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_1_2 »

С тудент группы	ИКБО-01-21	Резников Г.А.
Руководитель практики	Ассистент	Данилович Е.С.
Работа представлена	«___»_____ 2022 г.	_____
		(подпись студента)
Оценка		_____
		(подпись руководителя)

Москва 2022

СОДЕРЖАНИЕ

ВВЕДЕНИЕ..... 4

Постановка задачи..... 5

Метод решения.....8

Описание алгоритма..... 12

Блок-схема алгоритма..... 18

Код программы..... 20

Тестирование.....24

ЗАКЛЮЧЕНИЕ.....25

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ (ИСТОЧНИКОВ)..... 26

ВВЕДЕНИЕ

Постановка задачи

Иерархия наследования

Описать четыре класса которые последовательно наследуют друг друга, последовательными номерами классов 1,2,3,4.
Реализовать программу, в которой использовать единственный указатель на объект базового класса (номер класса 1).
Наследственность реализовать так, что можно было вызвать методы, принадлежащие объекту конкретного класса, только через объект данного класса.

В закрытом разделе каждого класса определены два свойства: строкового типа для наименования объекта и целого типа для значения определенного целочисленного выражения.

Описание каждого класса содержит один параметризированный конструктор с строковым и целочисленным параметром.
В реализации каждого конструктора объекта определяются значения закрытых свойств:
- наименование объекта по шаблону: «значение строкового параметра»_«номер класса»;
- целочисленного свойства значением выражения возведения в степень номера класса целочисленного значения параметра конструктора.

Еще в описании каждого класса определен метод с одинаковым наименованием для всех классов, реализующий вывод значений закрытых свойств класса.

В основной функции реализовать алгоритм:
1. Вводиться идентификатор и натуральное число от 2 до 10.

2. Создать объект класса 4, используя параметризированный конструктор, которому в качестве аргументов передаются введенный идентификатор и натуральное число.
3. Построчно, для всех объектов согласно наследственности, от объекта базового (класс 1) до производного объекта (класса 4) вывести наименование объекта класса и значение целочисленного свойства.

Описание входных данных

Первая строка: «идентификатор»«натуральное число»

Пример ввода:

Object 2

Описание выходных данных

Построчно (четыре строки): «идентификатор»_«номер класса»«значение целочисленного свойства»

Разделитель 1 пробел

Пример вывода:

Object_1 2
Object_2 4

Object_3 8
Object_4 16

Метод решения

Использование указателя p типа class_1, который ссылается на создаваемый объект класса class_4.

Использование объектов cin и cout потока ввода-вывода.

№	Имя класса	Классы-наследники	Модификатор доступа при наследовании	Описание	Номер	Комментарий
1	class_1			Базовый класс в иерархии наследования, содержит необходимые поля и метод вывода		
		class_2	public		2	
2	class_2			содержит необходимые поля и метод вывода		
		class_3	public		3	
3	class_3			содержит необходимые поля и метод вывода		
		class_4	public		4	
4	class_4			содержит необходимые поля и метод вывода		

Класс class_1:

- Свойства/поля:

        - Поле, хранящее получаемое объектом число в 1 степени
            - Имя - num
            - Тип - int
            - Модификатор доступа - private
        - Поле, хранящее имя объекта
            - Имя - name
            - Тип - string
            - Модификатор доступа - private
    - Функционал:
        - Метод class_1 - конструктор - принимает строковую (имя) и числовую (хранимое число) перменные
        - Метод show_priv - виртуальный метод - выводит значения закрытых полей класса

Класс class_2:

- Свойства /поля:
    - Поле, хранящее получаемое число во 2 степени
        - Имя - num
        - Тип - int
        - Модификатор доступа - pivate
    - Поле, хранящее имя объъекта
        - Имя - num
        - Тип - int
        - Модфикатор доступа - private
- Функционал:
    - Метод class_2 - конструктор - принимает строковую (имя) и числовую (хранимое число) переменные
    - Метод show_priv - виртуальный метод - выводит значения

закрытых полей класса

Класс class_3:

- Свойства/поля:
    - Поле, хранящее передаваемое значение в 3 степени
        - Имя - num
        - Тип - int
        - Модификатор доступа - private
    - Поле, хранящее имя объекта
        - Имя - name
        - Тип - string
        - Модификатор доступа - private
- Функционал:
    - Метод class_3 - конструктор - принимает строковую (имя) и числовую (хранимое число) переменные
    - Метод show_priv - виртуальный метод - выводит значения закрытых полей класса

Класс class_4 :

- Свойства/поля:
    - Поле, хранящее передаваемое значение в 4 степени
        - Имя - num
        - Тип - int
        - Модификтор доступа - private
    - Поле, хранящее имя объекта
        - Имя - name

        - Тип - string
        - Модификатор доступа - private
- Функционал:
    - Метод class_3 - конструктор - принимает строковую (имя) и числовую (хранимое число) переменные
    - Метод show_priv - виртуальный метод - выводит значения закрытых полей класса

Описание алгоритма

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

Функция: main

Функционал: Создание объекта class_4 и построчный для всех объектов согласно наследственности, от объекта базового (класс 1) до производного объекта (класса 4) вывод наименование объекта класса и значение целочисленного свойства.

Параметры: нет

Возвращаемое значение: int - код завершения работы программы

Алгоритм функции представлен в таблице 2.

Таблица 2. Алгоритм функции main

№	Предикат	Действия	№ перехода	Комментарий
1		Объявление строковой переменной n и целочисленной a	2	
2		Ввод n и a	3	
3		Создание объекта класса class_4 вызовом конструктора с передачей в качестве параметров n и a, и инициализания указателя p типа class_1* адресом созданного объекта	4	
4		Вызоа метода show_priv, принадлежащего внутреннему	5	

		классу class_1 у p		
5		Вызоа метода show_priv, принадлежащего внутреннему классу class_2 у p	6	
6		Вызоа метода show_priv, принадлежащего внутреннему классу class_3 у p	7	
7		Вызоа метода show_priv, принадлежащего внутреннему классу class_4 у p	8	
8		Возврат 0	∅	

Класс объекта: class_1

Модификатор доступа: public

Метод: class_1

Функционал: Создание объекта класса class_1

Параметры: string n - имя объекта, int a - хранимое значение

Возвращаемое значение: нет

Алгоритм метода представлен в таблице 3.

Таблица 3. Алгоритм метода class_1 класса class_1

№	Предикат	Действия	№ перехода	Комментарий
1		Присвоить полю name значение n+"_1"	2	
2		Привоить полю num значение a, возведенное в 1 степень	∅	

Класс объекта: class_1

Модификатор доступа: public

Метод: show_priv

Функционал: Вывод значений закрытых полей

Параметры: нет

Возвращаемое значение: нет

Алгоритм метода представлен в таблице 4.

Таблица 4. Алгоритм метода show_priv класса class_1

№	Предикат	Действия	№ перехода	Комментарий
1		Вывод через пробел значения поля name и num	∅	

Класс объекта: class_2

Модификатор доступа: public

Метод: class_2

Функционал: Создание объекта класса class_2

Параметры: string n - имя объекта, int a - хранимое значение

Возвращаемое значение: нет

Алгоритм метода представлен в таблице 5.

Таблица 5. Алгоритм метода class_2 класса class_2

№	Предикат	Действия	№ перехода	Комментарий
1		Присвоить полю name значение n+"_2"	2	
2		Привоить полю num значение a, возведенное в 2 степень	∅	

Класс объекта: class_2

Модификатор доступа: public

Метод: show_priv

Функционал: Вывод значений закрытых полей

Параметры: нет

Возвращаемое значение: нет

Алгоритм метода представлен в таблице 6.

Таблица 6. Алгоритм метода show_priv класса class_2

№	Предикат	Действия	№ перехода	Комментарий
1		Вывод через пробел значения поля name и num	∅	

Класс объекта: class_3

Модификатор доступа: public

Метод: class_3

Функционал: Создание объекта класса class_3

Параметры: string n - имя объекта, int a - хранимое значение

Возвращаемое значение: нет

Алгоритм метода представлен в таблице 7.

Таблица 7. Алгоритм метода class_3 класса class_3

№	Предикат	Действия	№ перехода	Комментарий
1		Присвоить полю name значение n+"_3"	2	
2		Привоить полю num значение a,	∅	

		возведенное в 3 степень		

Класс объекта: class_3

Модификатор доступа: public

Метод: show_priv

Функционал: Вывод значений закрытых полей

Параметры: нет

Возвращаемое значение: нет

Алгоритм метода представлен в таблице 8.

Таблица 8. Алгоритм метода show_priv класса class_3

№	Предикат	Действия	№ перехода	Комментарий
1		Вывод через пробел значения поля name и num	∅	

Класс объекта: class_4

Модификатор доступа: public

Метод: class_4

Функционал: Создание объекта класса class_3

Параметры: string n - имя объекта, int a - хранимое значение

Возвращаемое значение: нет

Алгоритм метода представлен в таблице 9.

Таблица 9. Алгоритм метода class_4 класса class_4

№	Предикат	Действия	№ перехода	Комментарий
1		Присвоить полю name значение n+"_4"	2	
2		Привоить полю num значение a, возведенное в 4 степень	∅	

Класс объекта: class_4

Модификатор доступа: public

Метод: show_priv

Функционал: Вывод значений закрытых полей

Параметры: нет

Возвращаемое значение: нет

Алгоритм метода представлен в таблице 10.

Таблица 10. Алгоритм метода show_priv класса class_4

№	Предикат	Действия	№ перехода	Комментарий
1		Вывод через пробел значения поля name и num	∅	

Блок-схема алгоритма

Представим описание алгоритмов в графическом виде на рисунках ниже.

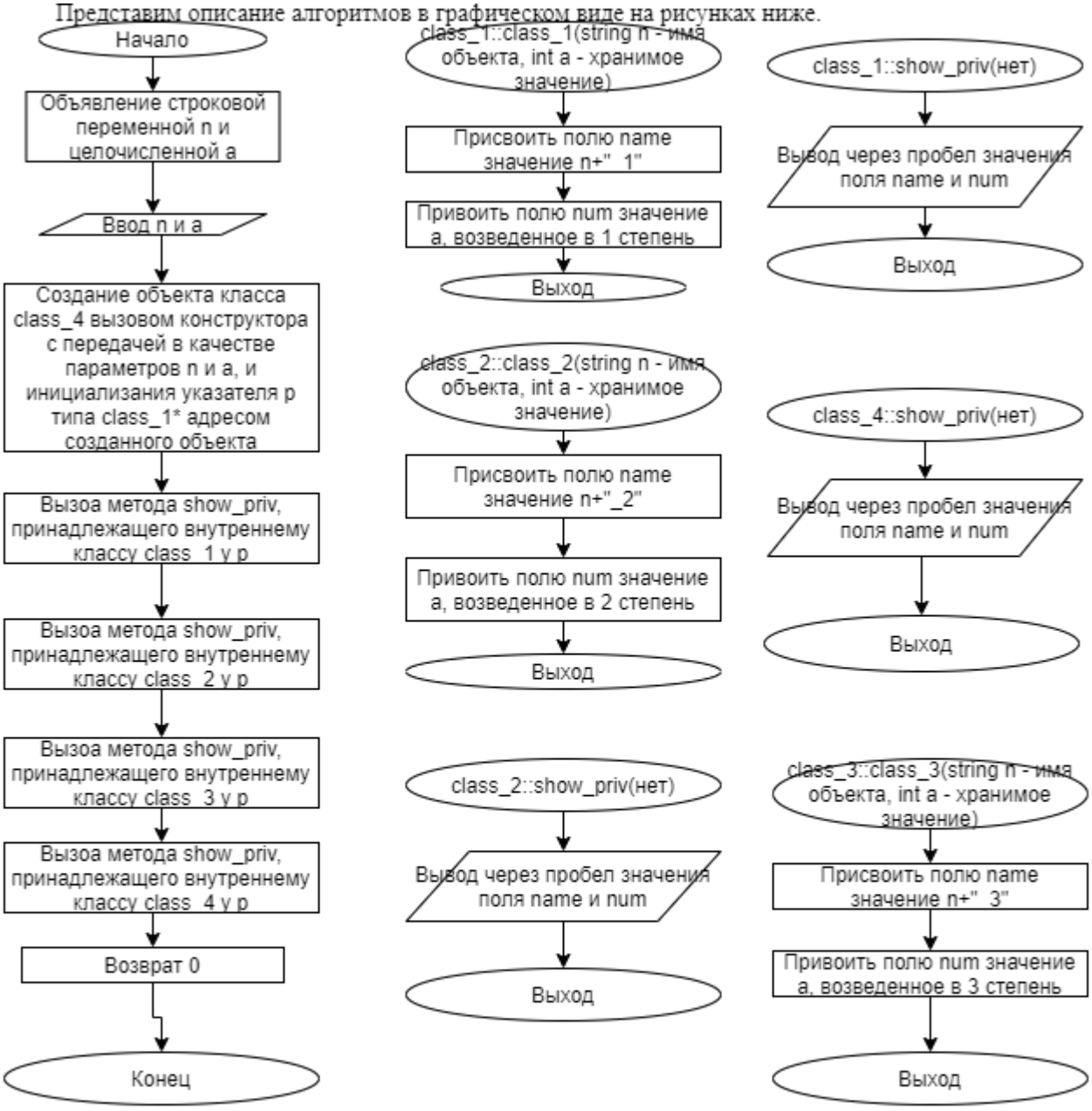

Рис. 1. Блок-схема алгоритма.

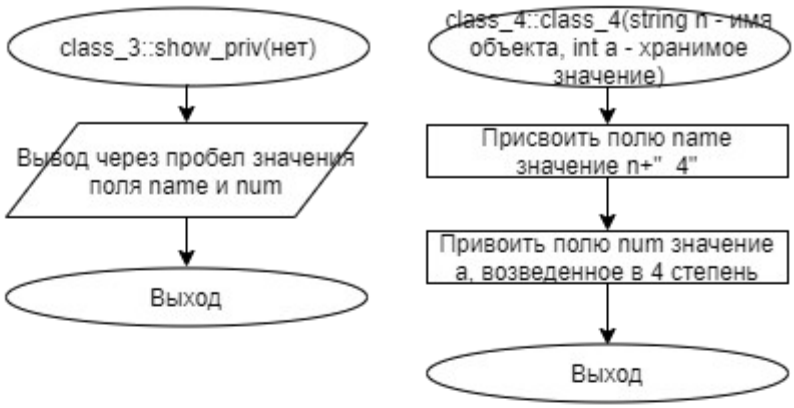

Рис. 2. Блок-схема алгоритма.

Код программы

Программная реализация алгоритмов для решения задачи представлена ниже.

Файл class_1

Файл class_1.cpp

```
#include "class_1.h"
#include <iostream>

void class_1::show_priv()
{
        cout << name << " " << num << endl;
}

class_1::class_1(string n, int a)
{
        name = n + "_1";
        num = a;
}
```

Файл class_1.h

```
#ifndef class_1_H
#define class_1_H
#include <string>
using namespace std;
class class_1
{
private:
        int num;
        string name;
public:
        virtual void show_priv();
        class_1(string, int);
};
#endif //
```

Файл class_2.cpp

```
#include "class_2.h"
#include <iostream>

void class_2::show_priv()
{
	cout << name << " " << num << endl;
}

class_2::class_2(string n, int a) :class_1(n, a)
{
	name = n + "_2";
	num = a*a;
}
```

Файл class_2.h

```
#ifndef class_2_H
#define class_2_H
#include "class_1.h"
class class_2 :
	public class_1
{
private:
	int num;
	string name;
public:
	virtual void show_priv() override;
	class_2(string, int);
};
#endif //
```

Файл class_3.cpp

```
#include "class_3.h"
#include <iostream>

void class_3::show_priv()
{
	cout << name << " " << num << endl;
}

class_3::class_3(string n, int a) :class_2(n, a)
{
	name = n + "_3";
	num = a*a*a;
```

```
}
```

Файл class_3.h

```
#ifndef class_3_H
#define class_3_H
#include "class_2.h"
class class_3 :
        public class_2
{
private:
        int num;
        string name;
public:
        virtual void show_priv() override;
        class_3(string, int);
};
#endif //
```

Файл class_4.cpp

```
#include "class_4.h"
#include <iostream>

void class_4::show_priv()
{
        cout << name <<" "<< num;
}

class_4::class_4(string n, int a):class_3(n,a)
{
        name = n+"_4";
        num = a*a*a*a;
}
```

Файл class_4.h

```
#ifndef class_4_H
#define class_4_H
#include "class_3.h"
class class_4 :
```

```
        public class_3
{
private:
        int num;
        string name;
public:
        virtual void show_priv() override;
        class_4(string, int);
};
#endif //
```

Файл main.cpp

```
#include <iostream>
#include <string>
#include "class_1.h"
#include "class_2.h"
#include "class_3.h"
#include "class_4.h"
using namespace std;
int main()
{
        string n; int a;
        cin >> n >> a;
        class_1* p= new class_4(n, a);
        p->class_1::show_priv();
        ((class_2*)p)->class_2::show_priv();
        ((class_3*)p)->class_3::show_priv();
        ((class_4*)p)->class_4::show_priv();
        return 0;
}
```

Тестирование

Результат тестирования программы представлен в следующей таблице.

Входные данные	Ожидаемые выходные данные	Фактические выходные данные
o 2	o_1 2 o_2 4 o_3 8 o_4 16	o_1 2 o_2 4 o_3 8 o_4 16
o 10	o_1 10 o_2 100 o_3 1000 o_4 10000	o_1 10 o_2 100 o_3 1000 o_4 10000
o 3	o_1 3 o_2 9 o_3 27 o_4 81	o_1 3 o_2 9 o_3 27 o_4 81

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).