгоритм метода представлен в таблице 5.

Таблица 5 – Алгоритм метода get_name класса class_2

№	Предикат	Действия	№ перехода
1		возврат значения поля name	∅

3.5 Алгоритм конструктора класса class_3

Функционал: конструктор.

Параметры: string, name, имя объекта.

Алгоритм конструктора представлен в таблице 6.

Таблица 6 – Алгоритм конструктора класса class_3

№	Предикат	Действия	№ перехода
1		значение поля name текущего объекта становиться равным name + _3	∅

3.6 Алгоритм метода get_name класса class_3

Функционал: вывод значения поля name.

Параметры: нет.

Возвращаемое значение: string, имя объекта.

Алгоритм метода представлен в таблице 7.

Таблица 7 – Алгоритм метода get_name класса class_3

№	Предикат	Действия	№ перехода
1		возврат значения поля name	∅

3.7 Алгоритм конструктора класса class_4

Функционал: конструктор.

Параметры: string, name, имя объекта.

Алгоритм конструктора представлен в таблице 8.

Таблица 8 – Алгоритм конструктора класса class_4

№	Предикат	Действия	№ перехода
1		значение поля name текущего объекта становиться равным name + _4	∅

3.8 Алгоритм метода get_name класса class_4

Функционал: вывод значения поля name.

Параметры: нет.

Возвращаемое значение: string, имя объекта.

Алгоритм метода представлен в таблице 9.

Таблица 9 – Алгоритм метода get_name класса class_4

№	Предикат	Действия	№ перехода
1		возврат значения поля name	∅

3.9 Алгоритм конструктора класса class_5

Функционал: конструтктор.

Параметры: string, name, имя объекта.

Алгоритм конструктора представлен в таблице 10.

Таблица 10 – Алгоритм конструктора класса class_5

№	Предикат	Действия	№ перехода
1		значение поля name текущего объекта становиться равным name + _5	∅

3.10 Алгоритм метода get_name класса class_5

Функционал: вывод значения поля name.

Параметры: нет.

Возвращаемое значение: string, имя объекта.

Алгоритм метода представлен в таблице 11.

Таблица 11 – Алгоритм метода get_name класса class_5

№	Предикат	Действия	№ перехода
1		возврат значения поля name	∅

3.11 Алгоритм конструктора класса class_6

Функционал: конструктор.

Параметры: string, name, имя объекта.

Алгоритм конструктора представлен в таблице 12.

Таблица 12 – Алгоритм конструктора класса class_6

№	Предикат	Действия	№ перехода
1		значение поля name текущего объекта становиться равным name + _6	∅

3.12 Алгоритм метода get_name класса class_6

Функционал: вывод значения поля name.

Параметры: нет.

Возвращаемое значение: string, имя объекта.

Алгоритм метода представлен в таблице 13.

Таблица 13 – Алгоритм метода get_name класса class_6

№	Предикат	Действия	№ перехода
1		возврат значения поля name	∅

3.13 Алгоритм конструктора класса class_7

Функционал: конструктор.

Параметры: string, name, имя объекта.

Алгоритм конструктора представлен в таблице 14.

Таблица 14 – Алгоритм конструктора класса class_7

№	Предикат	Действия	№ перехода
1		значение поля name текущего объекта становиться равным name + _7	∅

3.14 Алгоритм метода get_name класса class_7

Функционал: вывод значения поля name.

Параметры: нет.

Возвращаемое значение: string, имя объекта.

Алгоритм метода представлен в таблице 15.

Таблица 15 – Алгоритм метода get_name класса class_7

№	Предикат	Действия	№ перехода
1		возврат значения поля name	∅

3.15 Алгоритм конструктора класса class_8

Функционал: конструктор.

Параметры: string, name, имя объекта.

Алгоритм конструктора представлен в таблице 16.

Таблица 16 – Алгоритм конструктора класса class_8

№	Предикат	Действия	№ перехода
1		значение поля name текущего объекта становиться равным name + _8	∅

3.16 Алгоритм метода get_name класса class_8

Функционал: вывод значения поля name.

Параметры: нет.

Возвращаемое значение: string, имя объекта.

Алгоритм метода представлен в таблице 17.

Таблица 17 – Алгоритм метода get_name класса class_8

№	Предикат	Действия	№ перехода
1		возврат значения поля name	∅

3.17 Алгоритм функции main

Функционал: выполнение поставленной задачи.

Параметры: нет.

Возвращаемое значение: int - код успешности выполнения программы.

Алгоритм функции представлен в таблице 18.

Таблица 18 – Алгоритм функции main

№	Предикат	Действия	№ перехода
1		объявление строковой переменной id	2
2		ввод id	3
3		инициализация указателя на объект object класса class_8	4
4		вывод результата применения метода get_name для object явно приведенным к class_1, class_2, class_6	5
5		вывод результата применения метода get_name для object явно приведенным к class_1, class_3, class_6	6
6		вывод результата применения метода get_name для object явно приведенным к class_1, class_4, class_7	7
7		вывод результата применения метода get_name для object явно приведенным к class_1, class_5, class_7	8
8		вывод результата применения метода get_name для object явно приведенным к class_2, class_6	9
9		вывод результата применения метода get_name для object явно приведенным к class_3, class_6	10
10		вывод результата применения метода get_name для object явно приведенным к class_4, class_7	11
11		вывод результата применения метода get_name для object явно приведенным к class_5, class_7	12
12		вывод результата применения метода get_name для object явно приведенным к class_6	13
13		вывод результата применения метода get_name для object явно приведенным к class_7	14
14		вывод результата применения метода get_name для object	15

№	Предикат	Действия	№ перехода
15		возврат 0	∅

4 БЛОК-СХЕМЫ АЛГОРИТМОВ

Представим описание алгоритмов в графическом виде на рисунках 1-10.

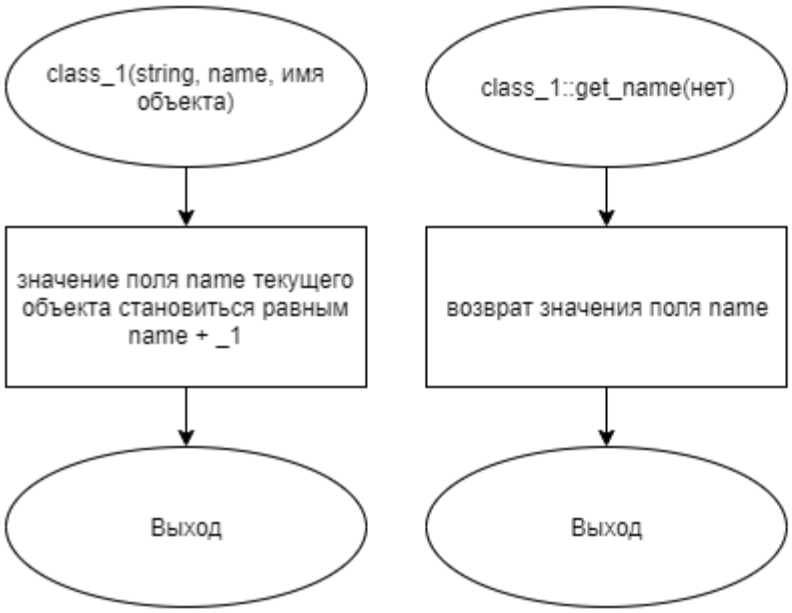

Рисунок 1 – Блок-схема алгоритма

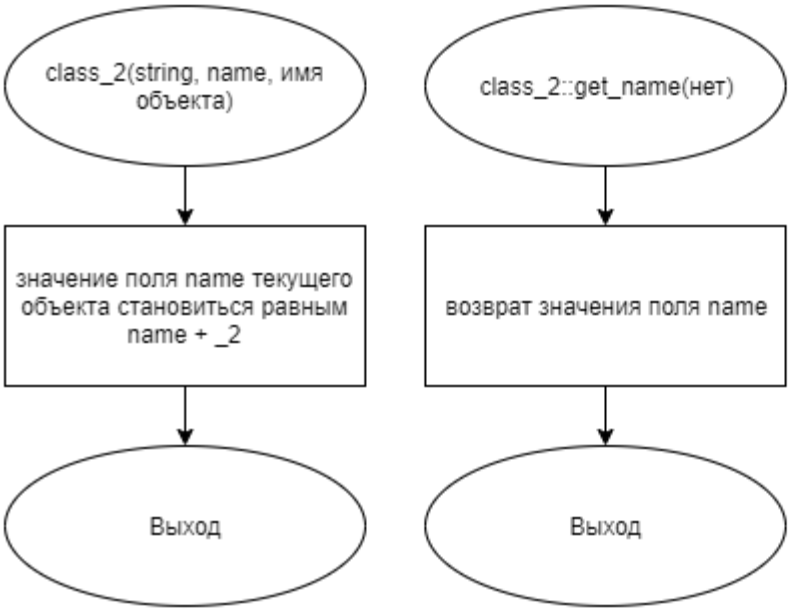

Рисунок 2 – Блок-схема алгоритма

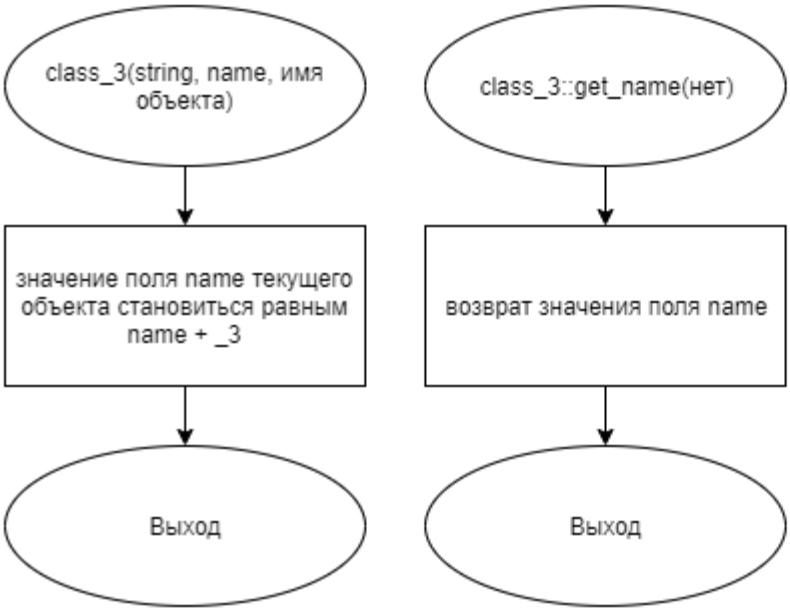

Рисунок 3 – Блок-схема алгоритма

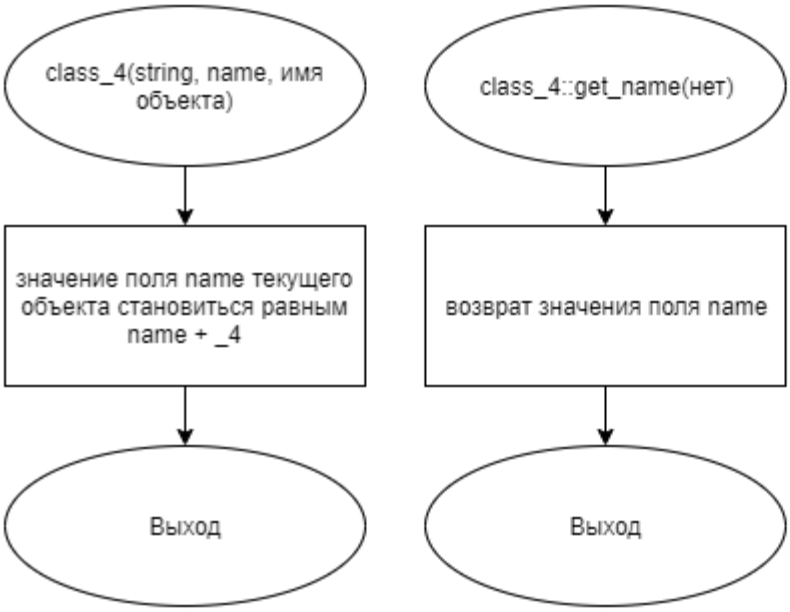

Рисунок 4 – Блок-схема алгоритма

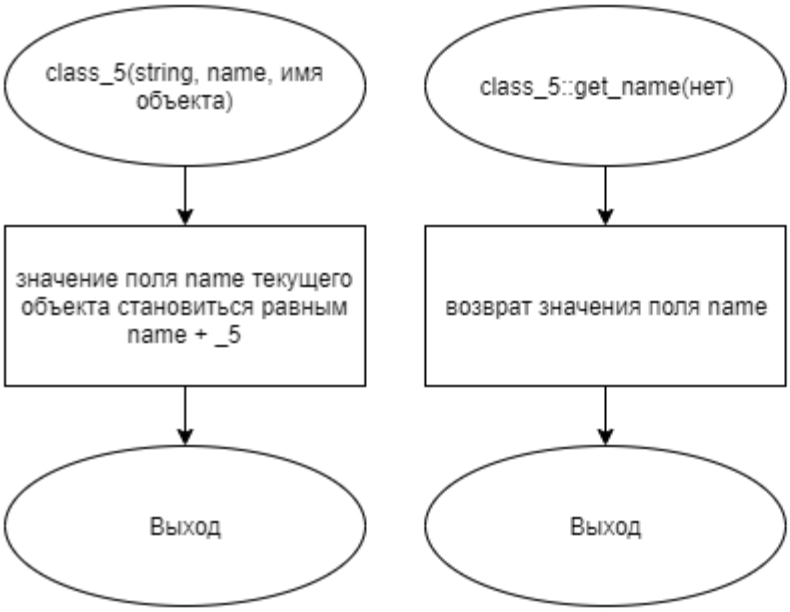

Рисунок 5 – Блок-схема алгоритма

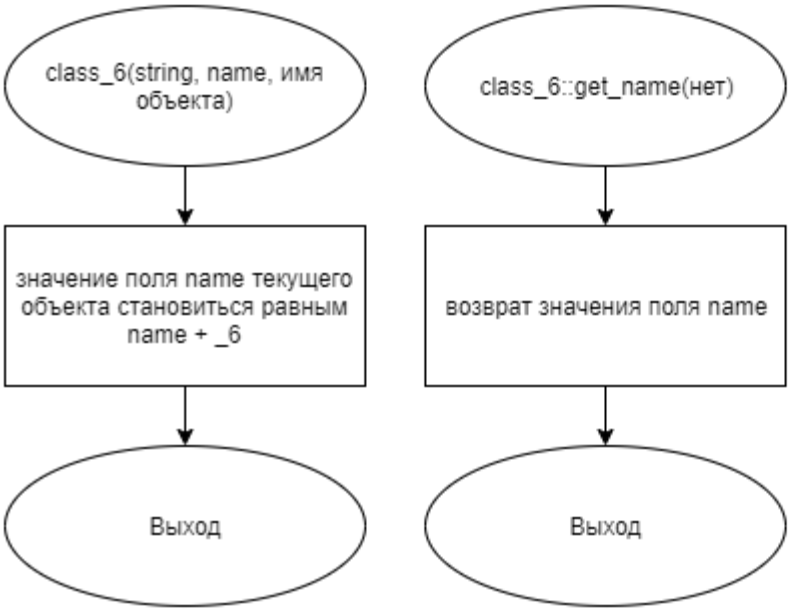

Рисунок 6 – Блок-схема алгоритма

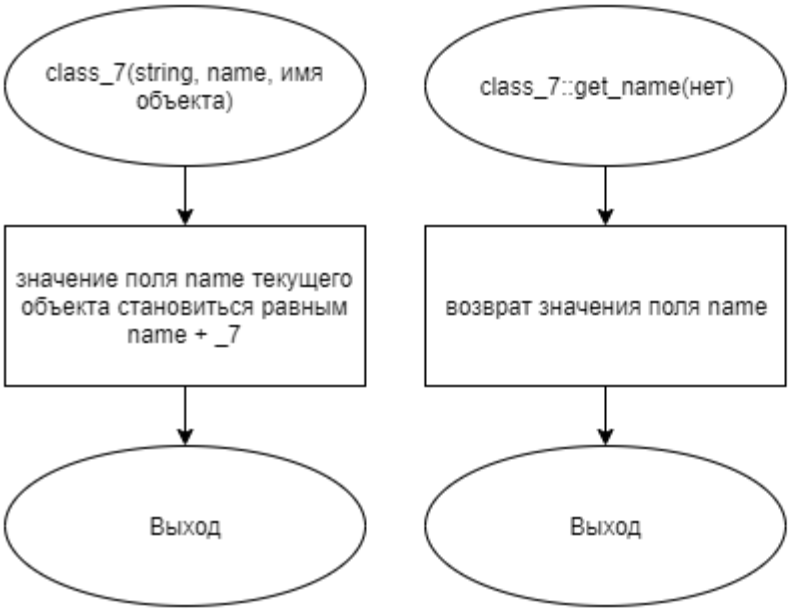

Рисунок 7 – Блок-схема алгоритма

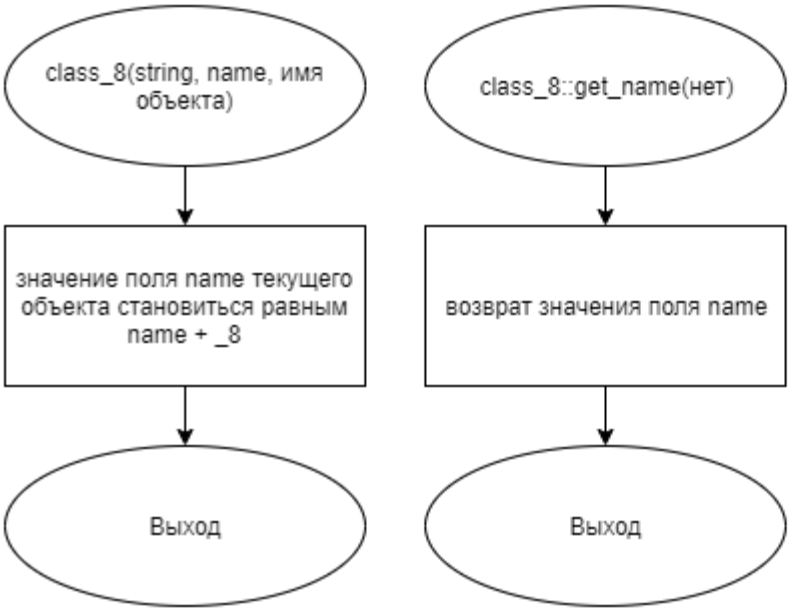

Рисунок 8 – Блок-схема алгоритма

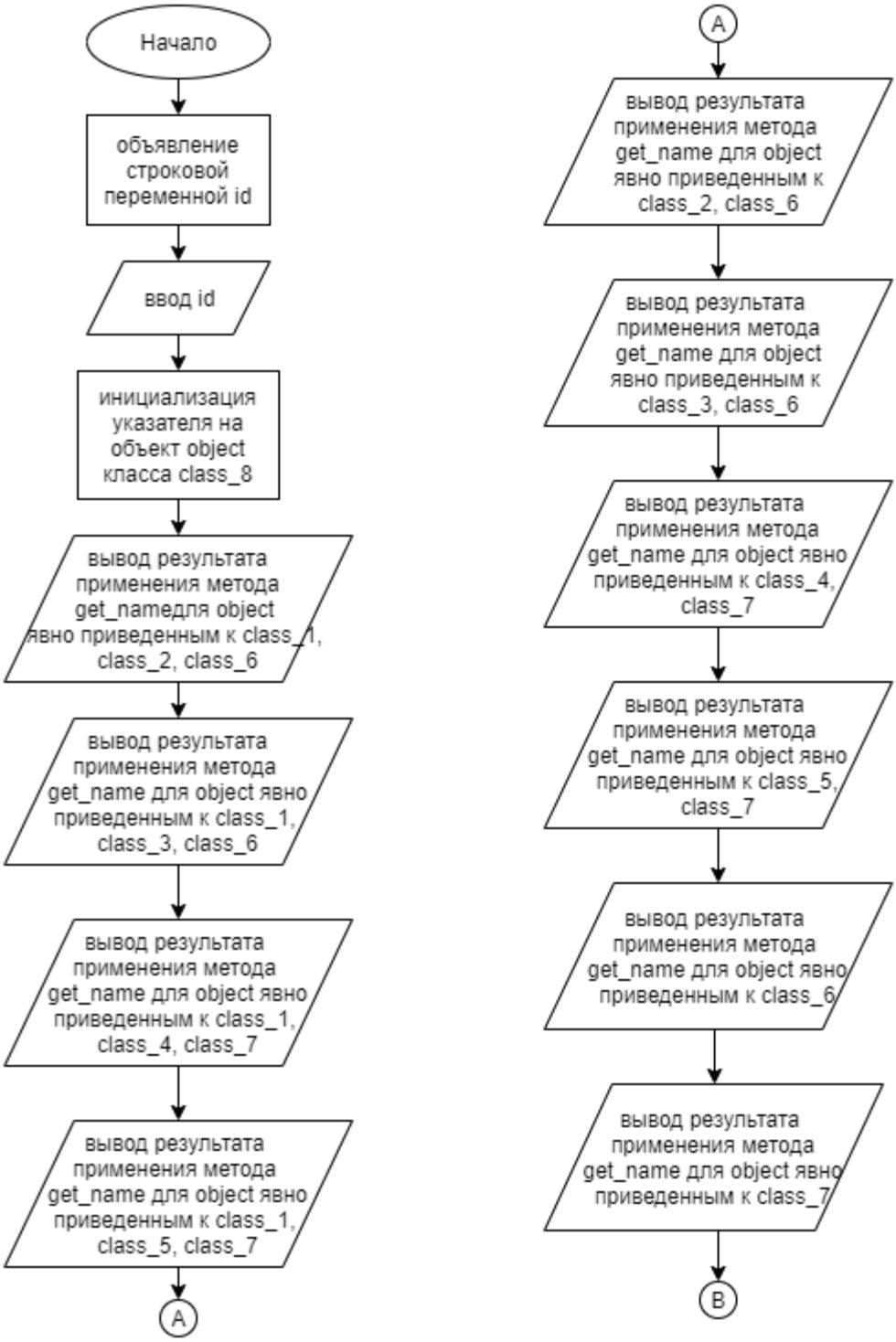

Рисунок 9 – Блок-схема алгоритма

Рисунок 10 – 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(std::string name){
   this->name = name + "_1";
}
std::string class_1::get_name(){
   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>

class class_1{
public:
   class_1(std::string name);
   std::string get_name();

private:
   std::string name;
};

#endif
```

5.3 Файл class_2.cpp

Листинг 3 – class_2.cpp

```
#include "class_2.h"

class_2::class_2(std::string name) : class_1(name + "_2"){
   this->name = name + "_2";
}
std::string class_2::get_name(){
   return name;
}
```

5.4 Файл class_2.h

Листинг 4 – class_2.h

```
#ifndef __CLASS_2__H
#define __CLASS_2__H
#include "class_1.h"

class class_2 : public class_1{
public:
   class_2(std::string name);
   std::string get_name();

private:
   std::string name;
};

#endif
```

5.5 Файл class_3.cpp

Листинг 5 – class_3.cpp

```
#include "class_3.h"

class_3::class_3(std::string name) : class_1(name + "_3"){
   this->name = name + "_3";
}
std::string class_3::get_name(){
   return name;
```

```
}
```

5.6 Файл class_3.h

Листинг 6 – class_3.h

```
#ifndef __CLASS_3__H
#define __CLASS_3__H
#include "class_2.h"

class class_3 : public class_1{
public:
   class_3(std::string name);
   std::string get_name();

private:
   std::string name;
};
#endif
```

5.7 Файл class_4.cpp

Листинг 7 – class_4.cpp

```
#include "class_4.h"

class_4::class_4(std::string name) : class_1(name + "_4"){
   this->name = name + "_4";
}
std::string class_4::get_name(){
   return name;
}
```

5.8 Файл class_4.h

Листинг 8 – class_4.h

```
#ifndef __CLASS_4__H
#define __CLASS_4__H
```

```
#include "class_3.h"


class class_4 : public virtual class_1{
public:
   std::string get_name();
   class_4(std::string name);

private:
   std::string name;
};

#endif
```

5.9 Файл class_5.cpp

Листинг 9 – class_5.cpp

```
#include "class_5.h"

class_5::class_5(std::string name) : class_1(name + "_5"){
   this->name = name + "_5";
}
std::string class_5::get_name(){
   return name;
}
```

5.10 Файл class_5.h

Листинг 10 – class_5.h

```
#ifndef __CLASS_5__H
#define __CLASS_5__H
#include "class_4.h"

class class_5 : public virtual class_1{
public:
   std::string get_name();
   class_5(std::string name);

private:
   std::string name;
};

#endif
```

5.11 Файл class_6.cpp

Листинг 11 – class_6.cpp

```
#include "class_6.h"

class_6::class_6(std::string name) : class_2(name + "_6"), class_3(name +
"_6"){
   this->name = name + "_6";
}
std::string class_6::get_name(){
   return name;
}
```

5.12 Файл class_6.h

Листинг 12 – class_6.h

```
#ifndef __CLASS_6__H
#define __CLASS_6__H
#include "class_5.h"

class class_6 : public class_2, public class_3{
public:
   class_6(std::string name);
   std::string get_name();

private:
   std::string name;
};

#endif
```

5.13 Файл class_7.cpp

Листинг 13 – class_7.cpp

```
#include "class_7.h"

class_7::class_7(std::string name) : class_4(name + "_7"), class_5(name +
```

```
"_7"), class_1(name + "_7"){
   this->name = name + "_7";
}
std::string class_7::get_name(){
   return name;
}
```

5.14 Файл class_7.h

Листинг 14 – class_7.h

```
#ifndef __CLASS_7__H
#define __CLASS_7__H
#include "class_6.h"

class class_7 : public class_4, public class_5{
public:
   std::string get_name();
   class_7(std::string name);
private:
   std::string name;
};

#endif
```

5.15 Файл class_8.cpp

Листинг 15 – class_8.cpp

```
#include "class_8.h"

class_8::class_8(std::string name) : class_6(name + "_8"), class_7(name +
"_8"), class_1(name + "_8"){
   this->name = name + "_8";
}
std::string class_8::get_name(){
   return name;
}
```

5.16 Файл class_8.h

Листинг 16 – class_8.h

```
#ifndef __CLASS_8__H
#define __CLASS_8__H
#include "class_7.h"

class class_8 : public class_6, public class_7{
public:
   std::string get_name();
   class_8(std::string name);

private:
   std::string name;
};

#endif
```

5.17 Файл main.cpp

Листинг 17 – main.cpp

```
#include <stdlib.h>
#include <stdio.h>
#include <iostream>
#include "class_8.h"

int main()
{
   std::string id;
   std::cin >> id;
   class_8* object = new class_8(id);
   std::cout << ((class_1*)((class_2*)(class_6*)object))->get_name() << "\n";
   std::cout << ((class_1*)((class_3*)(class_6*)object))->get_name() << "\n";
   std::cout << ((class_1*)((class_4*)(class_7*)object))->get_name() << "\n";
   std::cout << ((class_1*)((class_5*)(class_7*)object))->get_name() << "\n";
   std::cout << ((class_2*)(class_6*)object)->get_name() << "\n";
   std::cout << ((class_3*)(class_6*)object)->get_name() << "\n";
   std::cout << ((class_4*)(class_7*)object)->get_name() << "\n";
   std::cout << ((class_5*)(class_7*)object)->get_name() << "\n";
   std::cout << ((class_6*)object)->get_name() << "\n";
   std::cout << ((class_7*)object)->get_name() << "\n";
   std::cout << (object)->get_name();
   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).