Здесь будет титульник, листай ниже

СОДЕРЖАНИЕ

1 ПОСТАНОВКА ЗАДАЧИ

Описать класс cl_parent объекта, в котором следующий состав элементов:

В закрытом разделе:

− одно свойство целого типа;

− метод, с одним целочисленным параметром. который меняет значение свойства в закрытом разделе на удвоенное значение параметра.

В открытом разделе:

− одно свойство целого типа;

− параметризированный конструктор, с двумя целочисленными параметрами, который устанавливает значения свойств в закрытом и открытом разделе. Значение закрытого свойства меняется посредством вызова метода из закрытого раздела;

− метод с двумя целочисленными параметрами, который устанавливает значения свойств в закрытом и открытом разделе. Значение закрытого свойства меняется посредством вызова метода из закрытого раздела;

− метод, который выводит на экран значение обоих свойств. Сперва значение закрытого свойства, потом значение открытого свойства.

Назовем объект данного класса родительским. Соответственно его класс родительским классом.

На базе родительского объекта сконструируем производный объект. Производный объект должен сохранить открытый доступ к открытым элементам родительского класса. Он должен иметь следующие собственные элементы:

В закрытом разделе:

− одно свойство целого типа, наименование которого совпадает с наименование закрытого свойства родительского объекта;

В открытом разделе:

− одно свойство целого типа, наименование которого совпадает с наименование открытого свойства родительского объекта;

− параметризированный конструктор, с двумя целочисленными параметрами, который устанавливает значения свойств в закрытом и открытом разделе;

− метод с двумя целочисленными параметрами, который устанавливает значения свойств в закрытом и открытом разделе. Наименование метода совпадает с наименованием аналогичного метода родительского объекта;

− метод, который выводит на экран значение обоих свойств. Сперва значение закрытого свойства, потом значение открытого свойства. Наименование метода совпадает с наименованием аналогичного метода родительского объекта.

Разработать производный класс используя класс cl_parent в качестве родительского.

В основной функции реализовать алгоритм:

1. Ввод значения двух целочисленных переменных.

2. Создать объект производного класса используя целочисленных переменных в конструкторе в качестве аргументов в последовательности, как им были присвоены значения. Первый аргумент содержит значение для свойства закрытого раздела/,/ второй для свойства открытого раздела.

3. Вывод значений свойств родительского объекта.

4. Вывод значений свойств производного объекта.

5. Если исходное значение закрытого свойства больше нуля, то:

5.1. Переопределить значения свойств производного объекта, увеличив на единицу введенные исходные значения.

5.2. Переопределить значения свойств родительского объекта, уменьшив на единицу введенные исходные значения.

5.3. Вывод значений свойств производного объекта.

5.4. Вывод значений свойств родительского объекта.

6. Иначе:

6.1. Переопределить значения свойств родительского объекта, увеличив на единицу введенные исходные значения.

6.2. Переопределить значения свойств производного объекта, уменьшив на единицу введенные исходные значения.

6.3. Вывод значений свойств родительского объекта.

6.4. Вывод значений свойств производного объекта.

1.1 Описание входных данных

В первой строке:

«Целое число» «Целое число»

Пример ввода:

```
8 5
```

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»

Пример вывода:

```
16   5
8   5
9   6
14   4
```

2 МЕТОД РЕШЕНИЯ

Для решения задачи используется:

- объект child класса Child предназначен для экземпляр дочернего класса 'Child', наследование функциональности и свойств родительского класса 'Parent';
- объект cout класса объекты потокового вывода предназначен для функционирования системы;
- объект cin класса объекты потокового ввода предназначен для функционирования системы;
- оператор return - возврат значения из функции.

Класс Child:

- свойства/поля:
    - поле приватное целочисленное поле, которое добавляется дочерним классом:
        - наименование — private_value;
        - тип — целочисленный;
        - модификатор доступа — private;
    - поле унаследованное публичное поле от родительского класса 'Parent', которое представляет публичное значение объекта дочернего класса:
        - наименование — public_value;
        - тип — целочисленный;
        - модификатор доступа — public;
- функционал:
    - метод Child — конструктор;
    - метод change_values — изменение значений приватного и

публичного членов объекта дочернего класса;

- метод show_values — отображение значений приватного и публичного членов объекта дочернего класса.

Класс Parent:

- свойства/поля:
  - поле приватное целочисленное поле, которое представляет приватное значение объекта родительского класса:
    - наименование — private_value;
    - тип — целочисленный;
    - модификатор доступа — private;
  - поле публичное целочисленное поле, которое представляет публичное значение объекта родительского класса:
    - наименование — public_value;
    - тип — целочисленный;
    - модификатор доступа — public;
- функционал:
  - метод Parent — конструктор;
  - метод show_values — отображение значений приватного и публичного членов объекта родительского класса;
  - метод change_values — изменение значений приватного и публичного членов объекта родительского класса;
  - метод change_private_value — изменение значения приватного члена объекта родительского класса.

Таблица 1 – Иерархия наследования классов

№	Имя класса	Классы-наследники	Модификатор доступа при наследовании	Описание	Номер
1	Child			представление дочерних объектов, которые наследуют функциональность	

№	Имя класса	Классы-наследники	Модификатор доступа при наследовании	Описание	Номер
				и свойства от родительского класса 'Parent'	
2	Parent			определение общей функциональности и свойств, базовая реализация методов и переменных	

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: основной алгоритм работы программы.

Параметры: отсутсвуют.

Возвращаемое значение: int - индикатор корректности завершения работы программы.

Алгоритм функции представлен в таблице 2.

Таблица 2 – Алгоритм функции main

№	Предикат	Действия	№ перехода
1		инициализация целочисленных переменных privateInt, publicInt	2
2		ввод значений переменных privateInt, publicInt	3
3		создание объекта 'child' типа 'Child', используя конструктор класса 'Child' с передачей значений 'privateInt' и 'publicInt' в качестве аргументов	4
4		вызов метода 'show_values' из класса 'Parent' объекта 'child'	5
5		переход на следующую строку	6
6		вызов метода 'show_values()' объекта 'child'	7
7		переход на следующую строку	8
8	privateInt > 0	вызов метода 'change_values' объекта 'child', который увеличивает значение приватного и публичного членов объектов 'child', передача в	9

№	Предикат	Действия	№ перехода
		метод новых значений 'pritvateInt + 1' и 'publicInt + 1'	
			13
9		вызов метода 'change_values' из класса 'Parent' объекта 'child', изменение приватного и публичного членов объекта 'child', используя переданные значения 'privateInt - 1' и 'publicInt - 1'	10
10		вызов метода 'show_values()' , вывод измененных значений приватного и публичного членов объекта 'child', которые были изменены методом 'change_values()' ранее	11
11		переход на следующую строку	12
12		вызов метода 'show_values' из класса 'Parent' объекта 'child'	∅
13		вызов метода 'change_values' из класса 'Parent' объекта 'child', изменение приватного и публичного членов объекта 'child', используя переданные значения 'privateInt + 1' и 'publicInt + 1'	14
14		вызов метода 'change_values' объекта 'child', который увеличивает значение приватного и публичного членов объектов 'child', передача в метод новых значений 'pritvateInt - 1' и 'publicInt - 1'	15
15		вызов метода 'show_values' из класса 'Parent' объекта 'child'	16
16		переход на следующую строку	17
17		вызов метода 'show_values()' , вывод измененных значений приватного и публичного членов объекта 'child', которые были изменены методом	∅

№	Предикат	Действия	№ перехода
		'change_values()' ранее	

3.2 Алгоритм конструктора класса Parent

Функционал: конструктор.

Параметры: privateInt, publicInt - изменяемые значения, используются для установки членов объекта родительского класса.

Алгоритм конструктора представлен в таблице 3.

Таблица 3 – Алгоритм конструктора класса Parent

№	Предикат	Действия	№ перехода
1		вызов метода 'change_private_value()', который изменяет значение приватного члена объекта родительского класса на значение, переданное в аргументе 'privateInt'	2
2		значение 'publicInt' присваивается публичному члену объекта родительского класса 'Parent', устанавливая его значение	∅

3.3 Алгоритм метода show_values класса Parent

Функционал: отображение значений приватного и публичного членов объекта родительского класса.

Параметры: отсутствуют.

Возвращаемое значение: отсутствует.

Алгоритм метода представлен в таблице 4.

Таблица 4 – Алгоритм метода show_values класса Parent

№	Предикат	Действия	№ перехода
1		вывод на экран значения приватного члена, " ", публичного члена	∅

№	Предикат	Действия	№ перехода
		объекта родительского класса 'Parent'	

3.4 Алгоритм метода change_values класса Parent

Функционал: изменение значений приватного и публичного членов объекта родительского класса.

Параметры: int privateInt, publicInt - изменяемые значения, используются для установки членов объекта родительского класса.

Возвращаемое значение: отсутствует.

Алгоритм метода представлен в таблице 5.

Таблица 5 – Алгоритм метода change_values класса Parent

№	Предикат	Действия	№ перехода
1		вызов метода 'change_private_value()', который изменяет значение приватного члена объекта родительского класса на значение, переданное в аргументе 'privateInt'	2
2		значение 'publicInt' присваивается публичному члену объекта родительского класса 'Parent', устанавливая его значение	∅

3.5 Алгоритм метода change_private_value класса Parent

Функционал: изменение значения приватного члена объекта родительского класса.

Параметры: int privateInt.

Возвращаемое значение: отсутствует.

Алгоритм метода представлен в таблице 6.

Таблица 6 – Алгоритм метода change_private_value класса Parent

№	Предикат	Действия	№ перехода
1		присвоение нового значения приватному члену объекта родительского класса, новое значение вычисляется как удвоенное значение 'privateInt', которое было передано в метод 'change_private_value()'	∅

3.6 Алгоритм метода change_values класса Child

Функционал: изменение значений приватного и публичного членов объекта дочернего класса.

Параметры: privateInt, publicInt - изменяемые значения, используются для установки членов объекта родительского класса.

Возвращаемое значение: отсутствует.

Алгоритм метода представлен в таблице 7.

Таблица 7 – Алгоритм метода change_values класса Child

№	Предикат	Действия	№ перехода
1		значение 'privateInt' присваивается приватному члену объекта дочернего класса 'Child', устанавливая его значение	2
2		значение 'publicInt' присваивается публичному члену объекта дочернего класса 'Child', устанавливая его значение	∅

3.7 Алгоритм метода show_values класса Child

Функционал: отображение значений приватного и публичного членов объекта дочернего класса.

Параметры: отсутствуют.

Возвращаемое значение: отсутствует.

Алгоритм метода представлен в таблице 8.

Таблица 8 – Алгоритм метода show_values класса Child

№	Предикат	Действия	№ перехода
1		вывод на экран значения приватного члена, " ", публичного члена объекта дочернего класса 'Child'	∅

3.8 Алгоритм конструктора класса Child

Функционал: конструктор.

Параметры: privateInt, publicInt - унаследованные от родительского класса , изменяемые значения .

Алгоритм конструктора представлен в таблице 9.

Таблица 9 – Алгоритм конструктора класса Child

№	Предикат	Действия	№ перехода
1		значение 'privateInt' присваивается приватному члену объекта дочернего класса 'Child', устанавливая его значение	2
2		значение 'publicInt' присваивается публичному члену объекта дочернего класса 'Child', устанавливая его значение	∅

4 БЛОК-СХЕМЫ АЛГОРИТМОВ

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

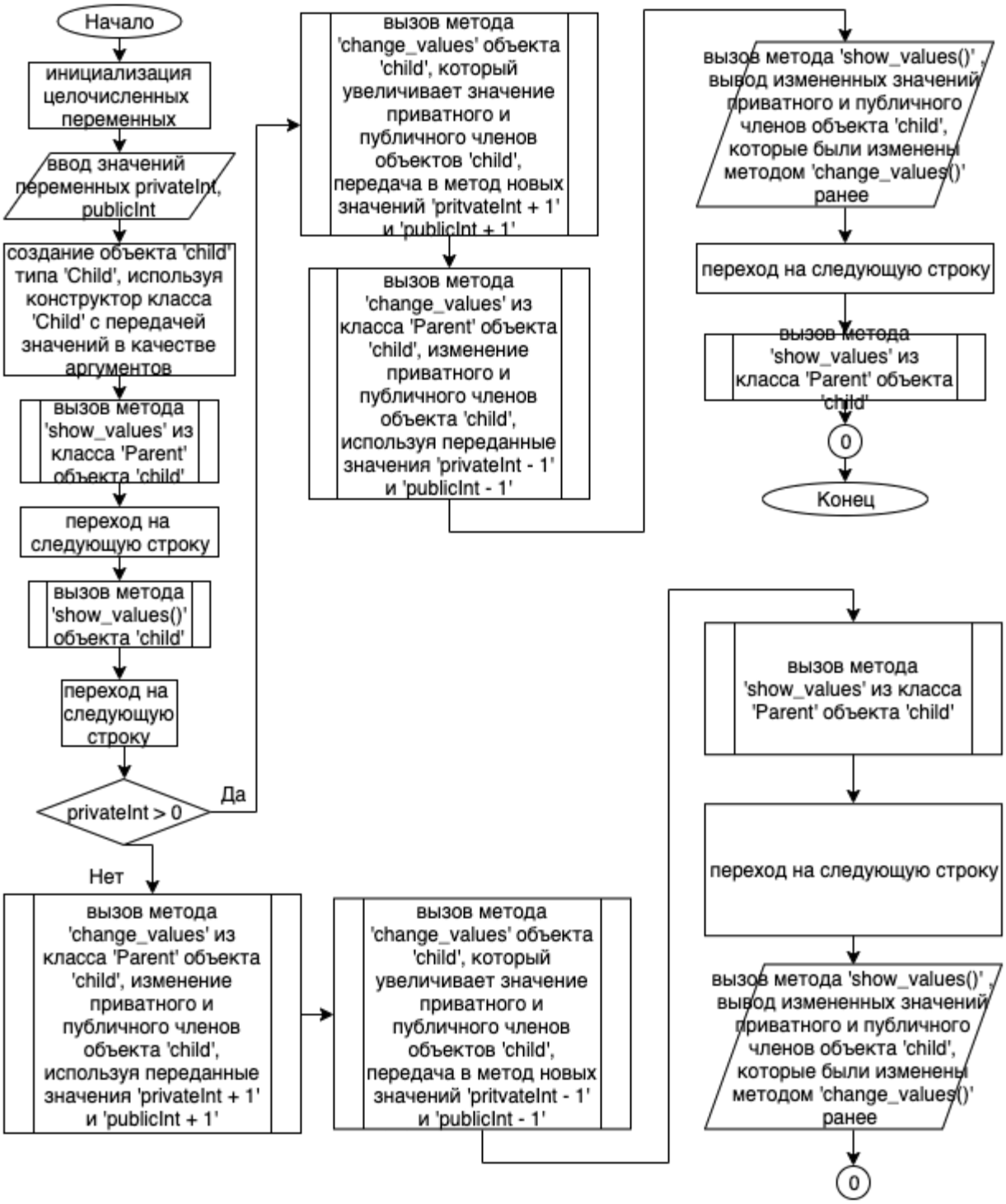

Рисунок 1 – Блок-схема алгоритма

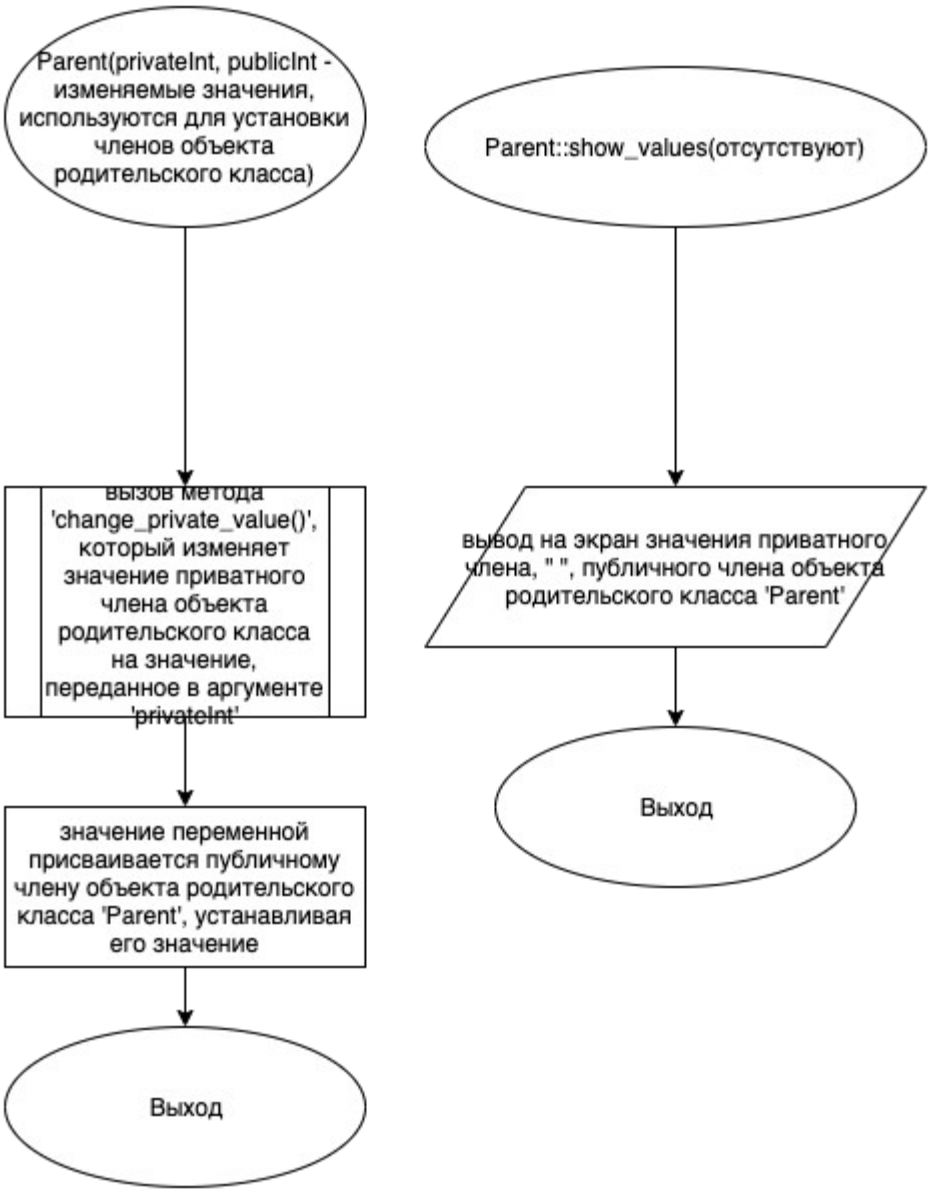

Рисунок 2 – Блок-схема алгоритма

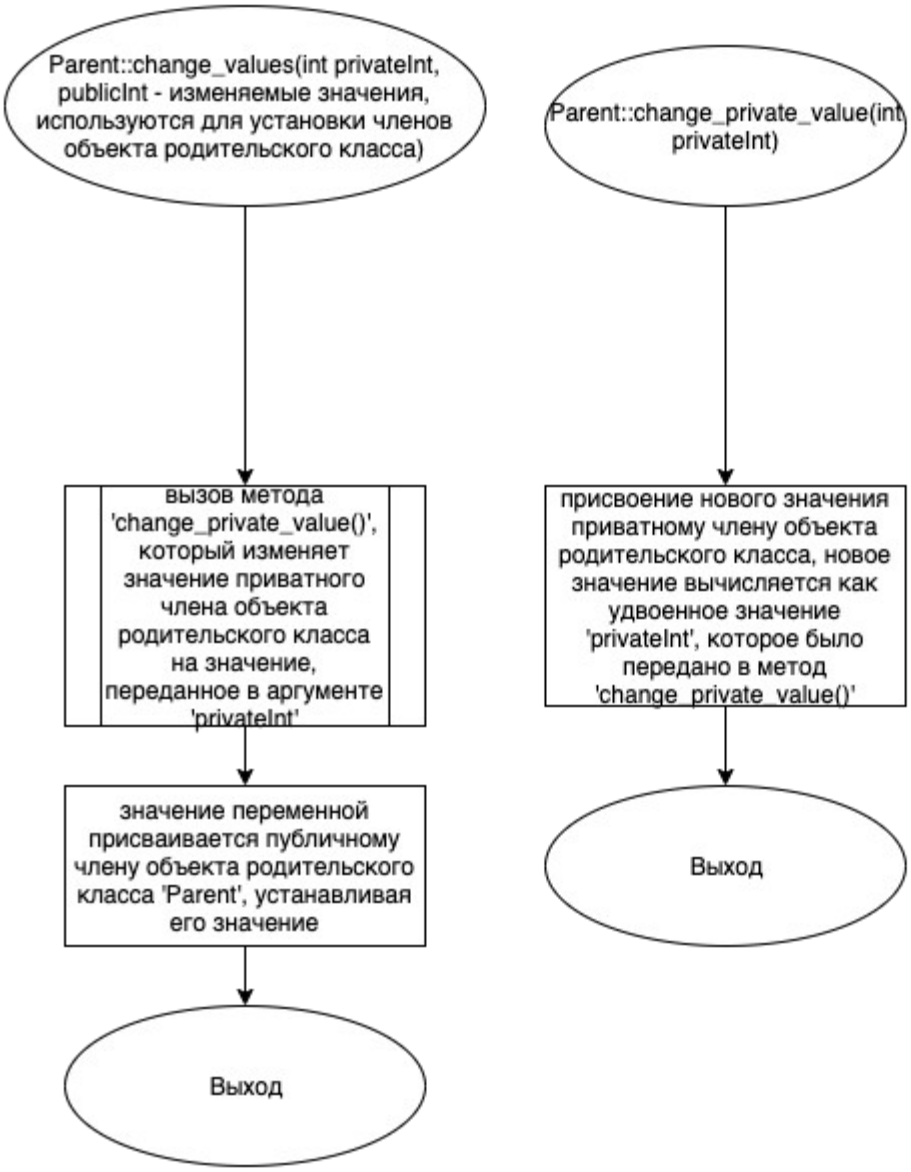

Рисунок 3 – Блок-схема алгоритма

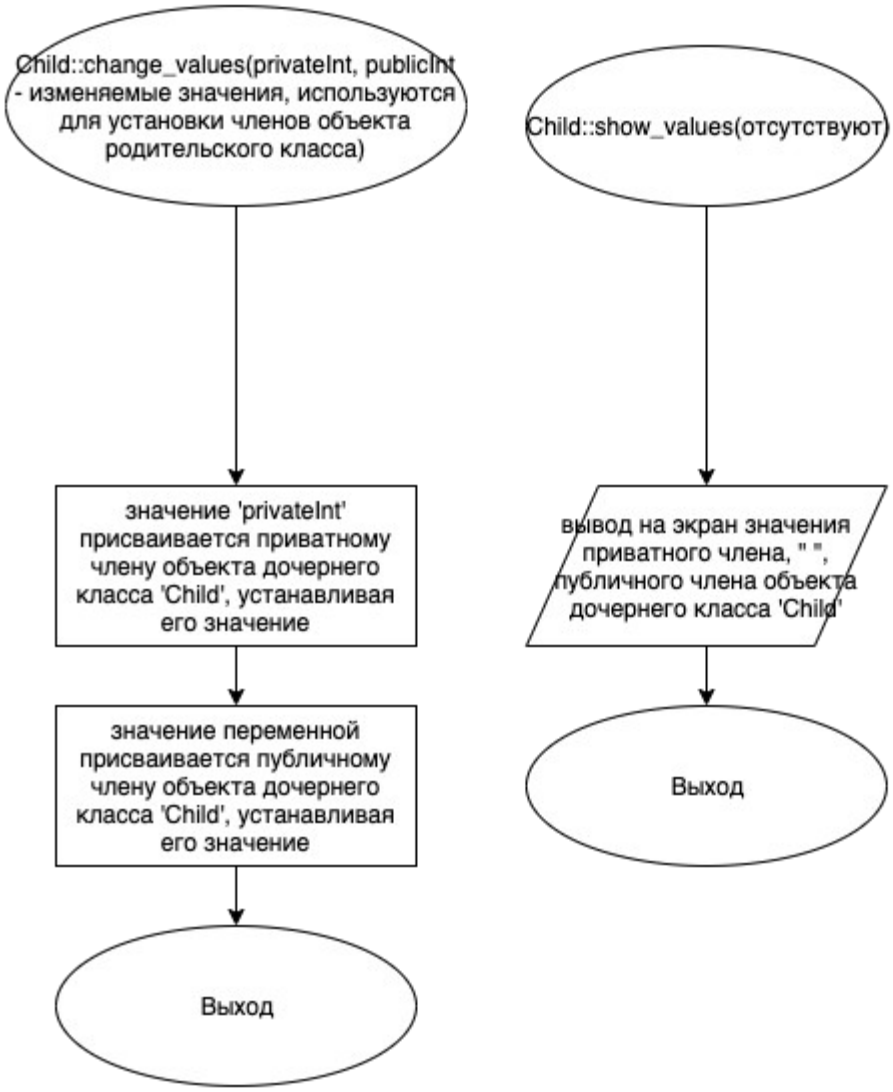

Рисунок 4 – Блок-схема алгоритма

Рисунок 5 – Блок-схема алгоритма

5 КОД ПРОГРАММЫ

Программная реализация алгоритмов для решения задачи представлена ниже.

5.1 Файл Child.cpp

Листинг 1 – Child.cpp

```
#include "Child.h"

#include <string>
#include <iostream>

using namespace std;

Child::Child(int privateInt, int publicInt) : Parent(privateInt, publicInt)
{
   private_value = privateInt;
   public_value = publicInt;
}

void Child::show_values() {
   //Parent::show_values();
   //cout << "    " << private_value;
   cout << private_value << "    " << public_value;
}

void Child::change_values(int privateInt, int publicInt) {
   //Parent::change_values(privateInt, publicInt);
   private_value = privateInt;
   public_value = publicInt;
}
```

5.2 Файл Child.h

Листинг 2 – Child.h

```
#ifndef CHILD_H
#define CHILD_H

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

```
#include <sstream>

class Child : public Parent {
public:
   Child(int, int);

   void show_values();
   void change_values(int, int);

   int public_value;
private:
   int private_value;
};

#endif
```

5.3 Файл main.cpp

Листинг 3 – main.cpp

```
#include <iostream>

#include "Parent.h"
#include "Child.h"

#include <cmath>

using namespace std;

int main() {
   int privateInt, publicInt;
   cin >> privateInt >> publicInt;

   Child child(privateInt, publicInt);
   //Parent parent = (Parent) child;

   //parent.show_values();
   //cout << endl;
   child.Parent::show_values();
   cout << endl;


   child.show_values();
   //cout << endl;


   //child.change_values(privateInt + 1, publicInt + 1);
   cout << endl;

   if (privateInt > 0) {
```

```
            child.change_values(privateInt + 1, publicInt + 1);
            //parent.change_values(privateInt - 1, publicInt - 1);

            child.Parent::change_values(privateInt - 1, publicInt - 1);

            child.show_values();
            cout << endl;

            child.Parent::show_values();
         } else {
            child.Parent::change_values(privateInt + 1, publicInt + 1);
            //parent.change_values(privateInt + 1, publicInt + 1);
            child.change_values(privateInt - 1, publicInt - 1);

            //parent.show_values();
            //cout << endl;

            child.Parent::show_values();
            cout << endl;

            child.show_values();
         }

         return(0);
      }
```

5.4 Файл Parent.cpp

Листинг 4 – Parent.cpp

```
      #include "Parent.h"

      #include <string>
      #include <iostream>

      using namespace std;

      Parent::Parent(int privateInt, int publicInt) {
         change_private_value(privateInt);
         public_value = publicInt;
      }

      void Parent::show_values() {
         cout << private_value << "    " << public_value;
      }

      void Parent::change_values(int privateInt, int publicInt) {
         change_private_value(privateInt);
         public_value = publicInt;
      }
```

```
void Parent::change_private_value(int privateInt) {
   this->private_value = privateInt * 2;
}
```

5.5 Файл Parent.h

Листинг 5 – Parent.h

```
#ifndef PARENT_H
#define PARENT_H
#include <iostream>

using namespace std;

class Parent {
public:
   Parent(int, int);

   void show_values();
   void change_values(int, int);

   int public_value;
private:
   void change_private_value(int);

   int private_value;
};

#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	16 5 8 5 9 6 14 4	16 5 8 5 9 6 14 4
8 6	16 6 8 6 9 7 14 5	16 6 8 6 9 7 14 5

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).