Здесь будет титульник, листай ниже

СОДЕРЖАНИЕ

1 ПОСТАНОВКА ЗАДАЧИ.....5
1.1 Описание входных данных.....7
1.2 Описание выходных данных.....8
2 МЕТОД РЕШЕНИЯ.....9
3 ОПИСАНИЕ АЛГОРИТМОВ.....11
3.1 Алгоритм метода ChangePrivate класса ParentClass.....11
3.2 Алгоритм конструктора класса ParentClass.....11
3.3 Алгоритм метода Change класса ParentClass.....12
3.4 Алгоритм метода Print класса ParentClass.....12
3.5 Алгоритм конструктора класса DerivedClass.....12
3.6 Алгоритм метода Change класса DerivedClass.....13
3.7 Алгоритм метода Print класса DerivedClass.....13
3.8 Алгоритм функции main.....14
4 БЛОК-СХЕМЫ АЛГОРИТМОВ.....16
5 КОД ПРОГРАММЫ.....21
5.1 Файл DerivedClass.cpp.....21
5.2 Файл DerivedClass.h.....21
5.3 Файл main.cpp.....22
5.4 Файл ParentClass.cpp.....23
5.5 Файл ParentClass.h.....23
6 ТЕСТИРОВАНИЕ.....25
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....26

1 ПОСТАНОВКА ЗАДАЧИ

Разработать систему, которая 1) демонстрирует возможность конструирования производного объекта на базе исходного объекта; 2) выполняет перераспределение прав доступа к элементам исходного объекта; 3) демонстрирует механизм однозначного обращения (использования, доступа) к элементам производного и исходного объекта.

Спроектировать исходный объект (разработать описание класса), который имеет элементы:

В закрытом доступе:

- одно свойство целого типа;
- метод, с одним целочисленным параметром, который меняет значение свойства в закрытом доступе на утроенное значение параметра.

В открытом доступе:

- одно свойство целого типа;
- параметризированный конструктор, с двумя целочисленными параметрами, который устанавливает значения свойств с закрытым и открытым доступом. Значение закрытого свойства меняется посредством вызова метода из закрытого раздела;
- метод с двумя целочисленными параметрами, который устанавливает значения свойств с закрытым и открытым доступом. Значение закрытого свойства меняется посредством вызова метода из закрытого доступа;
- метод, который выводит на экран значение обоих свойств. Сперва значение закрытого свойства, потом значение открытого свойства.

Спроектировать производный объект (разработать описание класса производного объекта), который содержит исходный объект и имеет элементы:

В закрытом доступе:

- одно свойство целого типа, наименование которого совпадает с наименование закрытого свойства исходного объекта;

В открытом доступе:

- одно свойство целого типа, наименование которого совпадает с наименование открытого свойства исходного объекта;
- параметризированный конструктор, с двумя целочисленными параметрами, который устанавливает значения свойств с закрытым и открытым доступом;
- метод с двумя целочисленными параметрами, который устанавливает значения свойств с закрытым и открытым доступом. Наименование метода совпадает с наименованием аналогичного метода исходного объекта;
- метод, который выводит на экран значение обоих свойств. Сперва значение свойства закрытым доступом, потом значение свойства открытым доступом. Наименование метода совпадает с наименованием аналогичного метода исходного объекта.

Производный объект спроектировать таким образом, чтобы к открытым элементам исходного объекта сохранить открытый доступ.

Алгоритм конструирования и отработки системы:

1. Добавление в состав системы двух целочисленных переменных.

2. Ввод значений двух целочисленных переменных.

3. Объявление объекта производного класса. При этом обеспечивается отработка параметризированного конструктора для исходного объекта и для производного. В качестве аргументов используются целочисленные переменные, в последовательности, как им были присвоены значения. Первый аргумент содержит значение для закрытого свойства, второй - для открытого свойства.

4. Вывод значений свойств исходного объекта.

5. Вывод значений свойств производного объекта.

6. Ввод значения одной целочисленной переменной.

7. Присвоение введенного значения открытому свойству производного объекта.

8. Присвоение введенного значения открытому свойству исходного объекта.

9. Вывод значений свойств производного объекта.

10. Вывод значений свойств родительского объекта

11. Ввод значений двух целочисленных переменных.

12. Если значение первой переменной больше нуля, то:

12.1. Переопределение значений свойств производного объекта - увеличение на единицу введенных исходных значений.

12.2. Переопределение значений свойств исходного объекта - уменьшение на единицу введенных исходных значений.

12.3. Вывод значений свойств производного объекта.

12.4. Вывод значений свойств родительского объекта.

13. Иначе:

13.1. Переопределение значений свойств родительского объекта - увеличение на единицу введенных исходных значений.

13.2. Переопределение значений свойств производного объекта - уменьшение на единицу введенных исходных значений.

13.3. Вывод значений свойств родительского объекта.

13.4. Вывод значений свойств производного объекта.

14. Завершение работы системы.

1.1 Описание входных данных

В первой строке:

«Целое число» «Целое число»

Во второй строке:

```
«Целое число»
```

В третьей строке:

```
«Целое число» «Целое число»
```

Пример ввода

```
8 5
11
-7 12
```

1.2 Описание выходных данных

Начиная с первой строки:

```
«Целое число»    «Целое число»
«Целое число»    «Целое число»
«Целое число»    «Целое число»
«Целое число»    «Целое число»
«Целое число»    «Целое число»
«Целое число»    «Целое число»
```

Пример вывода

```
24    5
8    5
8    11
24    11
-18    13
-8    11
```

2 МЕТОД РЕШЕНИЯ

Для решения задачи используется:

- объект derived класса DerivedClass предназначен для для демонстрации работы программы;
- функция main для определения входной точки программы;
- заголовочные файлы;
- классы;
- наследование.

Класс ParentClass:

- свойства/поля:
    - поле :
        - наименование — m_property;
        - тип — int;
        - модификатор доступа — private;
    - поле :
        - наименование — publicProperty;
        - тип — int;
        - модификатор доступа — public;
- функционал:
    - метод ChangePrivate — меняет значение приватного поля m_property на утроенное значение параметра;
    - метод ParentClass — инициализирует приватное и публичное поля;
    - метод Change — изменяет значения приватного и публичного полей;
    - метод Print — выводит значения приватного и публичного полей.

Класс DerivedClass:

- свойства/поля:

- поле :
    - наименование — m_property;
    - тип — int;
    - модификатор доступа — private;
- поле :
    - наименование — publicProperty;
    - тип — int;
    - модификатор доступа — public;

- функционал:
    - метод DerivedClass — инициализирует приватное и публичное поля;
    - метод Change — изменяет значения приватного и публичного полей;
    - метод Print — выводит значения приватного и публичного полей.

Таблица 1 – Иерархия наследования классов

№	Имя класса	Классы-наследники	Модификатор доступа при наследовании	Описание	Номер
1	ParentClass				
		DerivedClass	public		2
2	DerivedClass				

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 метода ChangePrivate класса ParentClass

Функционал: Меняет значение приватного поля m_property на утроенное значение параметра.

Параметры: int num.

Возвращаемое значение: void.

Алгоритм метода представлен в таблице 2.

Таблица 2 – Алгоритм метода ChangePrivate класса ParentClass

№	Предикат	Действия	№ перехода
1		Инициализация m_property = num * 3	∅

3.2 Алгоритм конструктора класса ParentClass

Функционал: Инициализирует приватное и публичное поля.

Параметры: int privateNum, int publicNum.

Алгоритм конструктора представлен в таблице 3.

Таблица 3 – Алгоритм конструктора класса ParentClass

№	Предикат	Действия	№ перехода
1		Вызов this->ChangePrivate(privateNum)	2
2		Инициализация publicProperty = publicNum	∅

3.3 Алгоритм метода Change класса ParentClass

Функционал: Изменяет значения приватного и публичного полей.

Параметры: int privateNum, int publicNum.

Возвращаемое значение: void.

Алгоритм метода представлен в таблице 4.

Таблица 4 – Алгоритм метода Change класса ParentClass

№	Предикат	Действия	№ перехода
1		Вызов this->ChangePrivate(privateNum)	2
2		Инициализация publicProperty = publicNum	∅

3.4 Алгоритм метода Print класса ParentClass

Функционал: Выводит значения приватного и публичного полей.

Параметры: нет.

Возвращаемое значение: void.

Алгоритм метода представлен в таблице 5.

Таблица 5 – Алгоритм метода Print класса ParentClass

№	Предикат	Действия	№ перехода
1		Вывод "m_property publicProperty"	∅

3.5 Алгоритм конструктора класса DerivedClass

Функционал: Инициализирует приватное и публичное поля.

Параметры: нет.

Алгоритм конструктора представлен в таблице 6.

Таблица 6 – Алгоритм конструктора класса DerivedClass

№	Предикат	Действия	№ перехода
1		Инициализация m_property = privateNum	2
2		Инициализация publicProperty = publicNum	∅

3.6 Алгоритм метода Change класса DerivedClass

Функционал: Изменяет значения приватного и публичного полей.

Параметры: int privateNum, int publicNum.

Возвращаемое значение: void.

Алгоритм метода представлен в таблице 7.

Таблица 7 – Алгоритм метода Change класса DerivedClass

№	Предикат	Действия	№ перехода
1		Инициализация m_property = privateNum	2
2		Инициализация publicProperty = publicNum	∅

3.7 Алгоритм метода Print класса DerivedClass

Функционал: Выводит значения приватного и публичного полей.

Параметры: нет.

Возвращаемое значение: void.

Алгоритм метода представлен в таблице 8.

Таблица 8 – Алгоритм метода Print класса DerivedClass

№	Предикат	Действия	№ перехода
1		Вывод "m_property publicProperty"	∅

3.8 Алгоритм функции main

Функционал: Входная точка программы.

Параметры: нет.

Возвращаемое значение: int.

Алгоритм функции представлен в таблице 9.

Таблица 9 – Алгоритм функции main

№	Предикат	Действия	№ перехода
1		Объявление целочисленных переменных privateNum, publicNum	2
2		Ввод privateNum, publicNum	3
3		Создание объекта derived класса DerivedClass с передачей privateNum и publicNum в качестве параметров	4
4		Вызов derived.ParentClass::Print()	5
5		Вызов derived.Print()	6
6		Ввод publicNum	7
7		Инициализация publicProperty = publicNum	8
8		Инициализация derived.ParentClass::publicProperty = publicNum	9
9		Вызов derived.Print()	10
10		Вызов derived.ParentClass::Print()	11
11		Ввод privateNum, publicNum	12
12	privateNum > 0	Вызов derived.Change(privateNum + 1, publicNum + 1)	13
		Вызов derived.ParentClass::Change(privateNum + 1, publicNum + 1)	16
13		Вызов derived.ParentClass::Change(privateNum - 1, publicNum - 1)	14

№	Предикат	Действия	№ перехода
14		Вызов derived.Print()	15
15		Вызов derived.ParentClass::Print()	∅
16		Вызов derived.ParentClass::Change(privateNum + 1, publicNum + 1)	17
17		Вызов derived.Change(privateNum - 1, publicNum - 1)	18
18		derived.ParentClass::Print()	19
19		derived.Print()	∅

4 БЛОК-СХЕМЫ АЛГОРИТМОВ

Представим описание алгоритмов в графическом виде на рисунках 1-5.

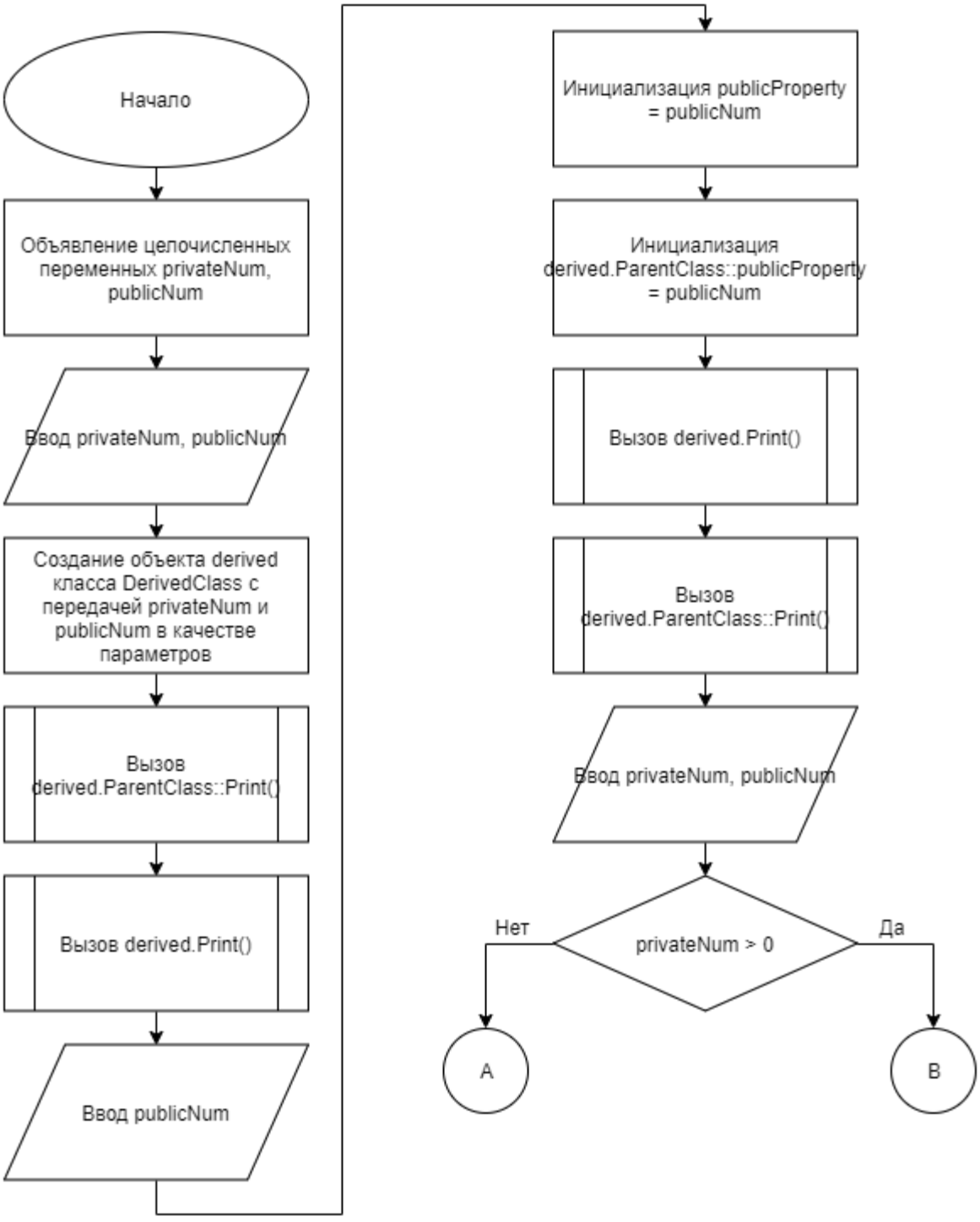

Рисунок 1 – Блок-схема алгоритма

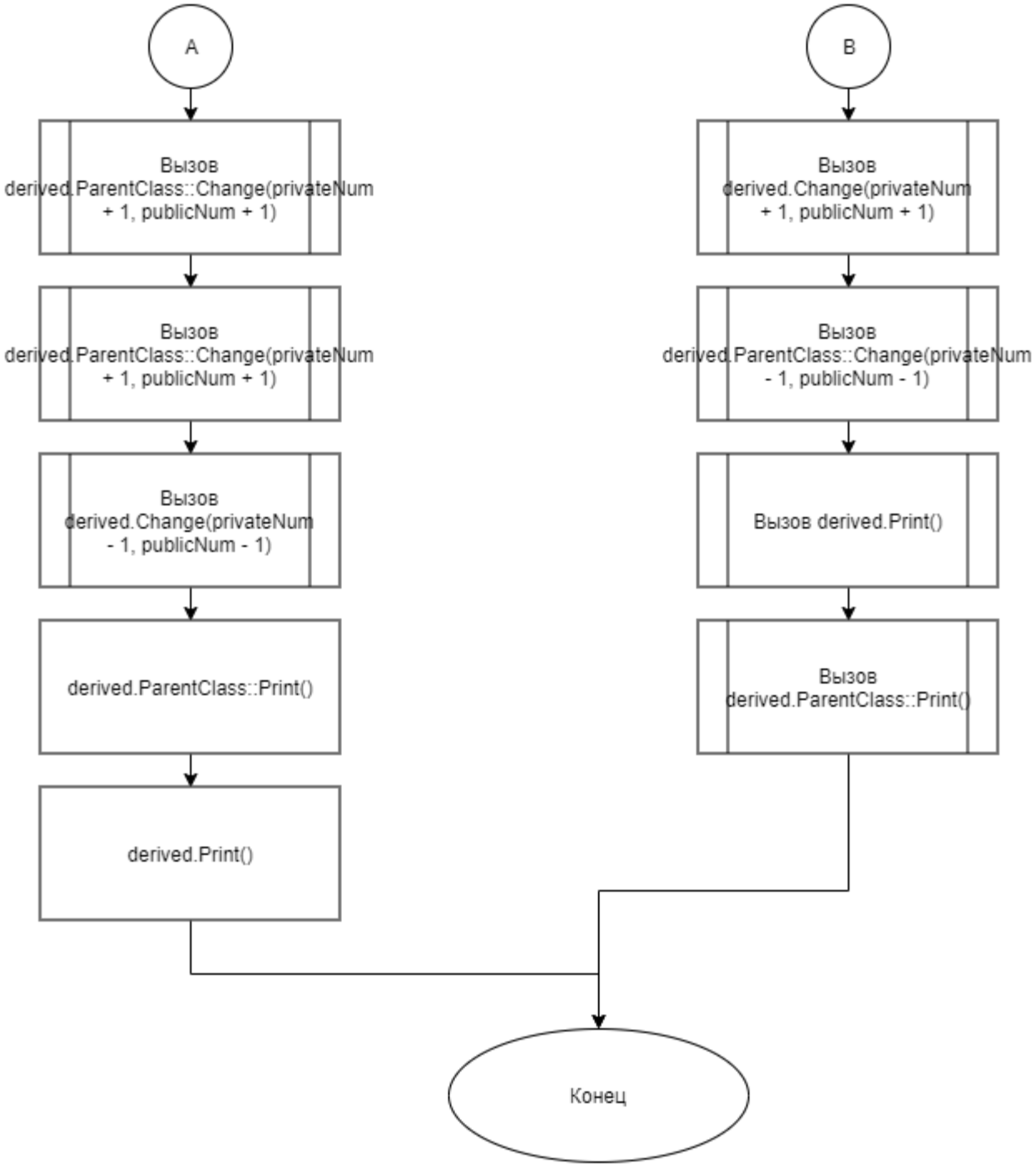

Рисунок 2 – Блок-схема алгоритма

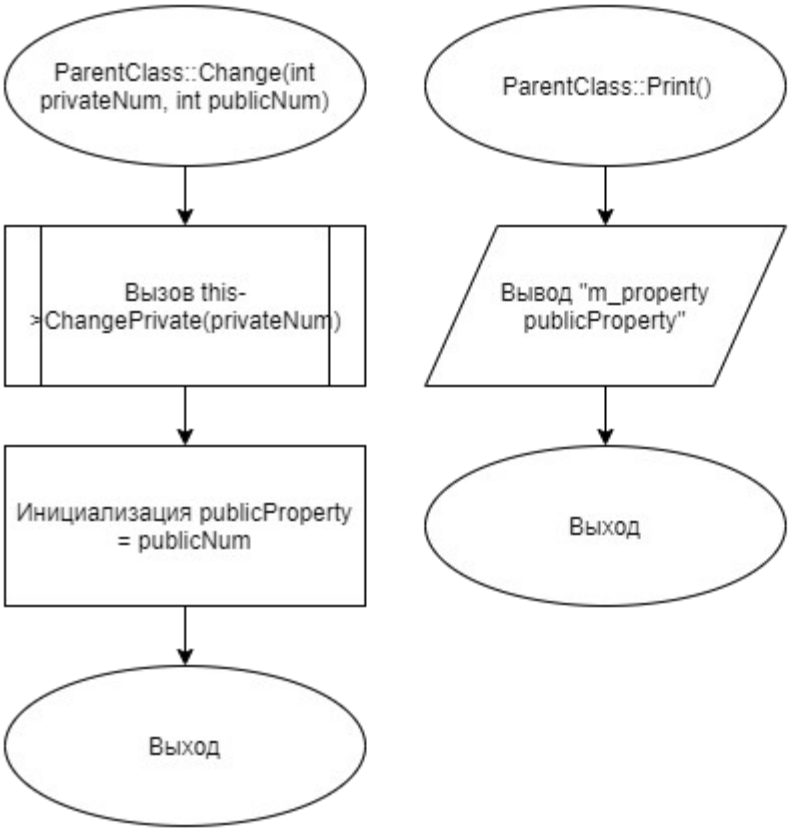

Рисунок 3 – Блок-схема алгоритма

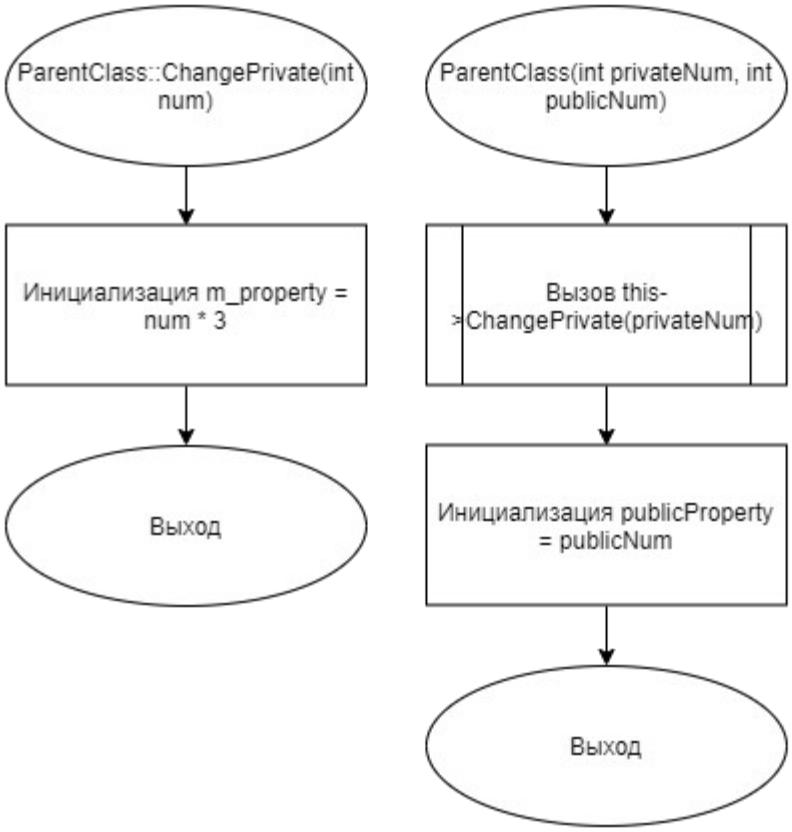

Рисунок 4 – Блок-схема алгоритма

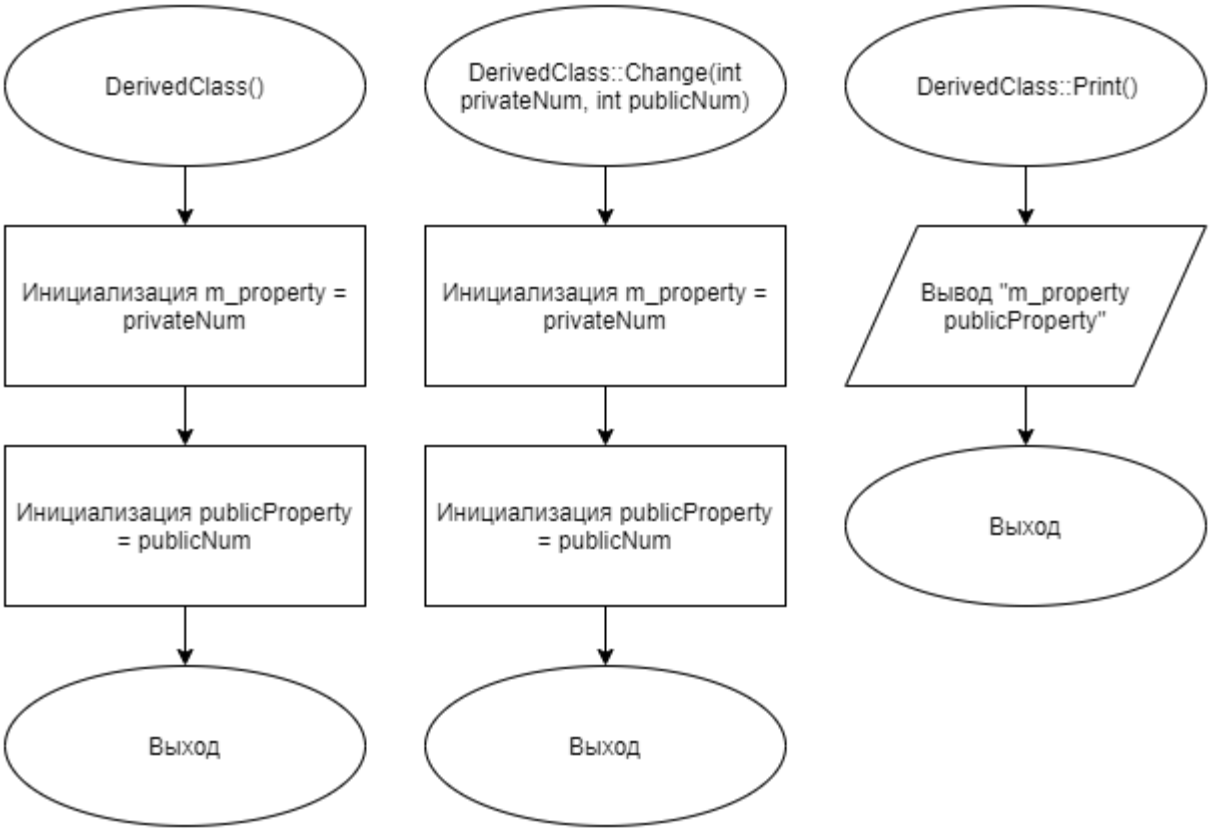

Рисунок 5 – Блок-схема алгоритма

5 КОД ПРОГРАММЫ

Программная реализация алгоритмов для решения задачи представлена ниже.

5.1 Файл DerivedClass.cpp

Листинг 1 – DerivedClass.cpp

```
#include "DerivedClass.h"

DerivedClass::DerivedClass(int privateNum, int publicNum) :
ParentClass(privateNum, publicNum)
{
   m_property = privateNum;
   publicProperty = publicNum;
}

void DerivedClass::Change(int privateNum, int publicNum)
{
   m_property = privateNum;
   publicProperty = publicNum;
}

void DerivedClass::Print()
{
   std::cout << m_property << "    " << publicProperty << std::endl;
}
```

5.2 Файл DerivedClass.h

Листинг 2 – DerivedClass.h

```
#ifndef __DERIVEDCLASS__H
#define __DERIVEDCLASS__H

#include "ParentClass.h"

class DerivedClass : public ParentClass {
   int m_property;
public:
   int publicProperty;
```

```
    DerivedClass(int, int);
    void Change(int privateNum, int publicNum);
    void Print();
};

#endif
```

5.3 Файл main.cpp

Листинг 3 – main.cpp

```
#include "ParentClass.h"
#include "DerivedClass.h"

int main()
{
    int privateNum, publicNum;
    std::cin >> privateNum >> publicNum;

    DerivedClass derived(privateNum, publicNum);

    derived.ParentClass::Print();
    derived.Print();

    std::cin >> publicNum;
    derived.publicProperty = publicNum;
    derived.ParentClass::publicProperty = publicNum;

    derived.Print();
    derived.ParentClass::Print();

    std::cin >> privateNum >> publicNum;

    if (privateNum > 0)
    {
        derived.Change(privateNum + 1, publicNum + 1);
        derived.ParentClass::Change(privateNum - 1, publicNum - 1);
        derived.Print();
        derived.ParentClass::Print();
    }
    else
    {
        derived.ParentClass::Change(privateNum + 1, publicNum + 1);
        derived.Change(privateNum - 1, publicNum - 1);
        derived.ParentClass::Print();
        derived.Print();
    }

    return 0;
}
```

5.4 Файл ParentClass.cpp

Листинг 4 – ParentClass.cpp

```
#include "ParentClass.h"

void ParentClass::ChangePrivate(int num)
{
   m_property = num * 3;
}

ParentClass::ParentClass(int privateNum, int publicNum)
{
   this->ChangePrivate(privateNum);
   publicProperty = publicNum;
}

void ParentClass::Change(int privateNum, int publicNum)
{
   this->ChangePrivate(privateNum);
   publicProperty = publicNum;
}

void ParentClass::Print()
{
   std::cout << m_property << "    " << publicProperty << std::endl;
}
```

5.5 Файл ParentClass.h

Листинг 5 – ParentClass.h

```
#ifndef __PARENTCLASS__H
#define __PARENTCLASS__H

#include <iostream>

class ParentClass {
   int m_property;
   void ChangePrivate(int num);
public:
   int publicProperty;

   ParentClass(int, int);
   void Change(int privateNum, int publicNum);
```

```
        void Print();
    };

    #endif
```

6 ТЕСТИРОВАНИЕ

Результат тестирования программы представлен в таблице 10.

Таблица 10 – Результат тестирования программы

Входные данные	Ожидаемые выходные данные	Фактические выходные данные
8 5 11 -7 12	24 5 8 5 8 11 24 11 -18 13 -8 11	24 5 8 5 8 11 24 11 -18 13 -8 11

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).