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-27-21	Шевелёв И.А.
Руководитель практики	Ассистент	Морозов В.А.
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

Основная программа:

Указатель

Строковый тип данных

класс class_8

Объекты ввода/вывода потока данных(cin/cout библиотеки <iostream>)

Объект object, pObject класса class_8

Класс class_1

Модификатор доступа public, private

Конструктор

Метод getname

Объекты ввода/вывода потока данных(cin/cout библиотеки <iostream>)

Класс class_2

Модификатор доступа public, private

Конструктор

Метод getname

Объекты ввода/вывода потока данных(cin/cout библиотеки <iostream>)

Класс class_3

Модификатор доступа public, private

Конструктор

Метод getname

Объекты ввода/вывода потока данных(cin/cout библиотеки <iostream>)

Класс class_4

Модификатор доступа public, private

Конструктор

Метод getname

Объекты ввода/вывода потока данных(cin/cout библиотеки <iostream>)

Класс class_5

Модификатор доступа public, private

Конструктор

Метод getname

Объекты ввода/вывода потока данных(cin/cout библиотеки <iostream>)

Класс class_6

Модификатор доступа public, private

Конструктор

Метод getname

Объекты ввода/вывода потока данных(cin/cout библиотеки <iostream>)

Класс class_7

Модификатор доступа public, private

Конструктор

Метод getname

Объекты ввода/вывода потока данных(cin/cout библиотеки <iostream>)

Класс class_8

Модификатор доступа public, private

Конструктор

Метод getname

Объекты ввода/вывода потока данных(cin/cout библиотеки <iostream>)

Описание алгоритма

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

Конструктор класса: class_1

Модификатор доступа: public

Функционал: Конструктор класса

Параметры: string name

Алгоритм конструктора представлен в таблице 1.

Таблица 1. Алгоритм конструктора класса class_1

№	Предикат	Действия	№ перехода	Комментарий
1		Присвоение name текущего объекта значение name + "_1"	2	
2		Вызов метода getname	∅	

Класс объекта: class_1

Модификатор доступа: public

Метод: getname

Функционал: Возращает имя объекта

Параметры: нет

Возвращаемое значение: string name

Алгоритм метода представлен в таблице 2.

Таблица 2. Алгоритм метода getname класса class_1

№	Предикат	Действия	№ перехода	Комментарий
1		Возврат свойства name	∅	

Конструктор класса: class_2

Модификатор доступа: public

Функционал: Конструктор класса

Параметры: string name

Алгоритм конструктора представлен в таблице 3.

Таблица 3. Алгоритм конструктора класса class_2

№	Предикат	Действия	№ перехода	Комментарий
1		Присвоение name текущего объекта значение name + "_2"	2	
2		Вызов метода getname	∅	

Класс объекта: class_2

Модификатор доступа: public

Метод: getname

Функционал: Возращает имя объекта

Параметры: нет

Возвращаемое значение: string name

Алгоритм метода представлен в таблице 4.

Таблица 4. Алгоритм метода getname класса class_2

№	Предикат	Действия	№ перехода	Комментарий
1		Возрат свойства name	∅	

Конструктор класса: class_3

Модификатор доступа: public

Функционал: Конструктор класса

Параметры: string name

Алгоритм конструктора представлен в таблице 5.

Таблица 5. Алгоритм конструктора класса class_3

№	Предикат	Действия	№ перехода	Комментарий
1		Присвоение name текущего объекта значение name + "_3"	2	
2		Вызов метода getname	∅	

Класс объекта: class_3

Модификатор доступа: public

Метод: getname

Функционал: Возращает имя объекта

Параметры: нет

Возвращаемое значение: string name

Алгоритм метода представлен в таблице 6.

Таблица 6. Алгоритм метода getname класса class_3

№	Предикат	Действия	№ перехода	Комментарий
1		Возрат свойства name	∅	

Конструктор класса: class_4

Модификатор доступа: public

Функционал: Конструктор класса

Параметры: string name

Алгоритм конструктора представлен в таблице 7.

Таблица 7. Алгоритм конструктора класса class_4

№	Предикат	Действия	№ перехода	Комментарий
1		Присвоение name текущего объекта значение name + "_4"	2	
2		Вызов метода getname	∅	

Класс объекта: class_4

Модификатор доступа: public

Метод: getname

Функционал: Возращает имя объекта

Параметры: нет

Возвращаемое значение: string name

Алгоритм метода представлен в таблице 8.

Таблица 8. Алгоритм метода getname класса class_4

№	Предикат	Действия	№ перехода	Комментарий
1		Возрат свойства name	∅	

Конструктор класса: class_5

Модификатор доступа: public

Функционал: Конструктор класса

Параметры: string name

Алгоритм конструктора представлен в таблице 9.

Таблица 9. Алгоритм конструктора класса class_5

№	Предикат	Действия	№ перехода	Комментарий
1		Присвоение name текущего объекта значение name + "_5"	2	
2		Вызов метода getname	∅	

Класс объекта: class_5

Модификатор доступа: public

Метод: getname

Функционал: Возращает имя объекта

Параметры: нет

Возвращаемое значение: string name

Алгоритм метода представлен в таблице 10.

Таблица 10. Алгоритм метода getname класса class_5

№	Предикат	Действия	№ перехода	Комментарий
1		Возрат свойства name	∅	

Конструктор класса: class_6

Модификатор доступа: public

Функционал: Конструктор класса

Параметры: string name

Алгоритм конструктора представлен в таблице 11.

Таблица 11. Алгоритм конструктора класса class_6

№	Предикат	Действия	№ перехода	Комментарий
1		Присвоение name текущего объекта значение name + "_6"	2	
2		Вызов метода getname	∅	

Класс объекта: class_6

Модификатор доступа: public

Метод: getname

Функционал: Возращает имя объекта

Параметры: нет

Возвращаемое значение: string name

Алгоритм метода представлен в таблице 12.

Таблица 12. Алгоритм метода getname класса class_6

№	Предикат	Действия	№ перехода	Комментарий
1		Возрат значение name	∅	

Конструктор класса: class_7

Модификатор доступа: public

Функционал: Конструктор класса

Параметры: string name

Алгоритм конструктора представлен в таблице 13.

Таблица 13. Алгоритм конструктора класса class_7

№	Предикат	Действия	№ перехода	Комментарий
1		Присвоение name текущего объекта значение name + "_7"	2	
2		Вызов метода getname	∅	

Класс объекта: class_7

Модификатор доступа: public

Метод: getname

Функционал: Возращает имя объекта

Параметры: нет

Возвращаемое значение: string name

Алгоритм метода представлен в таблице 14.

Таблица 14. Алгоритм метода getname класса class_7

№	Предикат	Действия	№ перехода	Комментарий
1		Возрат свойства name	∅	

Конструктор класса: class_8

Модификатор доступа: public

Функционал: Конструктор класса

Параметры: string name

Алгоритм конструктора представлен в таблице 15.

Таблица 15. Алгоритм конструктора класса class_8

№	Предикат	Действия	№ перехода	Комментарий
1		Присвоение name текущего объекта значение name + "_8"	2	
2		Вызом метода getname	∅	

Класс объекта: class_8

Модификатор доступа: public

Метод: getname

Функционал: Возращает имя объекта

Параметры: нет

Возвращаемое значение: string name

Алгоритм метода представлен в таблице 16.

Таблица 16. Алгоритм метода getname класса class_8

№	Предикат	Действия	№ перехода	Комментарий
1		Возрат свойства name	∅	

Функция: main

Функционал: Главная функция программы

Параметры: нет

Возвращаемое значение: int, Код возрата (успешное выполнение 0)

Алгоритм функции представлен в таблице 17.

Таблица 17. Алгоритм функции main

№	Предикат	Действия	№ перехода	Комментарий
1		Объявление указателя pObject класса class_8	2	
2		Объявление переменной name строкового типа	3	
3		Ввод значения name	4	
4		Присвоение указателю выделенную память под объект класса class_8	5	
5		Вызов метода getname класса class_1	6	
6		Вызов метода getname класса class_1	7	
7		Вызов метода getname класса class_1	8	
8		Вызов метода getname класса class_2	9	
9		Вызов метода getname класса class_3	10	
10		Вызов метода getname класса class_4	11	
11		Вызов метода getname класса class_5	12	
12		Вызов метода getname класса class_6	13	
13		Вызов метода getname класса class_7	14	
14		Вызов метода getname класса class_8	∅	

Блок-схема алгоритма

Представим описание алгоритмов в графическом виде на рисунках ниже.

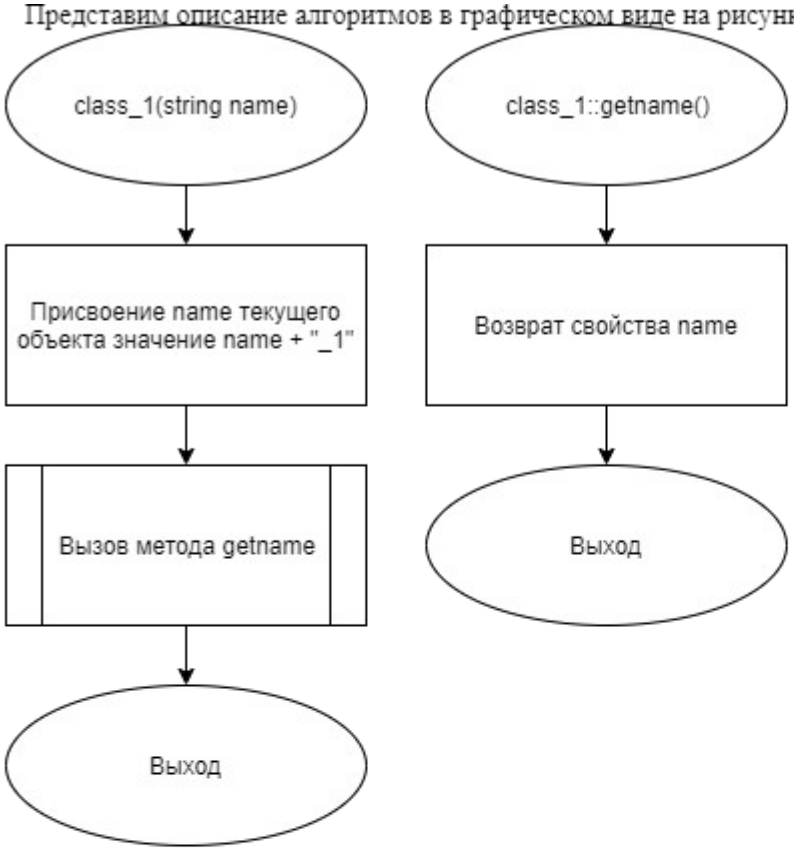

Рис. 1. Блок-схема алгоритма.

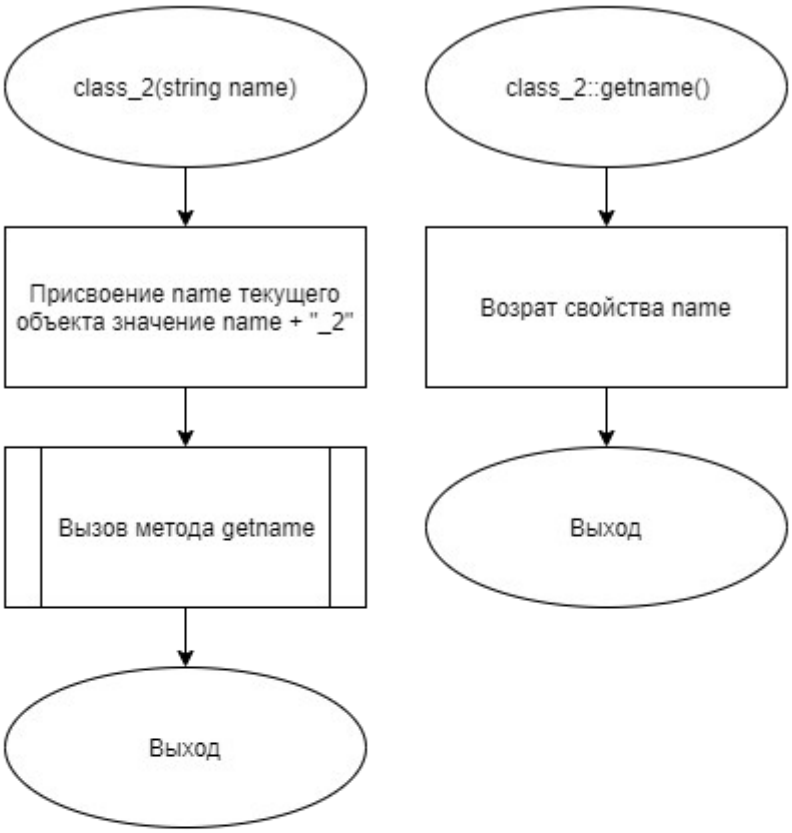

Рис. 2. Блок-схема алгоритма.

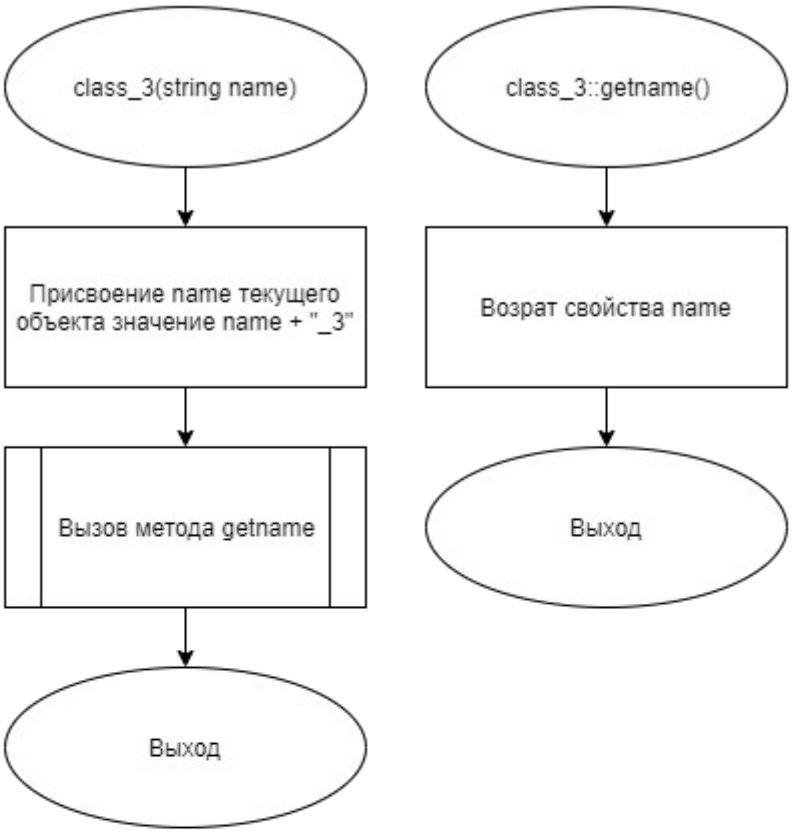

Рис. 3. Блок-схема алгоритма.

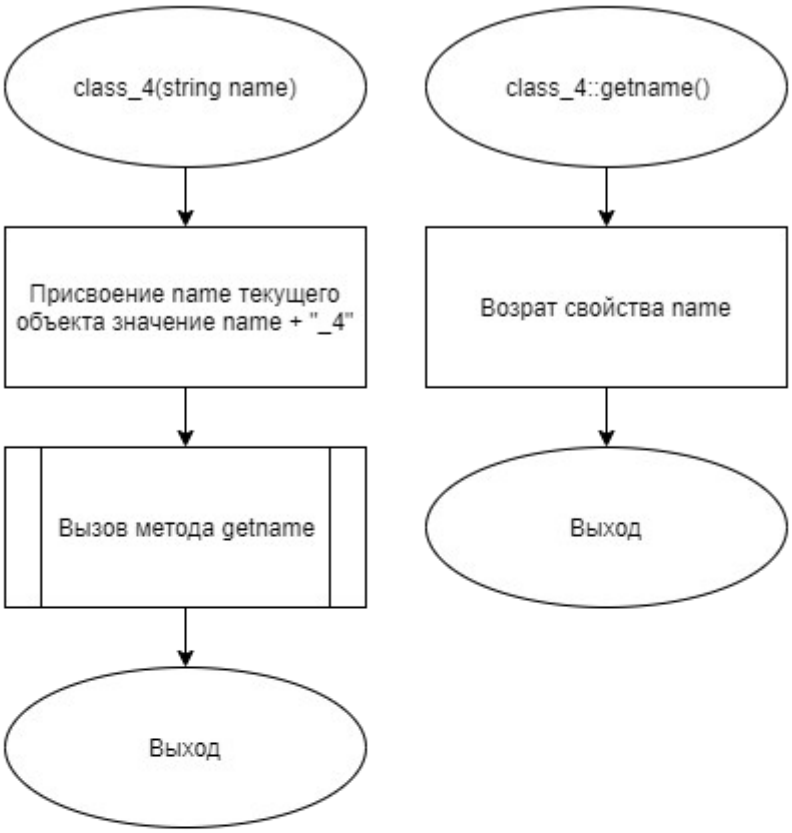

Рис. 4. Блок-схема алгоритма.

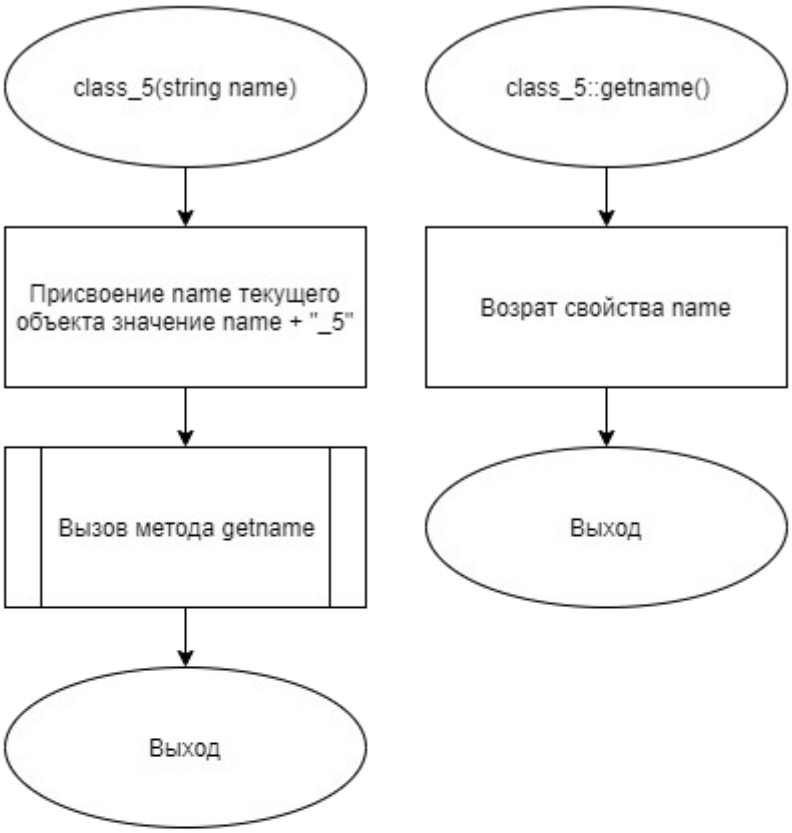

Рис. 5. Блок-схема алгоритма.

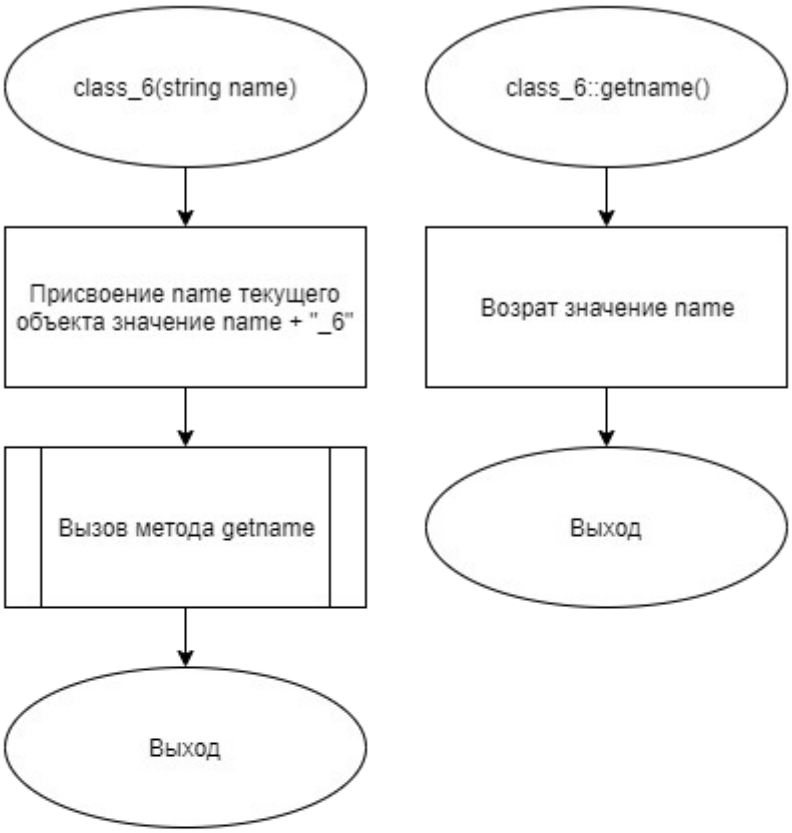

Рис. 6. Блок-схема алгоритма.

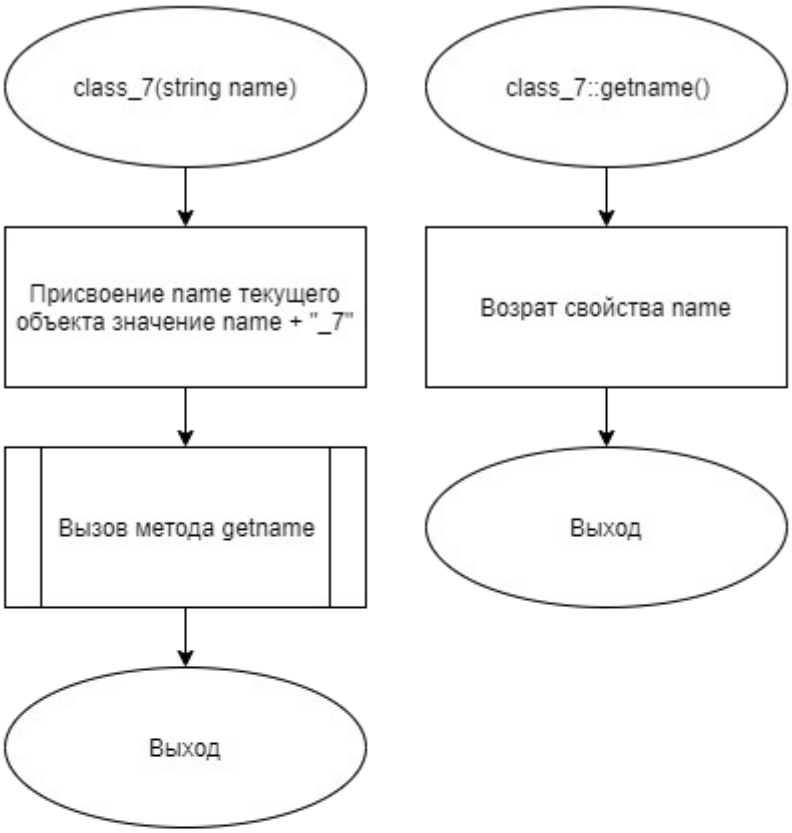

Рис. 7. Блок-схема алгоритма.

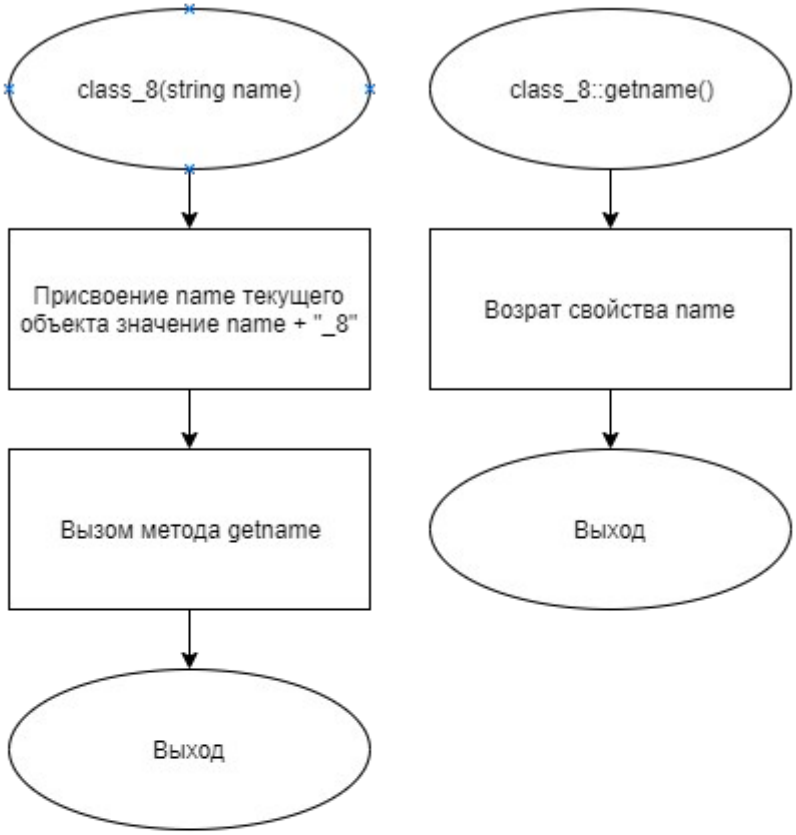

Рис. 8. Блок-схема алгоритма.

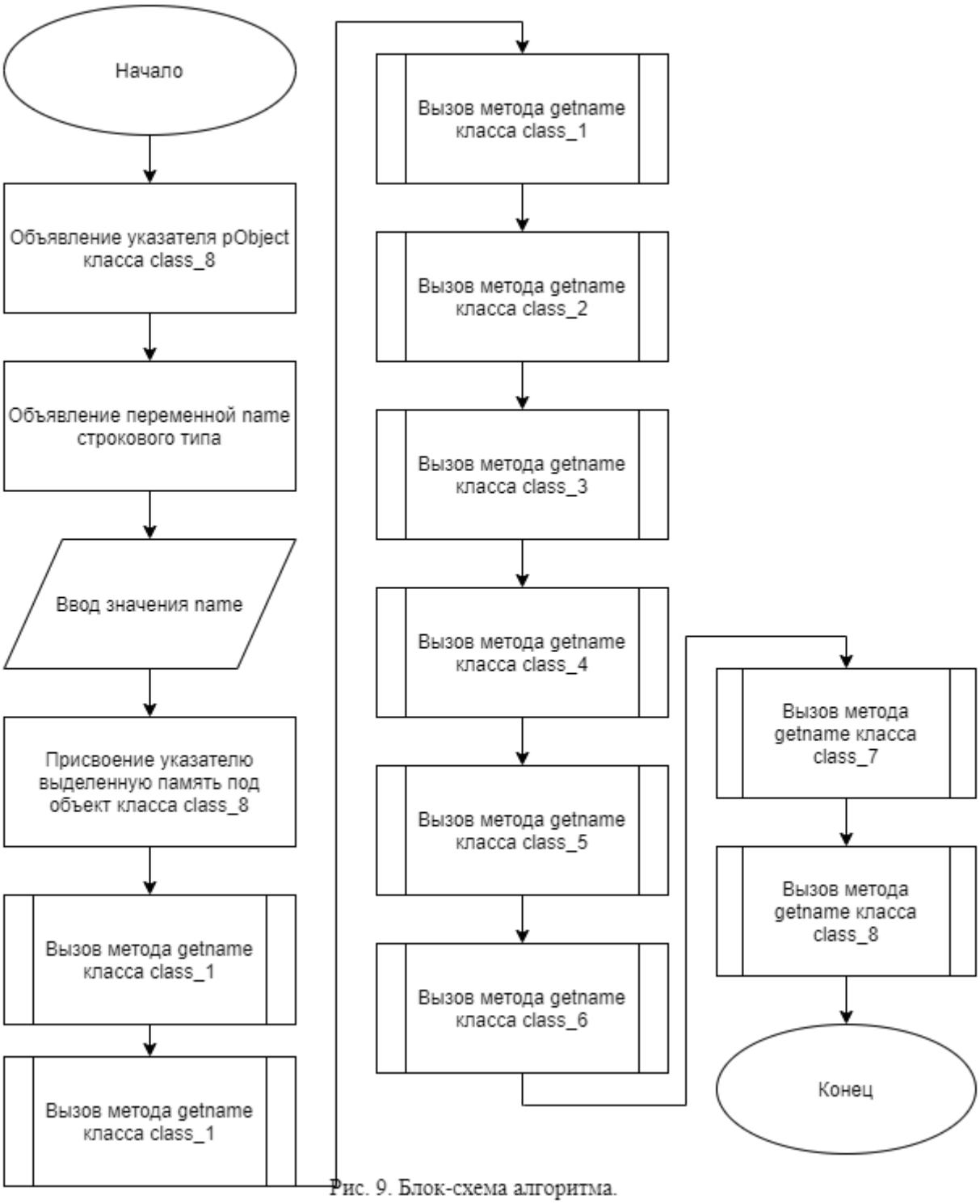

Рис. 9. Блок-схема алгоритма.

Код программы

Программная реализация алгоритмов для решения задачи представлена ниже.

Файл class_1.cpp

```
#include "class_1.h"

class_1::class_1(string name){
        this->name = name + "_1";
        getname();
}
string class_1::getname(){
        return name;
}
```

Файл class_1.h

```
#ifndef CLASS1_H
#define CLASS1_H
#include <string>
#include <iostream>
using namespace std;

class class_1{
private:
        string name;
public:
        class_1(string name);
        string getname();
};
#endif
```

Файл class_2.cpp

```
#include "class_2.h"

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

```
string class_2::getname(){
        return name;
}
```

Файл class_2.h

```
#ifndef CLASS2_H
#define CLASS2_H
#include "class_1.h"

class class_2 : virtual public class_1{
private:
        string name;
public:
        class_2(string name);
        string getname();
};
#endif
```

Файл class_3.cpp

```
#include "class_3.h"

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

Файл class_3.h

```
#ifndef CLASS3_H
#define CLASS3_H
#include "class_2.h"

class class_3 : virtual public class_1{
private:
        string name;
public:
        class_3(string name);
        string getname();
};
#endif
```

Файл class_4.cpp

```
#include "class_4.h"

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

Файл class_4.h

```
#ifndef CLASS4_H
#define CLASS4_H
#include "class_3.h"
class class_4 : virtual public class_1{
private:
        string name;
public:
        class_4(string name);
        string getname();
};
#endif
```

Файл class_5.cpp

```
#include "class_5.h"

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

Файл class_5.h

```
#ifndef CLASS5_H
#define CLASS5_H
#include "class_4.h"
class class_5 : virtual public class_1{
private:
        string name;
public:
        class_5(string name);
        string getname();
};
#endif
```

Файл class_6.cpp

```
#include "class_6.h"

class_6::class_6(string name) : class_2(name), class_3(name), class_1(name){
        this->name = name + "_6";
        getname();
}
string class_6::getname(){
        return name;
}
```

Файл class_6.h

```
#ifndef CLASS6_H
#define CLASS6_H
#include "class_5.h"
class class_6 : public class_2, public class_3{
private:
        string name;
public:
        class_6(string name);
        string getname();
};
#endif
```

Файл class_7.cpp

```
#include "class_7.h"

class_7::class_7(string name) : class_4(name), class_5(name), class_1(name){
        this->name = name + "_7";
        getname();
}
string class_7::getname(){
```

```
        return name;
}
```

Файл class_7.h

```
#ifndef CLASS7_H
#define CLASS7_H
#include "class_6.h"
class class_7 : public class_4, public class_5{
private:
        string name;
public:
        class_7(string name);
        string getname();
};
#endif
```

Файл class_8.cpp

```
#include "class_8.h"

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

Файл class_8.h

```
#ifndef CLASS8_H
#define CLASS8_H
#include "class_7.h"
class class_8 : public class_6, public class_7{
private:
        string name;
public:
        class_8(string name);
        string getname();
};
#endif
```

Файл main.cpp

```
#include <stdlib.h>
#include <stdio.h>
#include "class_8.h"
int main()
{
        // program here
        class_8* pObject;
        string  name;
        cin >> name;
        class_8* object = new class_8(name);
        pObject = object;

        cout << ((class_1*)pObject)->getname() << endl;
        cout << ((class_1*)pObject)->getname() << endl;
        cout << ((class_1*)pObject)->getname() << endl;
        cout << ((class_2*)pObject)->class_2::getname() << endl;
        cout << ((class_3*)pObject)->class_3::getname() << endl;
        cout << ((class_4*)pObject)->class_4::getname() << endl;
        cout << ((class_5*)pObject)->class_5::getname() << endl;
        cout << ((class_6*)pObject)->class_6::getname() << endl;
        cout << ((class_7*)pObject)->class_7::getname() << endl;
        cout << ((class_8*)pObject)->class_8::getname();

        return(0);
}
```

Тестирование

Результат тестирования программы представлен в следующей таблице.

Входные данные	Ожидаемые выходные данные	Фактические выходные данные
input	input_1 input_1 input_1 input_2 input_3 input_4 input_5 input_6 input_7 input_8	input_1 input_1 input_1 input_2 input_3 input_4 input_5 input_6 input_7 input_8
input	input_1 input_1 input_1 input_2 input_3 input_4 input_5 input_6 input_7 input_8	input_1 input_1 input_1 input_2 input_3 input_4 input_5 input_6 input_7 input_8
input	input_1 input_1 input_1 input_2 input_3 input_4 input_5 input_6 input_7 input_8	input_1 input_1 input_1 input_2 input_3 input_4 input_5 input_6 input_7 input_8
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

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).