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
IIdent

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

Оператор ввода-вывода

Условный оператор

Оператор цикла

Класс Class1:

Свойства класса Class1:

private:

- string name - имя объекта
- bool isPrinted - объект выводился в консоль?

protected:

- static bool isFirstPrint = true - ещё ни один объект не выводился в консоль?

Методы класса Class1:

public:

- Class1(string id) - конструктор с индентификатором id
- Class1() - пустой конструктор
- virtual getTree(int steps) - вывод имени этого объекта и детей-объектов

Класс Class[2-8]:

Свойства класса Class[2-8]:

private:

- string name - имя объекта
- bool isPrinted - объект выводился в консоль?

Методы класса Class[2-8]:

public:

- Class[2-8](string id) - конструктор с индентификатором id
- virtual getTree(int steps) - вывод имени этого объекта и детей-объектов

Наследования:

- Class2: public Class1
- Class3: public Class1
- Class4: virtual public Class1
- Class5: virtual public Class1
- Class6: public Class2, public Class3
- Class7: public Class4, public Class5
- Class8: public Class6, public Class7

Описание алгоритма

Функция: main

Функционал: Основная функция

Параметры: нет

Возвращаемое значение: int, код ошибки

№	Предикат	Действия	№ перехода	Комментарий
1		Ввод идентификатора	2	
2		Создание указателя с на объект класса ClassX параметризованным конструктором Class8	3	
3		Начало цикла i(3..0, -1)	4	Цикл для вывода разных уровней дерева созданных объектов
4	i меньше 0		∅	
	Else		5	
5		Динамическое преобразование указателя с из ClassX в Class8	6	
6		Вызов метода getTree объекта с	7	Вывод уровня дерева созданных объектов
7		i--	4	

Класс объекта: Class8

Модификатор доступа: public

Метод: Class8

Функционал: Параметризованный конструктор

Параметры: string, id, идентификатор

Возвращаемое значение: void

№	Предикат	Действия	№ перехода	Комментарий
1		Инициализация свойства name класса	∅	

Класс объекта: Class8

Модификатор доступа: public

Метод: getTree

Функционал: Вывод нужного уровня дерева

Параметры: int, steps, на сколько шагов осталось спуститься по дереву

Возвращаемое значение: void

№	Предикат	Действия	№ перехода	Комментарий
1	steps равен 0		2	
	Else		5	
2	isPrinted не истина	Присвоение isPrinted = true	3	
	Else		∅	
3	isFirstPrint истина	Присвоение isFirstPrint = false	4	
	Else	Вывод "\n"	4	
4		Вывод name - свойства класса	∅	
5		Вызов метода getTree(steps - 1) класса Class6	6	
6		Вызов метода getTree(steps - 1) класса Class7	∅	

Класс объекта: Class7

Модификатор доступа: public

Метод: Class7

Функционал: Параметризованный конструктор

Параметры: string, id, идентификатор

Возвращаемое значение: void

№	Предикат	Действия	№ перехода	Комментарий
1		Инициализация свойства name класса	∅	

Класс объекта: Class7

Модификатор доступа: public

Метод: getTree

Функционал: Вывод нужного уровня дерева

Параметры: int, steps, на сколько шагов осталось спуститься по дереву

Возвращаемое значение: void

№	Предикат	Действия	№ перехода	Комментарий
1	steps равен 0		2	
	Else		5	
2	isPrinted не истина	Присвоение isPrinted = true	3	

	Else		∅	
3	isFirstPrint истина	Присвоение isFirstPrint = false	4	
	Else	Вывод "\n"	4	
4		Вывод name - свойства класса	∅	
5		Вызов метода getTree(steps - 1) класса Class5	6	
6		Вызов метода getTree(steps - 1) класса Class4	∅	

Класс объекта: Class6

Модификатор доступа: public

Метод: Class6

Функционал: Параметризованный конструктор

Параметры: string, id, идентификатор

Возвращаемое значение: void

№	Предикат	Действия	№ перехода	Комментарий
1		Инициализация свойства name класса	∅	

Класс объекта: Class6

Модификатор доступа: public

Метод: getTree

Функционал: Вывод нужного уровня дерева

Параметры: int, steps, на сколько шагов осталось спуститься по дереву

Возвращаемое значение: void

№	Предикат	Действия	№ перехода	Комментарий
1	steps равен 0		2	
	Else		5	
2	isPrinted не истина	Присвоение isPrinted = true	3	
	Else		∅	
3	isFirstPrint	Присвоение isFirstPrint = false	4	

	истина			
	Else	Вывод "\n"	4	
4		Вывод name - свойства класса	∅	
5		Вызов метода getTree(steps - 1) класса Class3	6	
6		Вызов метода getTree(steps - 1) класса Class2	∅	

Класс объекта: Class5

Модификатор доступа: public

Метод: Class5

Функционал: Параметризованный конструктор

Параметры: int, steps, на сколько шагов осталось спуститься по дереву

Возвращаемое значение: void

№	Предикат	Действия	№ перехода	Комментарий
1		Инициализация свойства name класса	∅	

Класс объекта: Class5

Модификатор доступа: public

Метод: getTree

Функционал: Вывод нужного уровня дерева

Параметры: int, steps, на сколько шагов осталось спуститься по дереву

Возвращаемое значение: void

№	Предикат	Действия	№ перехода	Комментарий
1	steps равен 0		2	
	Else		5	
2	isPrinted не истина	Присвоение isPrinted = true	3	
	Else		∅	
3	isFirstPrint истина	Присвоение isFirstPrint = false	4	
	Else	Вывод "\n"	4	
4		Вывод name - свойства класса	∅	
5		Вызов метода getTree(steps - 1) класса Class1	∅	

Класс объекта: Class4

Модификатор доступа: public

Метод: Class4

Функционал: Параметризованный конструктор

Параметры: string, id, идентификатор

Возвращаемое значение: void

№	Предикат	Действия	№ перехода	Комментарий
1		Инициализация свойства name класса	∅	

Класс объекта: Class4

Модификатор доступа: public

Метод: getTree

Функционал: Вывод нужного уровня дерева

Параметры: int, steps, на сколько шагов осталось спуститься по дереву

Возвращаемое значение: void

№	Предикат	Действия	№ перехода	Комментарий
1	steps равен 0		2	
	Else		5	
2	isPrinted не истина	Присвоение isPrinted = true	3	
	Else		∅	
3	isFirstPrint истина	Присвоение isFirstPrint = false	4	
	Else	Вывод "\n"	4	
4		Вывод name - свойства класса	5	
5		Вызов метода getTree(steps - 1) класса Class1	∅	

Класс объекта: Class3

Модификатор доступа: public

Метод: Class3

Функционал: Параметризованный конструктор

Параметры: string, id, идентификатор

Возвращаемое значение: void

№	Предикат	Действия	№ перехода	Комментарий
1		Инициализация свойства name класса	∅	

Класс объекта: Class3

Модификатор доступа: public

Метод: getTree

Функционал: Вывод нужного уровня дерева

Параметры: int, steps, на сколько шагов осталось спуститься по дереву

Возвращаемое значение: void

№	Предикат	Действия	№ перехода	Комментарий
1	steps равен 0		2	
	Else		5	
2	isPrinted не истина	Присвоение isPrinted = true	3	
	Else		∅	
3	isFirstPrint истина	Присвоение isFirstPrint = false	4	
	Else	Вывод "\n"	4	
4		Вывод name - свойства класса	5	
5		Вызов метода getTree(steps - 1) класса Class1	∅	

Класс объекта: Class2

Модификатор доступа: public

Метод: Class2

Функционал: Параметризованный конструктор

Параметры: string, id, идентификатор

Возвращаемое значение: void

№	Предикат	Действия	№ перехода	Комментарий
1		Инициализация свойства name класса	∅	

Класс объекта: Class2

Модификатор доступа: public

Метод: getTree

Функционал: Вывод нужного уровня дерева

Параметры: int, steps, на сколько шагов осталось спуститься по дереву

Возвращаемое значение: void

№	Предикат	Действия	№ перехода	Комментарий
1	steps равен 0		2	
	Else		5	
2	isPrinted не истина	Присвоение isPrinted = true	3	
	Else		∅	
3	isFirstPrint истина	Присвоение isFirstPrint = false	4	
	Else	Вывод "\n"	4	
4		Вывод name - свойства класса	5	
5		Вызов метода getTree(steps - 1) класса Class1	∅	

Класс объекта: Class1

Модификатор доступа: public

Метод: Class1

Функционал: Параметризованный конструктор

Параметры: string, id, идентификатор

Возвращаемое значение: void

№	Предикат	Действия	№ перехода	Комментарий
1		Инициализация свойства name класса	∅	

Класс объекта: Class1

Модификатор доступа: public

Метод: getTree

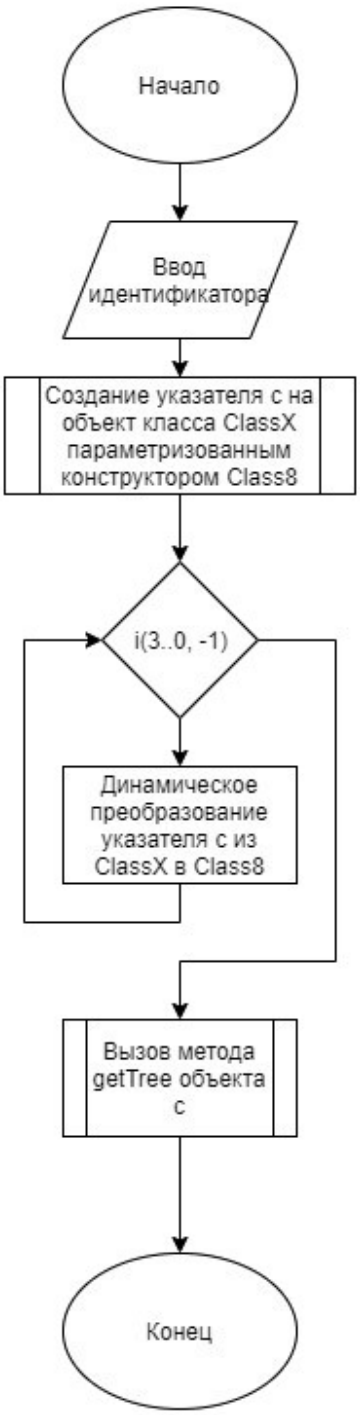
Функционал: Вывод нужного уровня дерева

Параметры: int, steps, на сколько шагов осталось спуститься по дереву

Возвращаемое значение: void

№	Предикат	Действия	№ перехода	Комментарий
1	steps равен 0		2	
	Else		∅	
2	isPrinted не истина	Присвоение isPrinted = true	3	
	Else		∅	
3	isFirstPrint истина	Присвоение isFirstPrint = false	4	
	Else	Вывод "\n"	4	
4		Вывод name - свойства класса	∅	

Блок-схема алгоритма


Начало
Ввод идентификатора
Создание указателя с на объект класса ClassX параметризованным конструктором Class8
i(3..0, -1)
Динамическое преобразование указателя с из ClassX в Class8
Вызов метода getTree объекта c
Конец


Начало
Class8::Class8
(string id)
Инициализация
свойства name
класса
Конец
Class8::Class8


Начало
Class7::Class7
(string id)
Инициализация
свойства name
класса
Конец
Class7::Class7


Начало
Class6::Class6
(string id)
Инициализация
свойства name
класса
Конец
Class6::Class6

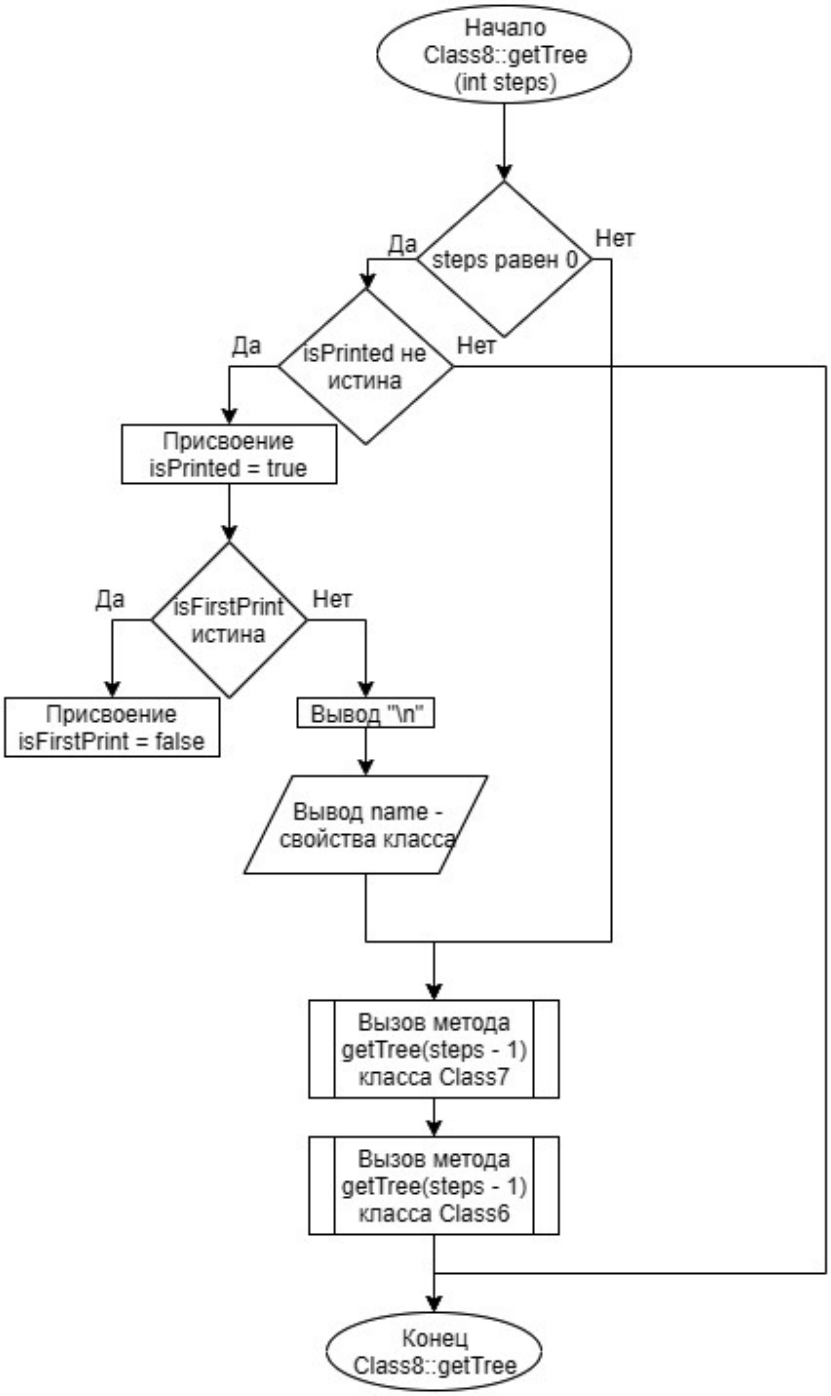

Начало
Class5::Class5
(string id)
Инициализация
свойства name
класса
Конец
Class5::Class5

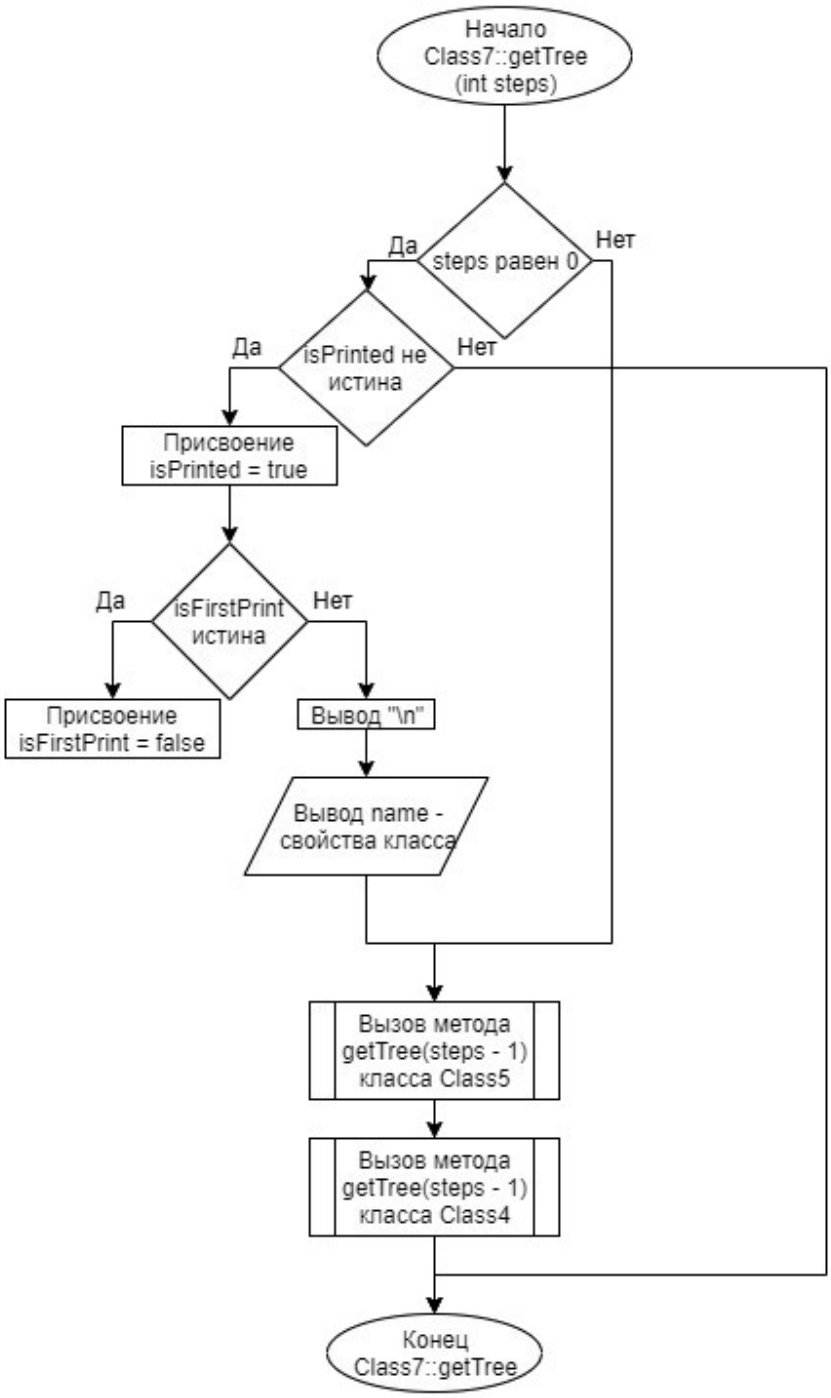

Начало
Class4::Class4
(string id)
Инициализация
свойства name
класса
Конец
Class4::Class4

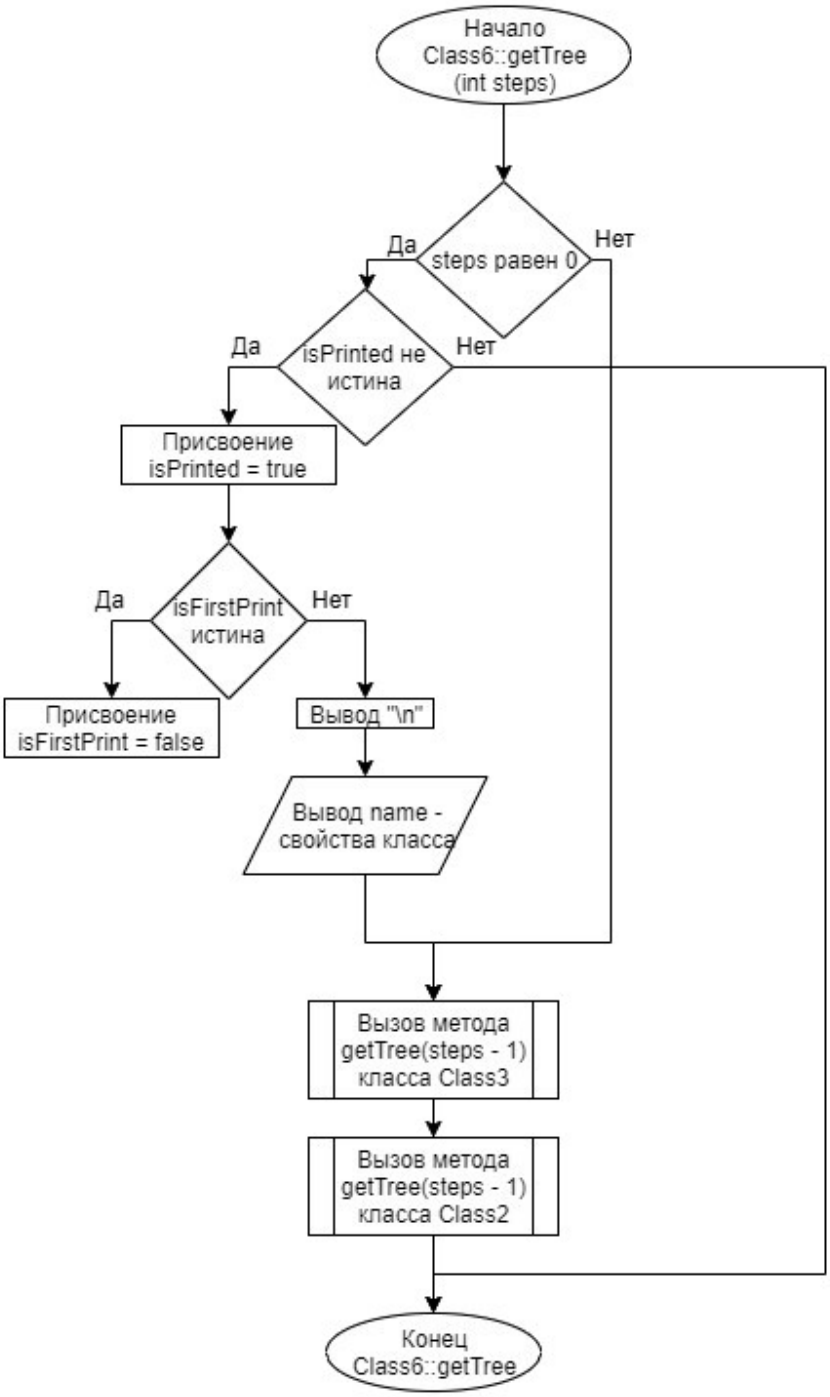

Начало
Class3::Class3
(string id)
Инициализация
свойства name
класса
Конец
Class3::Class3

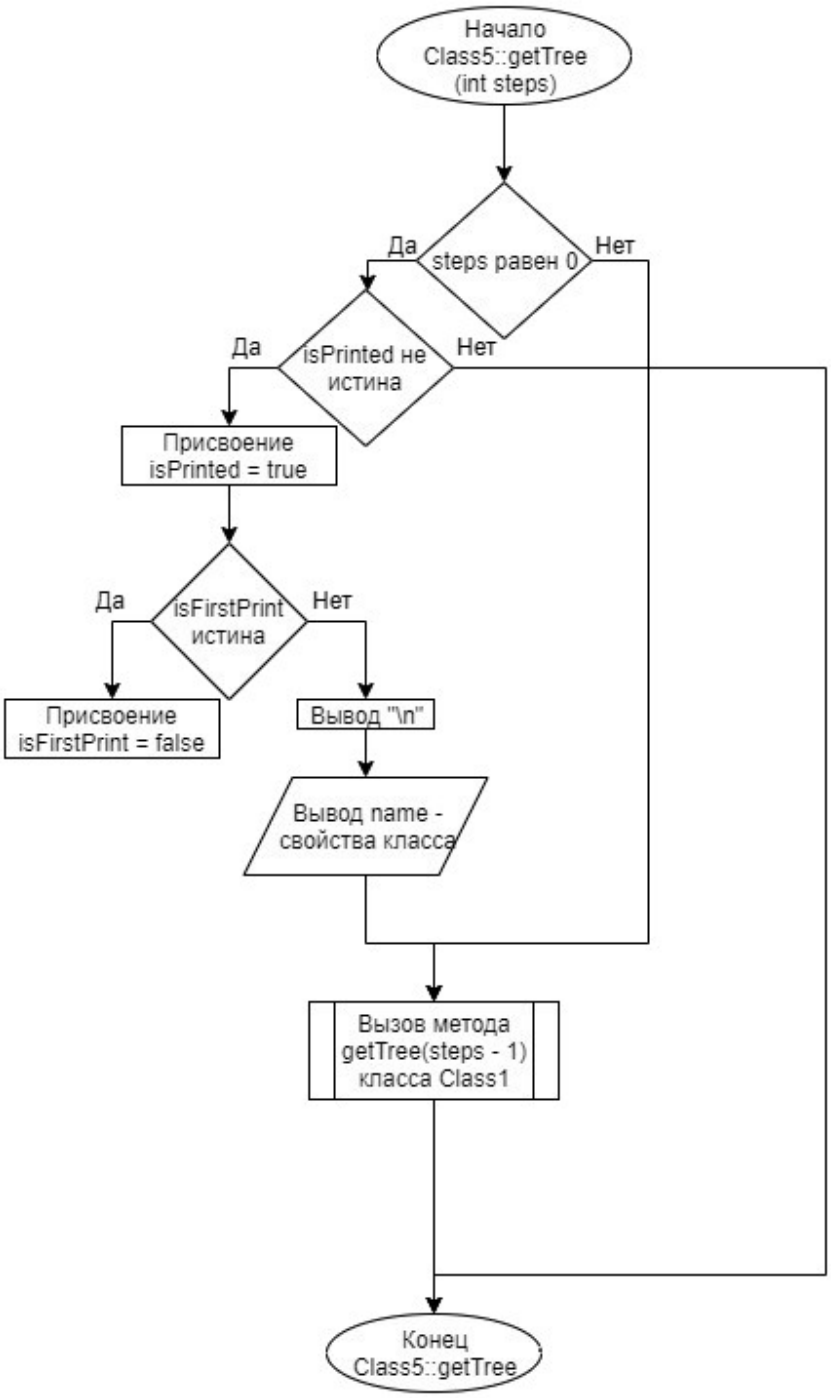

Начало
Class2::Class2
(string id)
Инициализация
свойства name
класса
Конец
Class2::Class2

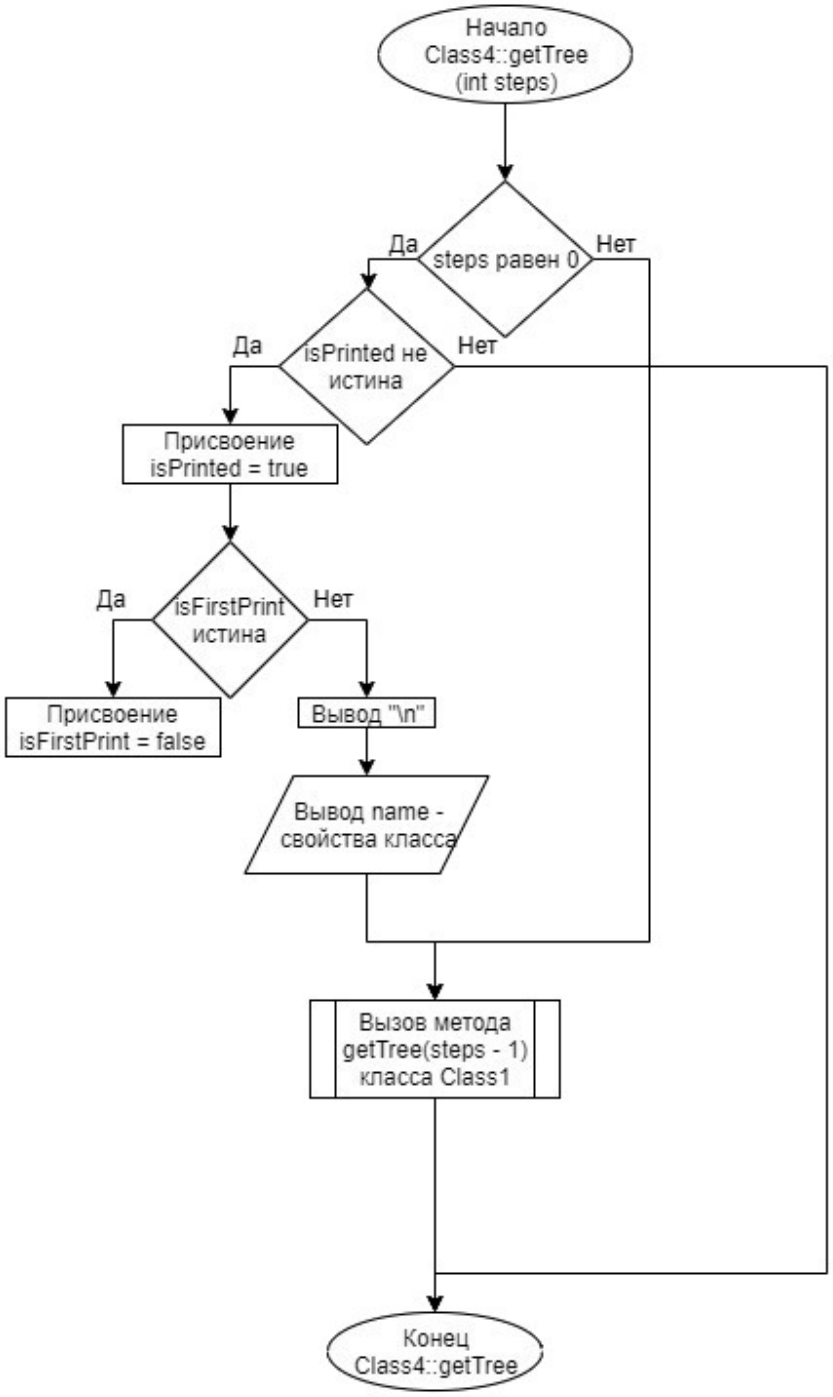

Начало
Class1::Class1
(string id)
Инициализация
свойства name
класса
Конец
Class1::Class1

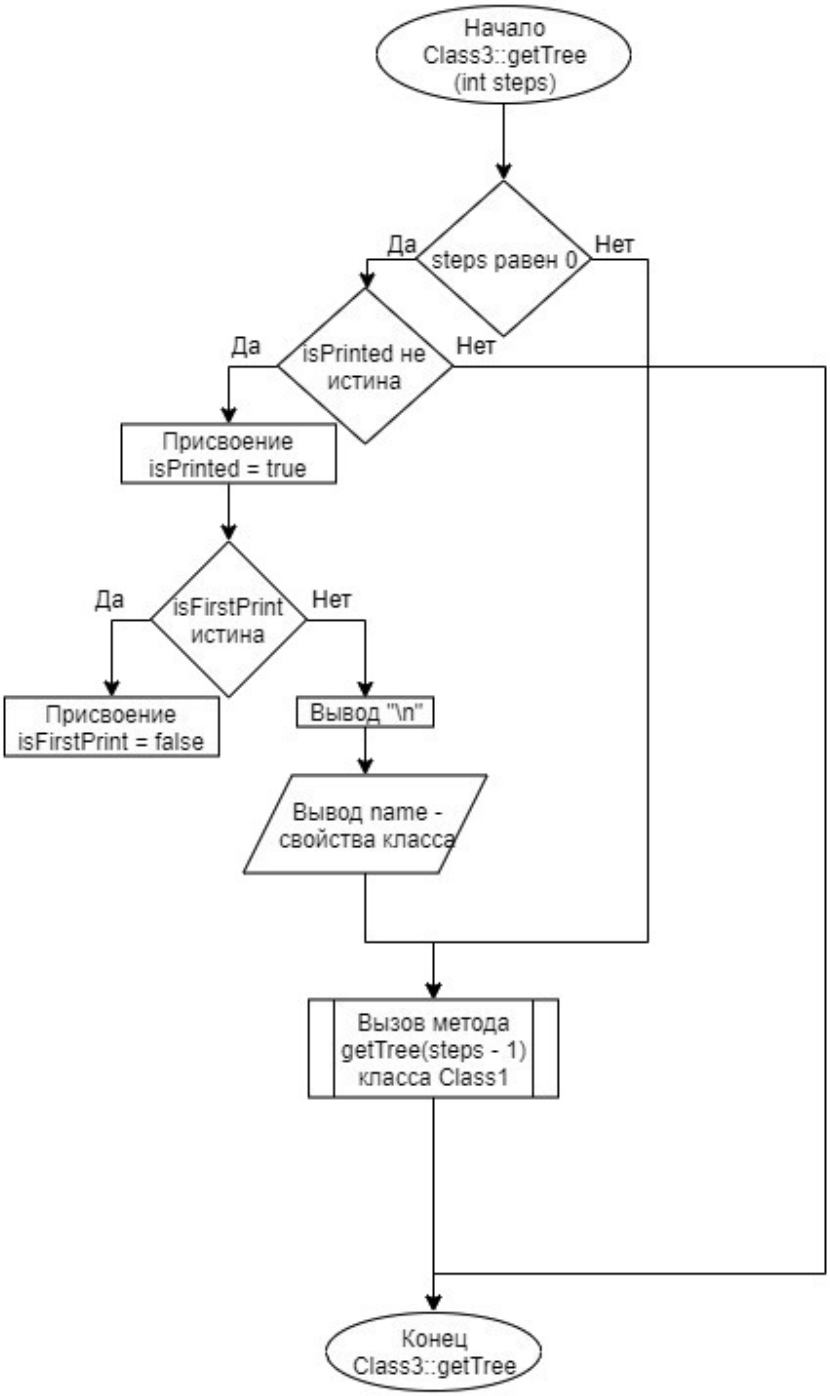

Начало
Class8::getTree
(int steps)
steps равен 0
Да
Нет
isPrinted не
истина
Да
Нет
Присвоение
isPrinted = true
isFirstPrint
истина
Да
Нет
Присвоение
isFirstPrint = false
Вывод "\n"
Вывод name -
свойства класса
Вызов метода
getTree(steps - 1)
класса Class7
Вызов метода
getTree(steps - 1)
класса Class6
Конец
Class8::getTree

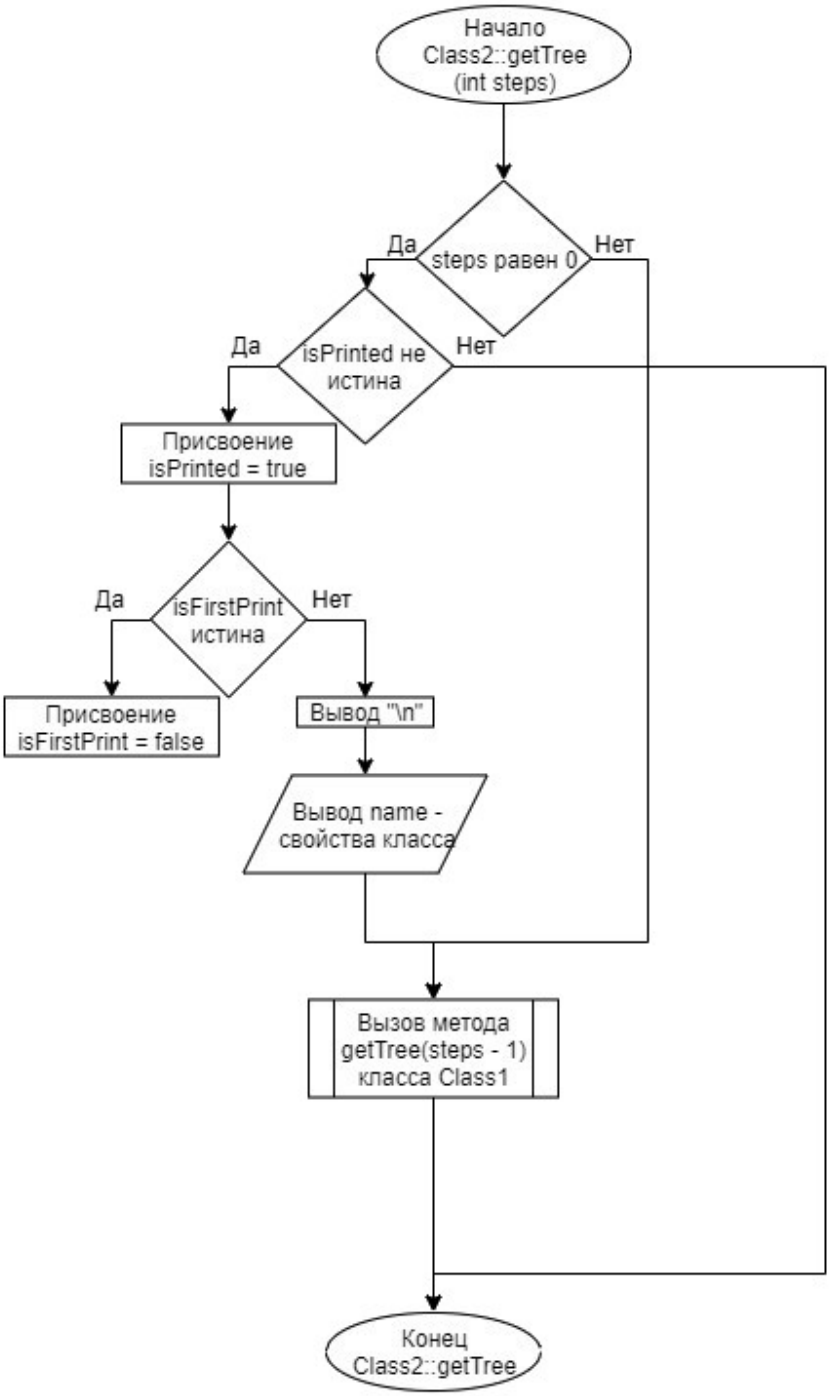

Начало
Class7::getTree
(int steps)
steps равен 0
Да
Нет
isPrinted не
истина
Да
Нет
Присвоение
isPrinted = true
isFirstPrint
истина
Да
Нет
Присвоение
isFirstPrint = false
Вывод "\n"
Вывод name -
свойства класса
Вызов метода
getTree(steps - 1)
класса Class5
Вызов метода
getTree(steps - 1)
класса Class4
Конец
Class7::getTree

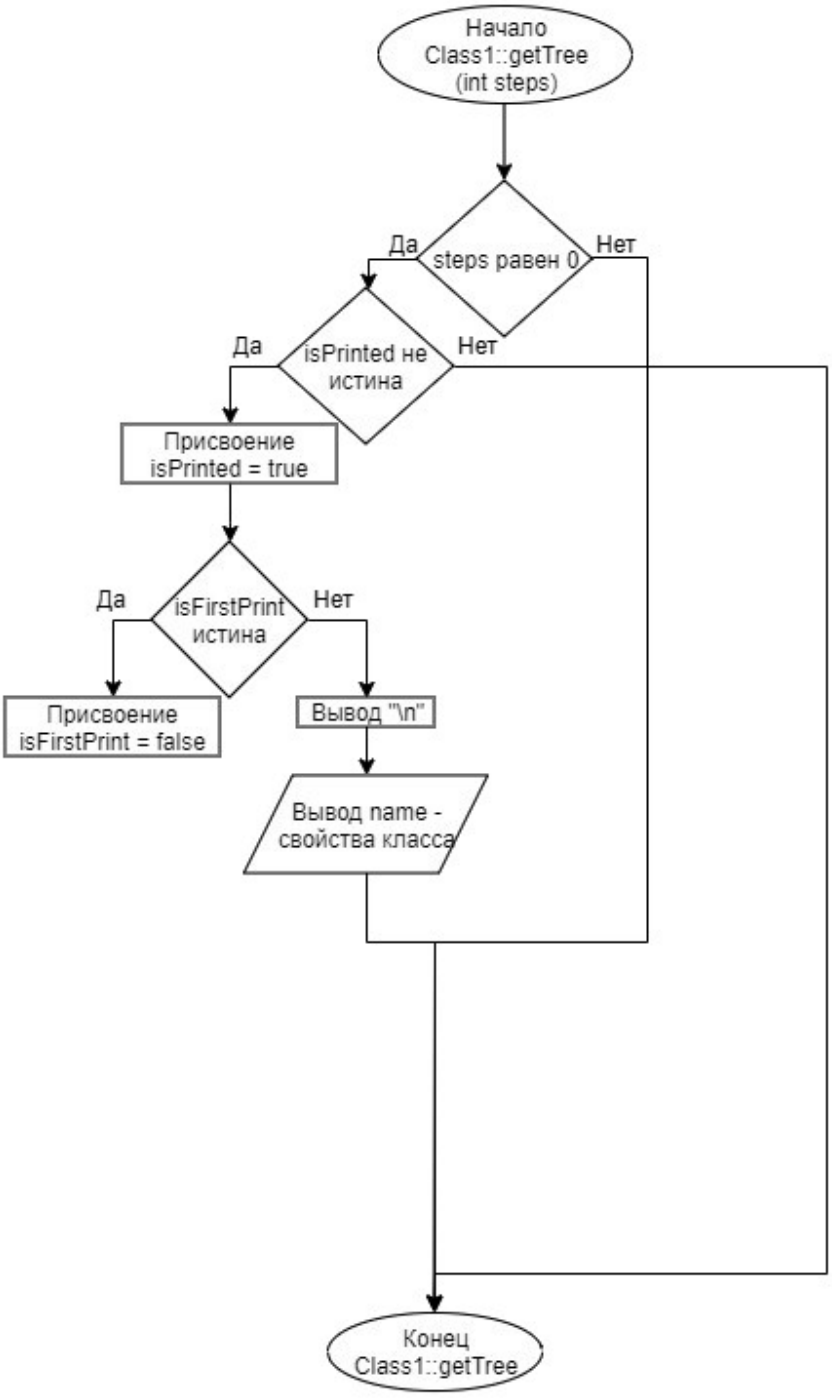

Начало
Class6::getTree
(int steps)
steps равен 0
Да
Нет
isPrinted не
истина
Да
Нет
Присвоение
isPrinted = true
isFirstPrint
истина
Да
Нет
Присвоение
isFirstPrint = false
Вывод "\n"
Вывод name -
свойства класса
Вызов метода
getTree(steps - 1)
класса Class3
Вызов метода
getTree(steps - 1)
класса Class2
Конец
Class6::getTree


Начало
Class5::getTree
(int steps)
steps равен 0
Да
Нет
isPrinted не
истина
Да
Нет
Присвоение
isPrinted = true
isFirstPrint
истина
Да
Нет
Присвоение
isFirstPrint = false
Вывод "\n"
Вывод name -
свойства класса
Вызов метода
getTree(steps - 1)
класса Class1
Конец
Class5::getTree


Начало
Class4::getTree
(int steps)
steps равен 0
Да
Нет
isPrinted не
истина
Да
Нет
Присвоение
isPrinted = true
isFirstPrint
истина
Да
Нет
Присвоение
isFirstPrint = false
Вывод "\n"
Вывод name -
свойства класса
Вызов метода
getTree(steps - 1)
класса Class1
Конец
Class4::getTree


Начало
Class3::getTree
(int steps)
steps равен 0
Да
Нет
isPrinted не
истина
Да
Нет
Присвоение
isPrinted = true
isFirstPrint
истина
Да
Нет
Присвоение
isFirstPrint = false
Вывод "\n"
Вывод name -
свойства класса
Вызов метода
getTree(steps - 1)
класса Class1
Конец
Class3::getTree


Начало
Class2::getTree
(int steps)
steps равен 0
Да
Нет
isPrinted не
истина
Да
Нет
Присвоение
isPrinted = true
isFirstPrint
истина
Да
Нет
Присвоение
isFirstPrint = false
Вывод "\n"
Вывод name -
свойства класса
Вызов метода
getTree(steps - 1)
класса Class1
Конец
Class2::getTree



Код программы

Файл Class1.cpp

```
#include "Class1.h"

bool Class1::isFirstPrint = true;

Class1::Class1(string id = "") {
	name = id + "_1";
}

void Class1::getTree(int steps) {
	if (steps == 0) {
		if (!isPrinted) isPrinted = true;
		else return;
		if (isFirstPrint) isFirstPrint = false;
		else cout << "\n";
		cout << name;
		return;
	}
}
```

Файл Class1.h

```
#ifndef CLASS1_H
#define CLASS1_H

#include <string>
#include <iostream>

using namespace std;

class Class1 {
private:
	string name = "untitled";
	bool isPrinted = false;
protected:
	static bool isFirstPrint;
public:
	Class1(string id);
	Class1() { cout << "Class1 EMPTY constructor\n"; }
	virtual void getTree(int = 0);
};

#endif
```

Файл Class2.cpp

```
#include "Class2.h"

Class2::Class2(string id) : Class1(id) {
	name = id + "_2";
}

void Class2::getTree(int steps) {
	if (steps == 0) {
		if (!isPrinted) isPrinted = true;
		else return;
		if (isFirstPrint) isFirstPrint = false;
		else cout << "\n";
		cout << name;
		return;
	}
	this->Class1::getTree(steps - 1);
}
```

Файл Class2.h

```
#ifndef CLASS2_H
#define CLASS2_H

#include <string>
#include "Class1.h"

using namespace std;

class Class2 : public Class1 {
private:
	string name;
	bool isPrinted = false;
public:
	Class2(string id);
	virtual void getTree(int = 0);
};

#endif
```

Файл Class3.cpp

```
#include "Class3.h"

Class3::Class3(string id) : Class1(id) {
```

```
	name = id + "_3";
}

void Class3::getTree(int steps) {
	if (steps == 0) {
		if (!isPrinted) isPrinted = true;
		else return;
		if (isFirstPrint) isFirstPrint = false;
		else cout << "\n";
		cout << name;
		return;
	}
	this->Class1::getTree(steps - 1);
}
```

Файл Class3.h

```
#ifndef CLASS3_H
#define CLASS3_H

#include <string>
#include "Class1.h"
#include "Class2.h"

using namespace std;

class Class3 : public Class1 {
private:
	string name;
	bool isPrinted = false;
public:
	Class3(string id);
	virtual void getTree(int steps = 0);
};

#endif
```

Файл Class4.cpp

```
#include "Class4.h"

Class4::Class4(string id) : Class1(id) {
	name = id + "_4";
}

void Class4::getTree(int steps) {
	if (steps == 0) {
		if (!isPrinted) isPrinted = true;
```

```
                else return;
                if (isFirstPrint) isFirstPrint = false;
                else cout << "\n";
                cout << name;
                return;
        }
        this->Class1::getTree(steps - 1);
}
```

Файл Class4.h

```
#ifndef CLASS4_H
#define CLASS4_H

#include <string>
#include "Class1.h"
#include "Class3.h"

using namespace std;

class Class4 : virtual public Class1 {
private:
        string name;
        bool isPrinted = false;
public:
        Class4(string id);
        virtual void getTree(int steps = 0);
};

#endif
```

Файл Class5.cpp

```
#include "Class5.h"

Class5::Class5(string id) : Class1(id) {
        name = id + "_5";
}

void Class5::getTree(int steps) {
        if (steps == 0) {
                if (!isPrinted) isPrinted = true;
                else return;
                if (isFirstPrint) isFirstPrint = false;
```

```
                else cout << "\n";
                cout << name;
                return;
        }
        this->Class1::getTree(steps - 1);
}
```

Файл Class5.h

```
#ifndef CLASS5_H
#define CLASS5_H

#include <string>
#include "Class1.h"
#include "Class4.h"

using namespace std;

class Class5 : virtual public Class1 {
private:
        string name;
        bool isPrinted = false;
public:
        Class5(string id);
        virtual void getTree(int steps = 0);
};

#endif
```

Файл Class6.cpp

```
#include "Class6.h"

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

void Class6::getTree(int steps) {
        if (steps == 0) {
                if (!isPrinted) isPrinted = true;
                else return;
                if (isFirstPrint) isFirstPrint = false;
                else cout << "\n";
                cout << name;
                return;
        }
        this->Class2::getTree(steps - 1);
        this->Class3::getTree(steps - 1);
```

```
}
```

Файл Class6.h

```
#ifndef CLASS6_H
#define CLASS6_H

#include <string>
#include "Class2.h"
#include "Class3.h"
#include "Class5.h"

using namespace std;

class Class6 : public Class2, public Class3 {
private:
        string name;
        bool isPrinted = false;
public:
        Class6(string id);
        virtual void getTree(int steps = 0);
};

#endif
```

Файл Class7.cpp

```
#include "Class7.h"

Class7::Class7(string id) : Class4(id), Class5(id) {
        name = id + "_7";
}

void Class7::getTree(int steps) {
        if (steps == 0) {
                if (!isPrinted) isPrinted = true;
                else return;
                if (isFirstPrint) isFirstPrint = false;
                else cout << "\n";
                cout << name;
                return;
        }
        this->Class4::getTree(steps - 1);
        this->Class5::getTree(steps - 1);
```

```
}
```

Файл Class7.h

```
#ifndef CLASS7_H
#define CLASS7_H

#include <string>
#include "Class4.h"
#include "Class5.h"
#include "Class6.h"

using namespace std;

class Class7 : public Class4, public Class5 {
private:
        string name;
        bool isPrinted = false;
public:
        Class7(string id);
        virtual void getTree(int steps = 0);
};

#endif
```

Файл Class8.cpp

```
#include "Class8.h"

Class8::Class8(string id) : Class6(id), Class7(id), Class5::Class1(id) {
        name = id + "_8";
}

void Class8::getTree(int steps) {
        if (steps == 0) {
                if (!isPrinted) isPrinted = true;
                else return;
                if (isFirstPrint) isFirstPrint = false;
                else cout << "\n";
                cout << name;
                return;
        }
        this->Class6::getTree(steps - 1);
        this->Class7::getTree(steps - 1);
}
```

Файл Class8.h

```
#ifndef CLASS8_H
#define CLASS8_H

#include <string>
#include "Class6.h"
#include "Class7.h"

using namespace std;

class Class8 : public Class6, public Class7 {
private:
        string name;
        bool isPrinted = false;
public:
        Class8(string id);
        virtual void getTree(int steps = 0);
};

#endif
```

Файл main.cpp

```
#include "main.h"

int main() {
        string id;
        cin >> id;
        Class2* c = dynamic_cast<Class2*>(new Class8(id));
        for (int i = 3; i >= 0; i--) {
                dynamic_cast<Class8*>(c)->Class8::getTree(i);
        }
        system("pause");
        return 0;
}
```

Файл main.h

```
#ifndef MAIN_H
#define MAIN_H
```

```
#include <stdlib.h>
#include <stdio.h>
#include <vector>
#include <iostream>
#include "Class1.h"
#include "Class2.h"
#include "Class3.h"
#include "Class4.h"
#include "Class5.h"
#include "Class6.h"
#include "Class7.h"
#include "Class8.h"

static bool isFirstPrint = true;

#endif
```

Тестирование

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