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-13-21 | Черномуров С.А. |
| Руководитель практики | Ассистент | Асадова Ю.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..... 23

Тестирование.....27

ЗАКЛЮЧЕНИЕ.....28

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ (ИСТОЧНИКОВ)..... 29

# 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

Для решения задачи используются:

- объекты стандартных потоков ввода и вывода cin и cout соответственно. Используются для ввода с клавиатуры и вывода на экран.
- Объект par класса Base.
- Объекты классов Child2, Child3.
- Объект obj класса Child4.
- **Класс Base:**
    - Свойства/поля:
        - Свойство:
            - Наименование - name;
            - Тип - строковый;
            - Модификатор доступа - закрытый.
        - Свойство:
            - Наименование - n;
            - Тип - целочисленный;
            - Модификатор доступа - закрытый.
    - Методы:
        - Метод Base:
            - Функционал - параметризированный конструктор со строковым и целочисленным параметром.
        - Метод Print_args:
            - Функционал - параметризированный метод, выводящий значения закрытых свойств класса на экран.

- **Класс Child2:**
    - Свойства/поля:
        - Свойство:
            - Наименование - name;
            - Тип - строковый;
            - Модификатор доступа - закрытый.
        - Свойство:
            - Наименование - n;
            - Тип - целочисленный;
            - Модификатор доступа - закрытый.
    - Методы:
        - Метод Child2:
            - Функционал - параметризированный конструктор со строковым и целочисленным параметром.
        - Метод Print_args:
            - Функционал - параметризированный метод, выводящий значения закрытых свойств класса на экран.
- **Класс Child3:**
    - Свойства/поля:
        - Свойство:
            - Наименование - name;
            - Тип - строковый;
            - Модификатор доступа - закрытый.
        - Свойство:
            - Наименование - n;
            - Тип - целочисленный;
            - Модификатор доступа - закрытый.

- Методы:
    - Метод Child3:
        - Функционал - параметризированный конструктор со строковым и целочисленным параметром.
    - Метод Print_args:
        - Функционал - параметризированный метод, выводящий значения закрытых свойств класса на экран.

- **Класс Child4:**
    - Свойства/поля:
        - Свойство:
            - Наименование - name;
            - Тип - строковый;
            - Модификатор доступа - закрытый.
        - Свойство:
            - Наименование - n;
            - Тип - целочисленный;
            - Модификатор доступа - закрытый.
    - Методы:
        - Метод Child4:
            - Функционал - параметризированный конструктор со строковым и целочисленным параметром.
        - Метод Print_args:
            - Функционал - параметризированный метод, выводящий значения закрытых свойств класса на экран.

Иерархия наследования классов:

| № | Имя класса | Классы-наследники | Модификатор доступа при наследовании | Описание | Номер | Комментарий |
|---|---|---|---|---|---|---|
| 1 | Base | | | Базовый класс в иерархии классов | | |
| | | Child2 | public | | 2 | |
| 2 | Child2 | | | Класс объектов, подчиненных классу Base | | |
| | | Child3 | public | | 3 | |
| 3 | Child3 | | | Класс объектов, подчиненных классу Child2 | | |
| | | Child4 | public | | 4 | |
| 4 | Child4 | | | Класс объектов, подчиненных классу Child3 | | |

## Описание алгоритма

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

Функция: main

Функционал: Основной алгоритм программы

Параметры: Отсутствуют

Возвращаемое значение: Целочисленный тип данных - код возврата

Алгоритм функции представлен в таблице 2.

Таблица 2. Алгоритм функции main

| № | Предикат | Действия | № перехода | Комментарий |
|---|---|---|---|---|
| 1 | | Объявление строковой переменной name и целочисленной переменной n | 2 | |
| 2 | | Считывание с клавиатуры значений переменных name, n | 3 | |
| 3 | | Создание объекта obj класса Base путем вызова параметризированного конструктора со строковым параметром name, целочисленным параметром n | 4 | |
| 4 | | Инициализация указателя на объект par класса Base ссылкой на объект obj | 5 | |
| 5 | | Вызов метода Print_args от указателя par | 6 | |

| | | | | |
|---|---|---|---|---|
| 6 | | Приведение указателя par к классу Child2, вызов от него метода Print_args | 7 | |
| 7 | | Приведение указателя par к классу Child3, вызов от него метода Print_args | 8 | |
| 8 | | Приведение указателя par к классу Child4, вызов от него метода Print_args | ∅ | |

Конструктор класса: Child2

Модификатор доступа: public

Функционал: Параметризированный конструктор со строковым и целочисленным параметром

Параметры: Строковый параметр name, целочисленный параметр n

Алгоритм конструктора представлен в таблице 3.

Таблица 3. Алгоритм конструктора класса Child2

| № | Предикат | Действия | № перехода | Комментарий |
|---|---|---|---|---|
| 1 | | Присвоение указателю свойства name параметра name="_2" | 2 | |
| 2 | | Присвоение указателю свойства n параметра n*n | ∅ | |

Конструктор класса: Child3

Модификатор доступа: public

Функционал: Параметризированный конструктор со строковым и целочисленным параметром

Параметры: Строковый параметр name, целочисленный параметр n

Алгоритм конструктора представлен в таблице 4.

Таблица 4. Алгоритм конструктора класса Child3

| № | Предикат | Действия | № перехода | Комментарий |
|---|---|---|---|---|
| 1 | | Присвоение указателю свойства name параметра name="_3" | 2 | |
| 2 | | Присвоение указателю свойства n параметра n*n*n | ∅ | |

Конструктор класса: Child4

Модификатор доступа: public

Функционал: Параметризированный конструктор со строковым и целочисленным параметром

Параметры: Строковый параметр name, целочисленный параметр n

Алгоритм конструктора представлен в таблице 5.

Таблица 5. Алгоритм конструктора класса Child4

| № | Предикат | Действия | № перехода | Комментарий |
|---|---|---|---|---|
| 1 | | Присвоение указателю свойства name параметра name="_4" | 2 | |
| 2 | | Присвоение указателю свойства n параметра n*n*n*n | ∅ | |

Конструктор класса: Base

Модификатор доступа: public

Функционал: Параметризированный конструктор со строковым и целочисленным параметром

Параметры: Строковый параметр name, целочисленный параметр n

Алгоритм конструктора представлен в таблице 6.

Таблица 6. Алгоритм конструктора класса Base

| № | Предикат | Действия | № перехода | Комментарий |
|---|---|---|---|---|
| 1 | | Присвоение указателю свойства name параметра name="_1" | 2 | |
| 2 | | Присвоение указателю свойства n параметра n | ∅ | |

Класс объекта: Base

Модификатор доступа: public

Метод: Print_args

Функционал: Параметризированный метод, выводящий значения закрытых полей класса на экран

Параметры: Отсутствуют

Возвращаемое значение: Отсутствует

Алгоритм метода представлен в таблице 7.

Таблица 7. Алгоритм метода Print_args класса Base

| № | Предикат | Действия | № перехода | Комментарий |
|---|---|---|---|---|
| 1 | | Вывод на экран значений закрытых свойств текущего объекта | ∅ | |

Класс объекта: Child2

Модификатор доступа: public

Метод: Print_args

Функционал: Параметризированный метод, выводящий значения закрытых

полей класса на экран

Параметры: Отсутствуют

Возвращаемое значение: Отсутствует

Алгоритм метода представлен в таблице 8.

Таблица 8. Алгоритм метода Print_args класса Child2

| № | Предикат | Действия | № перехода | Комментарий |
|---|---|---|---|---|
| 1 | | Вывод на экран значений закрытых свойств текущего объекта | ∅ | |

Класс объекта: Child3

Модификатор доступа: public

Метод: Print_args

Функционал: Параметризированный метод, выводящий значения закрытых полей класса на экран

Параметры: Отсутствуют

Возвращаемое значение: Отсутствует

Алгоритм метода представлен в таблице 9.

Таблица 9. Алгоритм метода Print_args класса Child3

| № | Предикат | Действия | № перехода | Комментарий |
|---|---|---|---|---|
| 1 | | Вывод на экран значений закрытых свойств текущего | ∅ | |

| | | | | |
|---|---|---|---|---|
| | | объекта | | |

Класс объекта: Child4

Модификатор доступа: public

Метод: Print_args

Функционал: Параметризированный метод, выводящий значения закрытых полей класса на экран

Параметры: Отсутствуют

Возвращаемое значение: Отсутствует

Алгоритм метода представлен в таблице 10.

Таблица 10. Алгоритм метода Print_args класса Child4

| № | Предикат | Действия | № перехода | Комментарий |
|---|---|---|---|---|
| 1 | | Вывод на экран значений закрытых свойств текущего объекта | ∅ | |

## Блок-схема алгоритма

Представим описание алгоритмов в графическом виде на рисунках ниже.

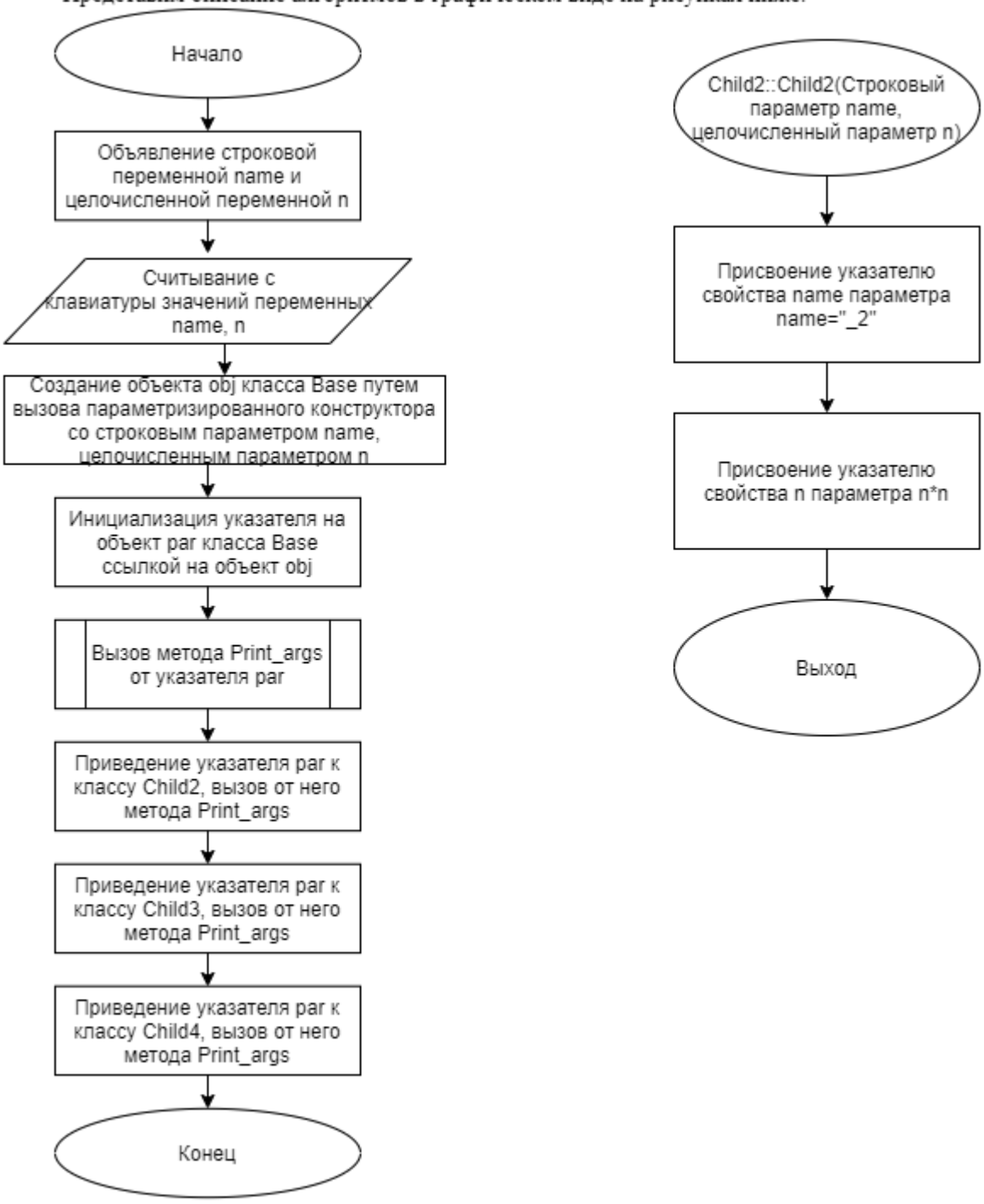


Рис. 1. Блок-схема алгоритма.

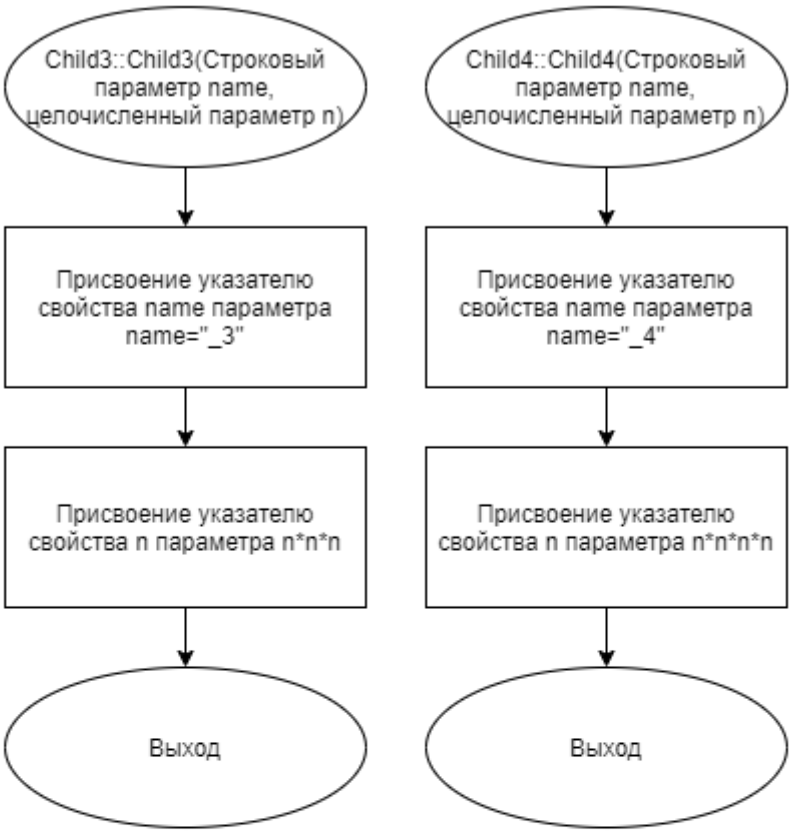


Рис. 2. Блок-схема алгоритма.

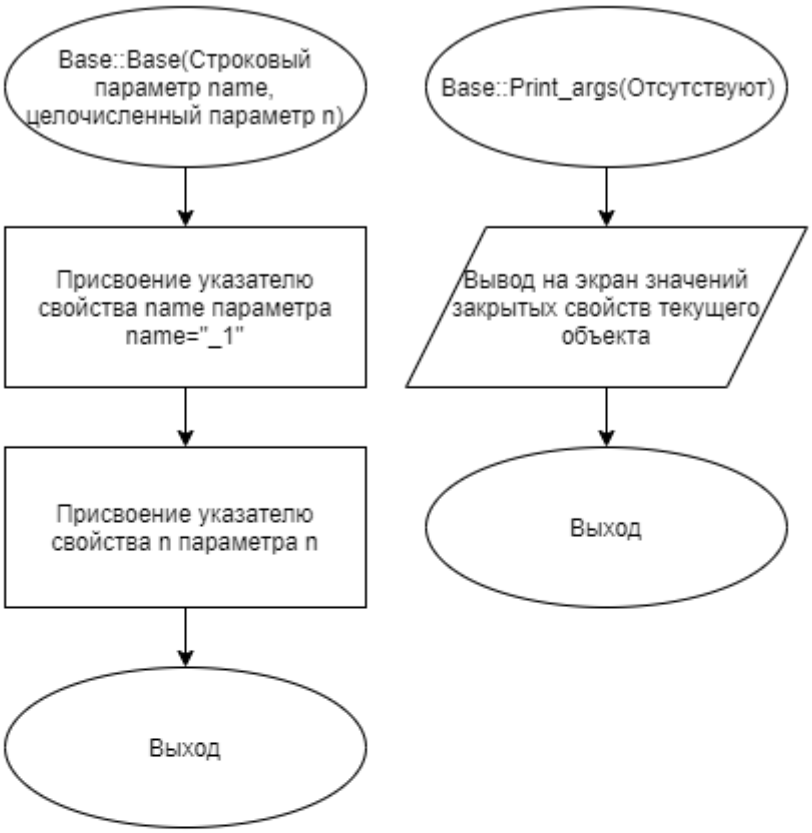


Рис. 3. Блок-схема алгоритма.

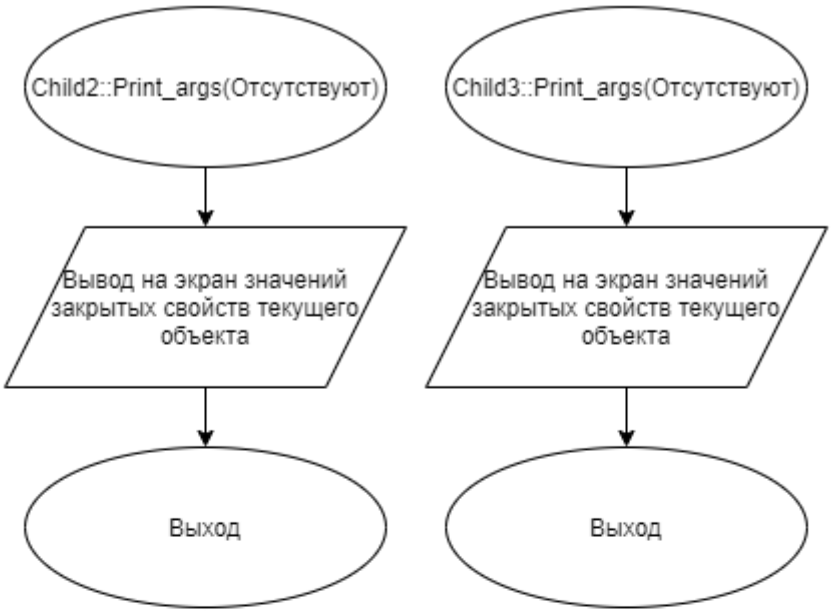


Рис. 4. Блок-схема алгоритма.


Рис. 5. Блок-схема алгоритма.

## Код программы

Программная реализация алгоритмов для решения задачи представлена ниже.

### Файл Base.cpp

```
#include "Base.h"
#include <iostream>
#include <string>
using namespace std;

Base :: Base(string name, int n)
        :name(name + "_1"), n(n)
        {}


void Base :: Print_args(){
        cout << this -> name << " " << this -> n << "\n";
}
```

### Файл Base.h

```
#ifndef BASE_H
#define BASE_H

#include <iostream>
using namespace std;

class Base{
        private:
                string name;
                int n;
        public:
                Base(string name, int n);
                void Print_args();
};

#endif
```

### Файл Child2.cpp

```
#include "Child2.h"
#include "Base.h"
#include <iostream>
#include <string>
```

```
using namespace std;

Child2 :: Child2(string name, int n)
        :Base(name, n), name(name + "_2"), n(n*n)
        {}


void Child2 :: Print_args(){
        cout << this -> name << " " << this -> n << "\n";
}
```

## Файл Child2.h

```
#ifndef CHILD2_H
#define CHILD2_H

#include "Base.h"
#include <iostream>
using namespace std;

class Child2 : public Base{
        private:
                string name;
                int n;
        public:
                Child2(string name, int n);
                void Print_args();
};

#endif
```

## Файл Child3.cpp

```
#include "Child2.h"
#include "Child3.h"
#include <iostream>
#include <string>
using namespace std;

Child3 :: Child3(string name, int n)
        :Child2(name, n), name(name + "_3"), n(n*n*n)
        {}


void Child3 :: Print_args(){
        cout << this -> name << " " << this -> n << "\n";
}
```

## Файл Child3.h

```
#ifndef CHILD3_H
#define CHILD3_H

#include "Child2.h"
#include <iostream>
using namespace std;

class Child3 : public Child2{
        private:
                string name;
                int n;
        public:
                Child3(string name, int n);
                void Print_args();
};

#endif
```

## Файл Child4.cpp

```
#include "Child3.h"
#include "Child4.h"
#include <iostream>
#include <string>
using namespace std;

Child4 :: Child4(string name, int n)
        :Child3(name, n), name(name + "_4"), n(n*n*n*n)
        {}


void Child4 :: Print_args(){
        cout << this -> name << " " << this -> n;
}
```

## Файл Child4.h

```
#ifndef CHILD4_H
#define CHILD4_H

#include "Child3.h"
#include <iostream>
using namespace std;

class Child4 : public Child3{
        private:
```

```
                string name;
                int n;
        public:
                Child4(string name, int n);
                void Print_args();
};

#endif
```

## Файл main.cpp

```
#include "Base.h"
#include "Child2.h"
#include "Child3.h"
#include "Child4.h"
#include <iostream>
#include <string>
using namespace std;

int main()
{
        string name;
        int n;

        cin >> name >> n;
        Child4 obj(name, n);

        Base* par = &obj;

        par -> Print_args();
        ((Child2*) par) -> Print_args();
        ((Child3*) par) -> Print_args();
        ((Child4*) par) -> Print_args();

        return 0;
}
```

## Тестирование

Результат тестирования программы представлен в следующей таблице.

| Входные данные | Ожидаемые выходные данные | Фактические выходные данные |
|---|---|---|
| Object 1 | Object_1 1 Object_2 1 Object_3 1 Object_4 1 | Object_1 1 Object_2 1 Object_3 1 Object_4 1 |
| Normik 3 | Normik_1 3 Normik_2 9 Normik_3 27 Normik_4 81 | Normik_1 3 Normik_2 9 Normik_3 27 Normik_4 81 |
| Object 2 | Object_1 2 Object_2 4 Object_3 8 Object_4 16 | Object_1 2 Object_2 4 Object_3 8 Object_4 16 |

# 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).