Здесь будет титульник, листай ниже

СОДЕРЖАНИЕ

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 Class1.h предназначен для ;
- объект класса Class2.h предназначен для ;
- объект класса Class3.h предназначен для ;
- объект класса Class4.h предназначен для ;
- объект класса Class5.h предназначен для ;
- объект класса Class6.h предназначен для ;
- объект класса Class7.h предназначен для ;
- объект класса Class8.h предназначен для ;
- стандартная библиотека объектов ввода и вывода cin/cout;
- оператор присвоения =.

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 функции main

Функционал: основная функция.

Параметры: нет.

Возвращаемое значение: int.

Алгоритм функции представлен в таблице 1.

Таблица 1 – Алгоритм функции main

№	Предикат	Действия	№ перехода
1		объявление и ввод строки str	2
2		создание объекта obj класса Class8	3
3		вызов print у ref, присвоенного к Class1 через Class2	4
4		вызов print у ref, присвоенного к Class1 через Class2	5
5		вызов print у ref, присвоенного к Class1 через Class2	6
6		вызов print у ref, присвоенного к Class1 через Class2	7
7		вывод перехода на новую строку	8
8		возврат 0	∅

3.2 Алгоритм конструктора класса Class1

Функционал: создание объекта.

Параметры: нет.

Алгоритм конструктора представлен в таблице 2.

Таблица 2 – Алгоритм конструктора класса Class1

№	Предикат	Действия	№ перехода
1		присвоить полю name значение str	∅

3.3 Алгоритм конструктора класса Class2

Функционал: создание объекта, установовление полей.

Параметры: нет.

Алгоритм конструктора представлен в таблице 3.

Таблица 3 – Алгоритм конструктора класса Class2

№	Предикат	Действия	№ перехода
1		вызов конст. класса родителя Class1 с параметрром str+"_1"	2
2		присвоить полю name значение str	∅

3.4 Алгоритм конструктора класса Class3

Функционал: создание объекта, установовление полей.

Параметры: нет.

Алгоритм конструктора представлен в таблице 4.

Таблица 4 – Алгоритм конструктора класса Class3

№	Предикат	Действия	№ перехода
1		вызов конст. класса родителя Class1 с параметрром str+"_1"	2
2		присвоить полю name значение str	∅

3.5 Алгоритм конструктора класса Class4

Функционал: создание объекта, установовление полей.

Параметры: нет.

Алгоритм конструктора представлен в таблице 5.

Таблица 5 – Алгоритм конструктора класса Class4

№	Предикат	Действия	№ перехода
1		вызов конст. класса родителя Class1 с параметрром str+"_1"	2
2		присвоить полю name значение str	∅

3.6 Алгоритм конструктора класса Class5

Функционал: создание объекта, установовление полей.

Параметры: нет.

Алгоритм конструктора представлен в таблице 6.

Таблица 6 – Алгоритм конструктора класса Class5

№	Предикат	Действия	№ перехода
1		вызов конст. класса родителя Class1 с параметрром str+"_1"	2
2		присвоить полю name значение str	∅

3.7 Алгоритм конструктора класса Class6

Функционал: создание объекта, установовление полей.

Параметры: нет.

Алгоритм конструктора представлен в таблице 7.

Таблица 7 – Алгоритм конструктора класса Class6

№	Предикат	Действия	№ перехода
1		вызов конст. класса родителя Class1 с параметрром str+"_2", класса родителя Class3 с параметром str "_3"	2
2		присвоить полю name значение str	∅

3.8 Алгоритм конструктора класса Class7

Функционал: создание объекта, установовление полей.

Параметры: нет.

Алгоритм конструктора представлен в таблице 8.

Таблица 8 – Алгоритм конструктора класса Class7

№	Предикат	Действия	№ перехода
1		вызов конст. класса родителя Class1 с параметрром str+"_4", класса родителя Class5 с параметром str "_5"	2
2		присвоить полю name значение str	∅

3.9 Алгоритм конструктора класса Class8

Функционал: создание объекта, установовление полей.

Параметры: нет.

Алгоритм конструктора представлен в таблице 9.

Таблица 9 – Алгоритм конструктора класса Class8

№	Предикат	Действия	№ перехода
1		вызов конст. класса родителя Class1 с параметрром str+"_4", класса родителя Class5 с параметром str "_5"	2
2		присвоить полю name значение str	∅

3.10 Алгоритм метода print класса CLass1

Функционал: вывод наименования объекта.

Параметры: нет.

Возвращаемое значение: нет.

Алгоритм метода представлен в таблице 10.

Таблица 10 – Алгоритм метода print класса CLass1

№	Предикат	Действия	№ перехода
1		вывод name, переход на новую строку	∅

3.11 Алгоритм метода print класса Class2

Функционал: вывод наименования обкта.

Параметры: нет.

Возвращаемое значение: нет.

Алгоритм метода представлен в таблице 11.

Таблица 11 – Алгоритм метода print класса Class2

№	Предикат	Действия	№ перехода
1		вывод name	∅

3.12 Алгоритм метода print класса Class8

Функционал: вывод наименования обкта.

Параметры: нет.

Возвращаемое значение: нет.

Алгоритм метода представлен в таблице 12.

Таблица 12 – Алгоритм метода print класса Class8

№	Предикат	Действия	№ перехода
1		вывод name	∅

3.13 Алгоритм метода print класса Class3

Функционал: вывод наименования обкта.

Параметры: нет.

Возвращаемое значение: нет.

Алгоритм метода представлен в таблице 13.

Таблица 13 – Алгоритм метода print класса Class3

№	Предикат	Действия	№ перехода
1		вывод с новой строки name	∅

3.14 Алгоритм метода print класса Class4

Функционал: вывод наименования обкта.

Параметры: нет.

Возвращаемое значение: нет.

Алгоритм метода представлен в таблице 14.

Таблица 14 – Алгоритм метода print класса Class4

№	Предикат	Действия	№ перехода
1		вывод с новой строки name	∅

3.15 Алгоритм метода print класса Class5

Функционал: вывод наименования обкта.

Параметры: нет.

Возвращаемое значение: нет.

Алгоритм метода представлен в таблице 15.

Таблица 15 – Алгоритм метода print класса Class5

№	Предикат	Действия	№ перехода
1		вывод с новой строки name	∅

3.16 Алгоритм метода print класса Class6

Функционал: вывод наименования обкта.

Параметры: нет.

Возвращаемое значение: нет.

Алгоритм метода представлен в таблице 16.

Таблица 16 – Алгоритм метода print класса Class6

№	Предикат	Действия	№ перехода
1		вывод с новой строки name	∅

3.17 Алгоритм метода print класса Class7

Функционал: вывод наименования обкта.

Параметры: нет.

Возвращаемое значение: нет.

Алгоритм метода представлен в таблице 17.

Таблица 17 – Алгоритм метода print класса Class7

№	Предикат	Действия	№ перехода
1		вывод с новой строки name	∅

4 БЛОК-СХЕМЫ АЛГОРИТМОВ

Представим описание алгоритмов в графическом виде на рисунках 1-9.

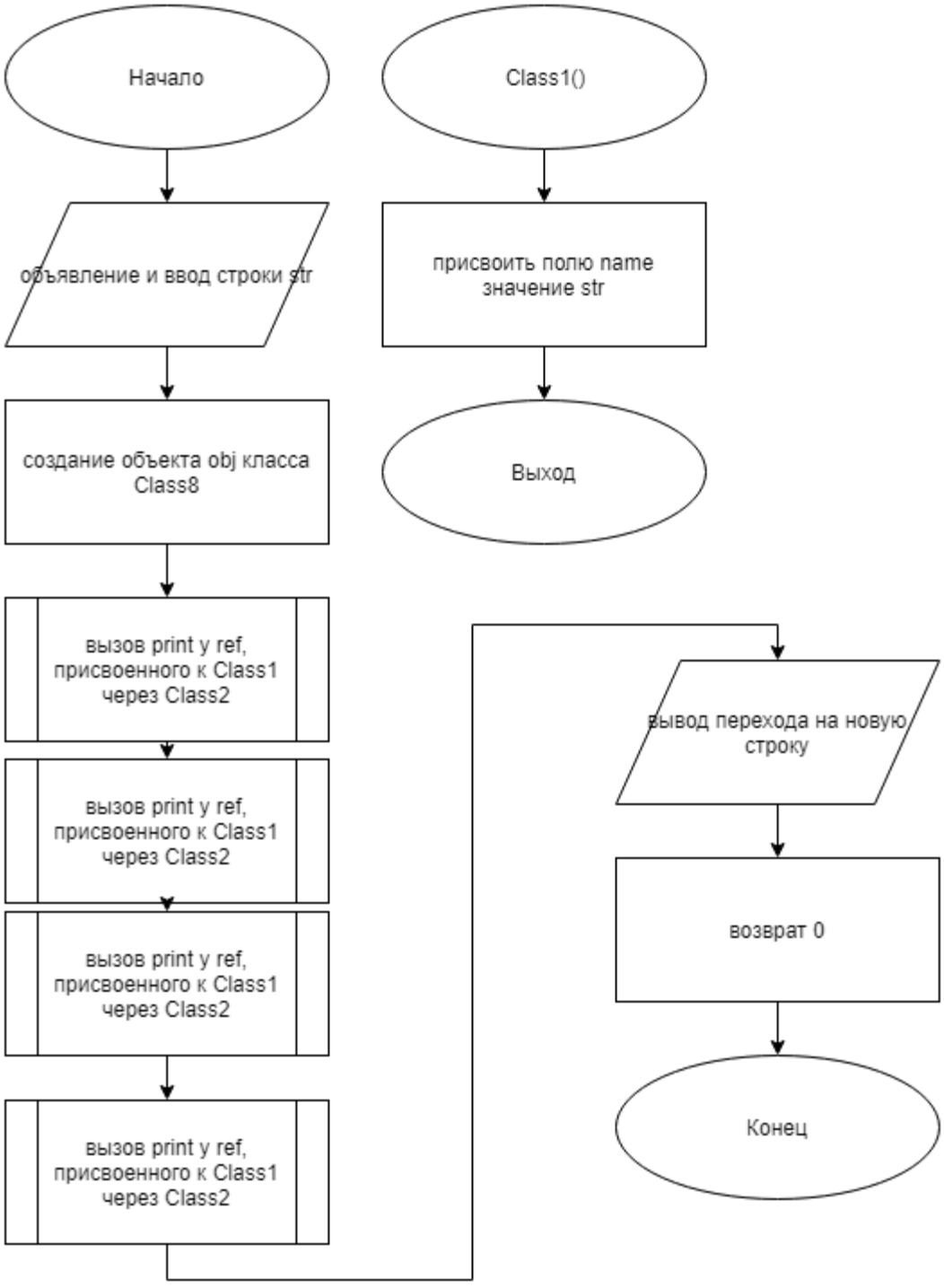

Рисунок 1 – Блок-схема алгоритма

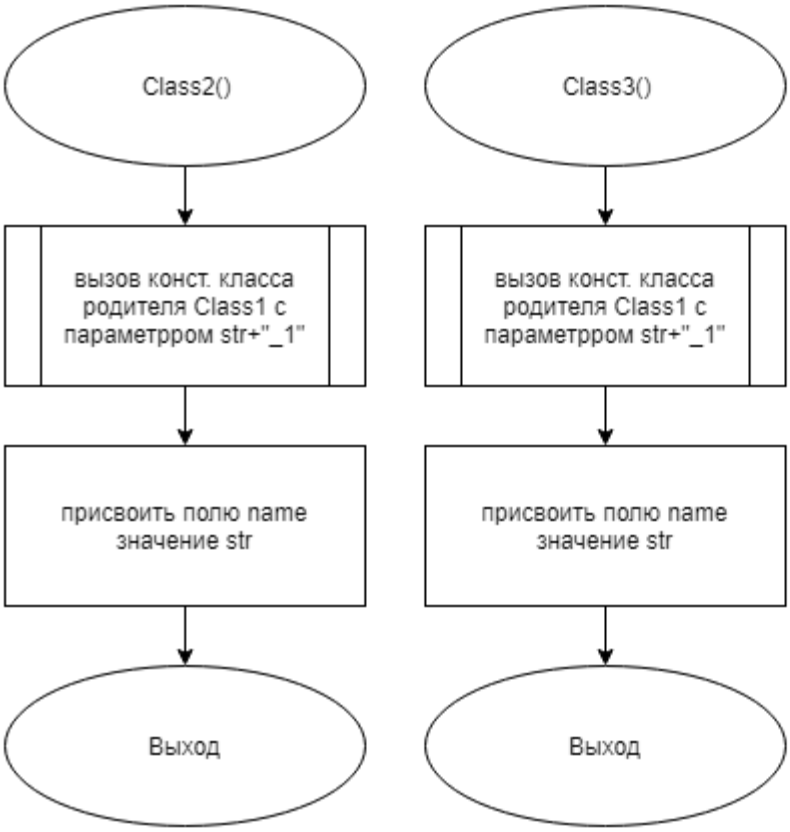

Рисунок 2 – Блок-схема алгоритма

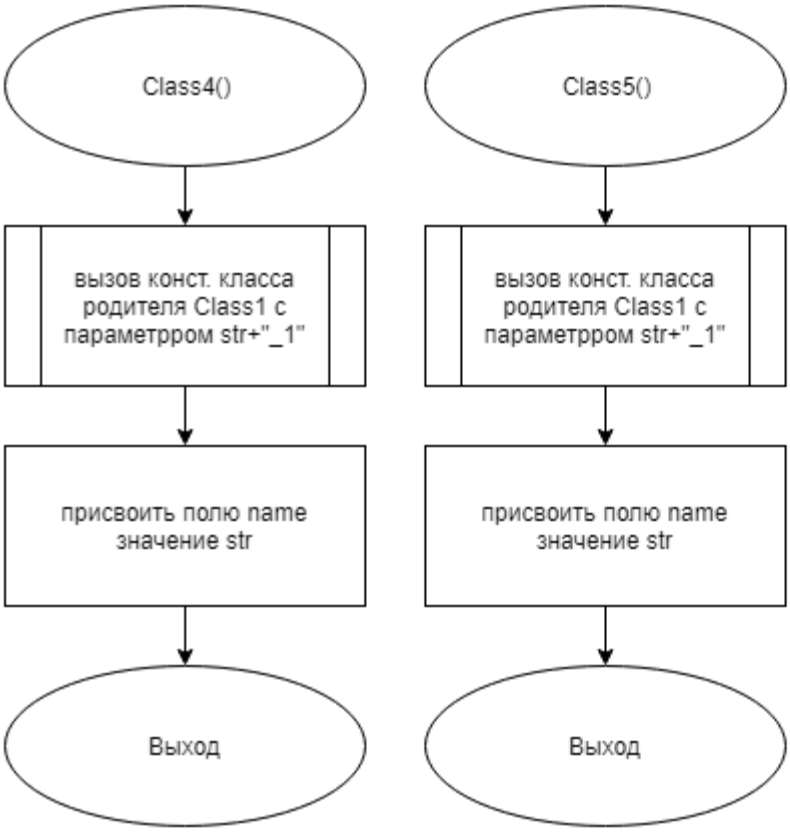

Рисунок 3 – Блок-схема алгоритма

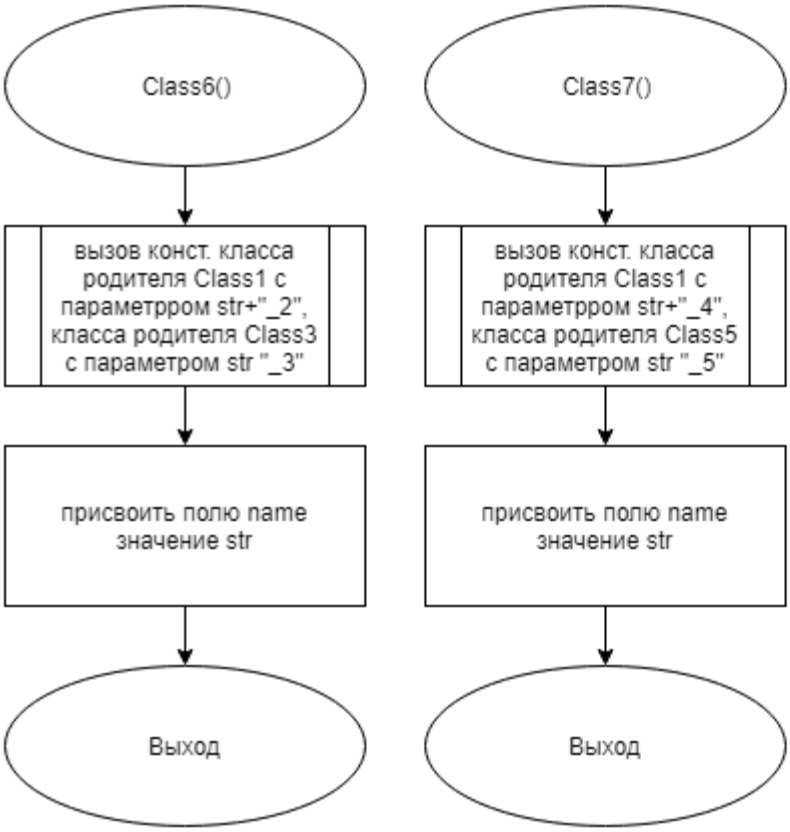

Рисунок 4 – Блок-схема алгоритма

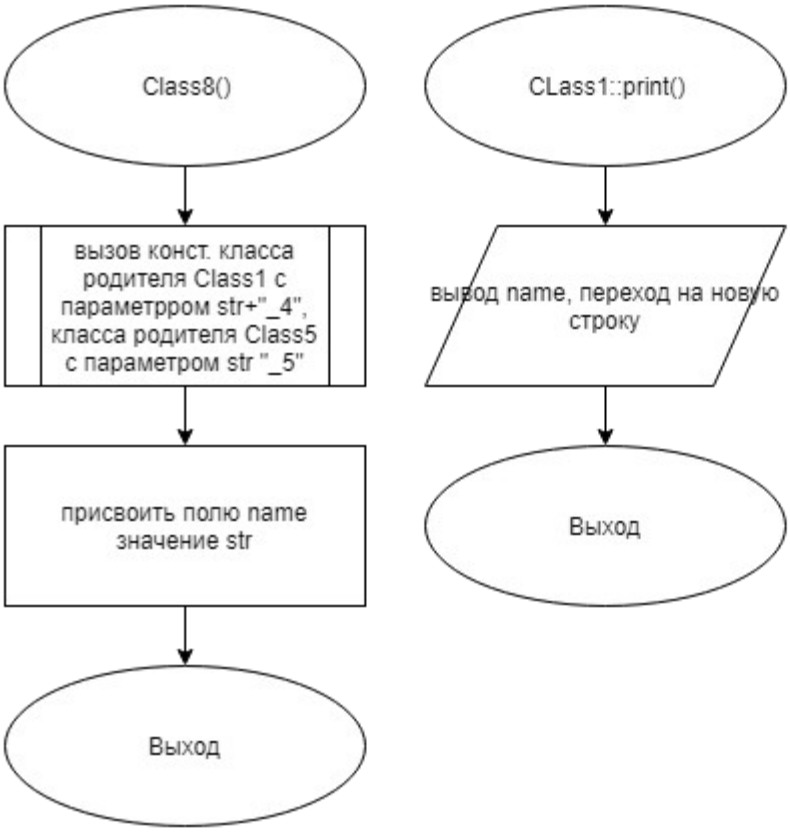

Рисунок 5 – Блок-схема алгоритма

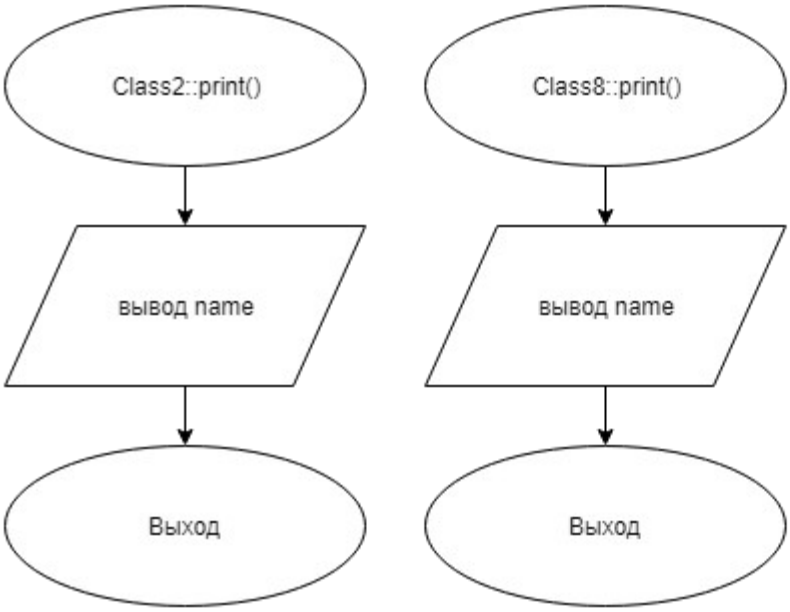

Рисунок 6 – Блок-схема алгоритма

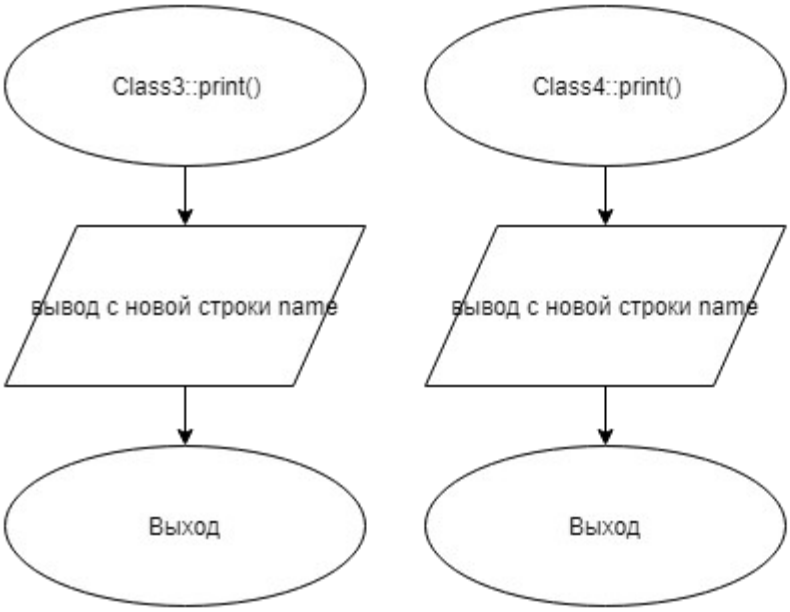

Рисунок 7 – Блок-схема алгоритма

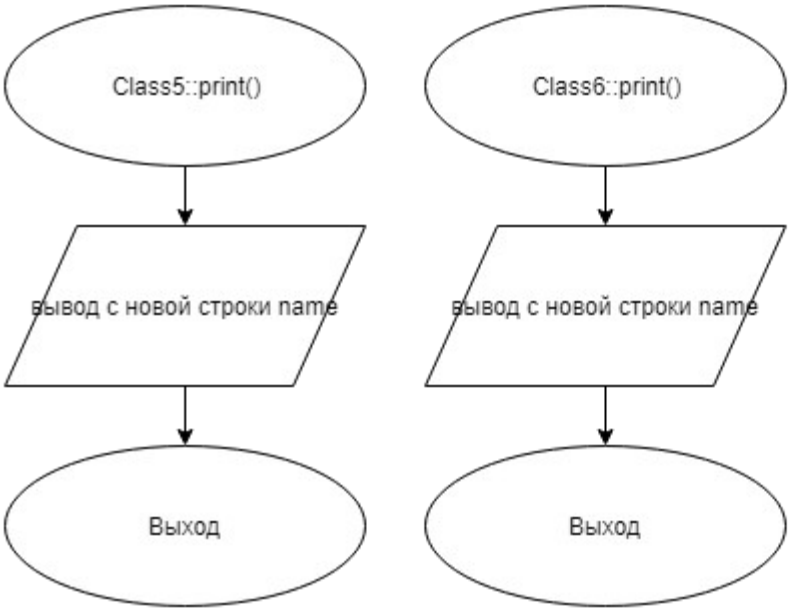

Рисунок 8 – Блок-схема алгоритма

Рисунок 9 – Блок-схема алгоритма

5 КОД ПРОГРАММЫ

Программная реализация алгоритмов для решения задачи представлена ниже.

5.1 Файл Class1.cpp

Листинг 1 – Class1.cpp

```
#include "Class1.h"

Class1::Class1(string name)
{
   this->name = name + "_1";
}

string Class1::print()
{
   return name;
}
```

5.2 Файл Class1.h

Листинг 2 – Class1.h

```
#ifndef __CLASS1__H
#define __CLASS1__H
#include <iostream>
#include <string>
using namespace std;

class Class1
{
private:
   string name;
public:
   Class1(string name);
   string print();
};


#endif
```

5.3 Файл Class2.cpp

Листинг 3 – Class2.cpp

```
#include "Class2.h"

Class2::Class2(string name):Class1(name + "_2")
{
   this-> name = name + "_2";
}

string Class2::print()
{
   return name;
}
```

5.4 Файл Class2.h

Листинг 4 – Class2.h

```
#ifndef __CLASS2__H
#define __CLASS2__H
#include "Class1.h"

class Class2:public Class1
{
private:
   string name;
public:
   Class2(string name);
   string print();
};


#endif
```

5.5 Файл Class3.cpp

Листинг 5 – Class3.cpp

```
#include "Class3.h"

Class3::Class3(string name):Class1(name + "_3")
```

```
{
   this-> name = name + "_3";
}

string Class3::print()
{
   return name;
}
```

5.6 Файл Class3.h

Листинг 6 – Class3.h

```
#ifndef __CLASS3__H
#define __CLASS3__H
#include "Class1.h"

class Class3:public Class1
{
private:
   string name;
public:
   Class3(string name);
   string print();
};


#endif
```

5.7 Файл Class4.cpp

Листинг 7 – Class4.cpp

```
#include "Class4.h"

Class4::Class4(string name):Class1(name + "_4")
{
   this-> name = name + "_4";
}

string Class4::print()
{
   return name;
}
```

5.8 Файл Class4.h

Листинг 8 – Class4.h

```
#ifndef __CLASS4__H
#define __CLASS4__H

#include "Class1.h"

class Class4:virtual public Class1
{
private:
   string name;
public:
   Class4(string name);
   string print();
};


#endif
```

5.9 Файл Class5.cpp

Листинг 9 – Class5.cpp

```
#include "Class5.h"

Class5::Class5(string name):Class1(name + "_5")
{
   this-> name = name + "_5";
}

string Class5::print()
{
   return name;
}
```

5.10 Файл Class5.h

Листинг 10 – Class5.h

```
#ifndef __CLASS5__H
#define __CLASS5__H
```

```
#include "Class1.h"

class Class5:virtual public Class1
{
private:
   string name;
public:
   Class5(string name);
   string print();
};


#endif
```

5.11 Файл Class6.cpp

Листинг 11 – Class6.cpp

```
#include "Class6.h"

Class6::Class6(string name):Class2(name + "_6"), Class3(name + "_6")
{
   this-> name = name + "_6";
}

string Class6::print()
{
   return name;
}
```

5.12 Файл Class6.h

Листинг 12 – Class6.h

```
#ifndef __CLASS6__H
#define __CLASS6__H

#include "Class2.h"
#include "Class3.h"

class Class6:public Class2, public Class3
{
   private:
      string name;
```

```
   public:
      Class6(string name);
      string print();
};



#endif
```

5.13 Файл Class7.cpp

Листинг 13 – Class7.cpp

```
#include "Class7.h"

Class7::Class7(string  name):Class1(name  +  "_7"),  Class4(name  +  "_7"),
Class5(name + "_7")
{
   this-> name = name + "_7";
}

string Class7::print()
{
   return name;
}
```

5.14 Файл Class7.h

Листинг 14 – Class7.h

```
#ifndef __CLASS7__H
#define __CLASS7__H
#include "Class4.h"
#include "Class5.h"

class Class7:public Class4, public Class5
{
   private:
      string name;
   public:
      Class7(string name);
      string print();
};

#endif
```

5.15 Файл Class8.cpp

Листинг 15 – Class8.cpp

```
#include "Class8.h"

Class8::Class8(string  name):Class7::Class1(name  +  "_8"),  Class6(name  +
"_8"), Class7(name + "_8")
{
   this-> name = name + "_8";
}

string Class8::print()
{
   return name;
}
```

5.16 Файл Class8.h

Листинг 16 – Class8.h

```
#ifndef __CLASS8__H
#define __CLASS8__H
#include "Class6.h"
#include "Class7.h"

class Class8:public Class6, public Class7
{
   private:
      string name;
   public:
      Class8(string name);
      string print();
};



#endif
```

5.17 Файл main.cpp

Листинг 17 – main.cpp

```
#include <stdlib.h>
#include <stdio.h>
#include "Class8.h"

int main()
{
   Class8* pobj;
   string name;
   cin >> name;
   Class8 obj(name);
   pobj = &obj;
   cout << ((Class1*)(Class2*)pobj)->print() << '\n';
   cout << ((Class1*)(Class3*)pobj)->print() << '\n';
   cout << ((Class1*)(Class4*)pobj)->print() << '\n';
   cout << ((Class1*)(Class5*)pobj)->print() << '\n';
   cout << ((Class2*)pobj) ->print() << '\n';
   cout << ((Class3*)pobj) ->print() << '\n';
   cout << ((Class4*)pobj) ->print() << '\n';
   cout << ((Class5*)pobj) ->print() << '\n';
   cout << ((Class6*)pobj) ->print() << '\n';
   cout << ((Class7*)pobj) ->print() << '\n';
   cout << pobj->print();
   return(0);
}
```

6 ТЕСТИРОВАНИЕ

Результат тестирования программы представлен в таблице 18.

Таблица 18 – 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).