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 base класса cl_heritage предназначен для демострации работы наследования;
- cin - объект стандартного потока;
- cout - объект стандартного потока.

Класс cl_basic:

- свойства/поля:
    - поле демонстрация работы с открытыми полми:
        - наименование — public_value;
        - тип — int;
        - модификатор доступа — public;
    - поле демонстрация работы с закрытыми полями:
        - наименование — private_value;
        - тип — int;
        - модификатор доступа — private;
- функционал:
    - метод ShowValues — вывод значений полей;
    - метод ChangeValues — изменение значений полей;
    - метод ChangePrivateValue — изменение значения приватного поля;
    - метод cl_basic — конструктор класса.

Класс cl_heritage:

- свойства/поля:
    - поле демонстрация работы с открытыми полями:
        - наименование — public_value;
        - тип — int;

        - модификатор доступа — public;
    - поле демонстрация работы с закрытыми полями:
        - наименование — private_value;
        - тип — int;
        - модификатор доступа — private;
- функционал:
    - метод ShowValues — выод значения полей;
    - метод ChangeValues — изменение значения полей;
    - метод cl_heritage — конструктор.

Таблица 1 – Иерархия наследования классов

№	Имя класса	Классы-наследники	Модификатор доступа при наследовании	Описание	Номер
1	cl_basic			базовый класс	
		cl_heritage	public		2
2	cl_heritage			класс демонстрирующий работу наследования	

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 ShowValues класса cl_basic

Функционал: вывод значений полей.

Параметры: нет.

Возвращаемое значение: нет.

Алгоритм метода представлен в таблице 2.

Таблица 2 – Алгоритм метода ShowValues класса cl_basic

№	Предикат	Действия	№ перехода
1		вывод значения закрытого поля, вывод значения открытого поля, переход на новую строчку	∅

3.2 Алгоритм метода ChangeValues класса cl_basic

Функционал: изменение значений полей.

Параметры: int, value_1, значение передаваемое закрытому полю; int, value_2, значение передаваемое открытому полю.

Возвращаемое значение: нет.

Алгоритм метода представлен в таблице 3.

Таблица 3 – Алгоритм метода ChangeValues класса cl_basic

№	Предикат	Действия	№ перехода
1		вызов метода ChangePrivateValue с параметром value_1	2

№	Предикат	Действия	№ перехода
2		присваивание значению public_value значение переменной value_2	∅

3.3 Алгоритм метода ChangePrivateValue класса cl_basic

Функционал: изменение значения приватного поля.

Параметры: int, value, значение передаваемое закрытому полю.

Возвращаемое значение: нет.

Алгоритм метода представлен в таблице 4.

Таблица 4 – Алгоритм метода ChangePrivateValue класса cl_basic

№	Предикат	Действия	№ перехода
1		закрытому полю текущего объекта присваиваеться значение равное удвоенному значению переменной value	∅

3.4 Алгоритм конструктора класса cl_basic

Функционал: конструктор класса.

Параметры: int, value_1, значение передаваемое закрытому полю; int, value_2, значение передаваемое открытому полю.

Алгоритм конструктора представлен в таблице 5.

Таблица 5 – Алгоритм конструктора класса cl_basic

№	Предикат	Действия	№ перехода
1		вызов метода ChangePrivateValue с параметром value_1	2
2		присваивание значению public_value значение переменной value_2	∅

3.5 Алгоритм метода ShowValues класса cl_heritage

Функционал: выод значения полей.

Параметры: нет.

Возвращаемое значение: нет.

Алгоритм метода представлен в таблице 6.

Таблица 6 – Алгоритм метода ShowValues класса cl_heritage

№	Предикат	Действия	№ перехода
1		вывод значения закрытого поля, вывод значения открытого поля, переход на новую строчку	∅

3.6 Алгоритм метода ChangeValues класса cl_heritage

Функционал: изменение значения полей.

Параметры: int, value_1, значение передаваемое закрытому полю; int, value_2, значение передаваемое открытому полю.

Возвращаемое значение: нет.

Алгоритм метода представлен в таблице 7.

Таблица 7 – Алгоритм метода ChangeValues класса cl_heritage

№	Предикат	Действия	№ перехода
1		значение закрытого поля приравниваем к значению value_1	2
2		значение открытого поля приравниваем к значению value_2	∅

3.7 Алгоритм конструктора класса cl_heritage

Функционал: конструктор.

Параметры: int, value_1, значение передаваемое закрытому полю; int, value_2, значение передаваемое открытому полю.

Алгоритм конструктора представлен в таблице 8.

Таблица 8 – Алгоритм конструктора класса cl_heritage

№	Предикат	Действия	№ перехода
1		значение закрытого поля приравниваем к значению value_1	2
2		значение открытого поля приравниваем к значению value_2	∅

3.8 Алгоритм функции main

Функционал: выполнение поставленной задачи.

Параметры: нет.

Возвращаемое значение: int - код успешности выполнения программы.

Алгоритм функции представлен в таблице 9.

Таблица 9 – Алгоритм функции main

№	Предикат	Действия	№ перехода
1		объявление целочисленной переменной PrivateValue и PublicValue	2
2		ввод значений для PrivateValue, PublicValue	3
3		инициализация объекта object класса cl_heritage с парасетрами PrivateValue, PublicValue	4
4		вызов метода ShowValues для объекта object	5
5		вызов метода ShowValues для объекта object родительского класса	6
6	PrivateValue > 0	вызов метода ChangeValues для object с параметрами PrivateValue+1, PublicValue+1	7
		вызов метода ChangeValues для object родительского класса с параметрами PrivateValue+1, PublicValue+1	10
7		вызов метода CHangeValues для object родительского класса с параметрами PrivateValue-1, PublicValue-1	8

№	Предикат	Действия	№ перехода
8		вызов метода ShowValues для object	9
9		вызов метода ShowValues для object родительского класса	13
10		вызов метода ChangeValues для object с параметрами PrivateValue-1, PublicValue-1	11
11		вызов метода ShowValues для object родительского класса	12
12		вызов метода ShowValues для object	13
13		возврат 0	∅

4 БЛОК-СХЕМЫ АЛГОРИТМОВ

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

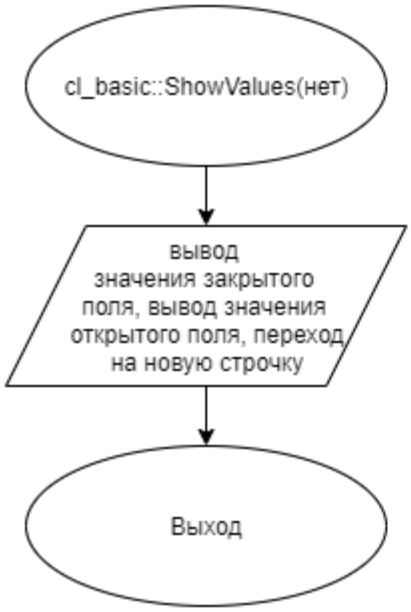

Рисунок 1 – Блок-схема алгоритма

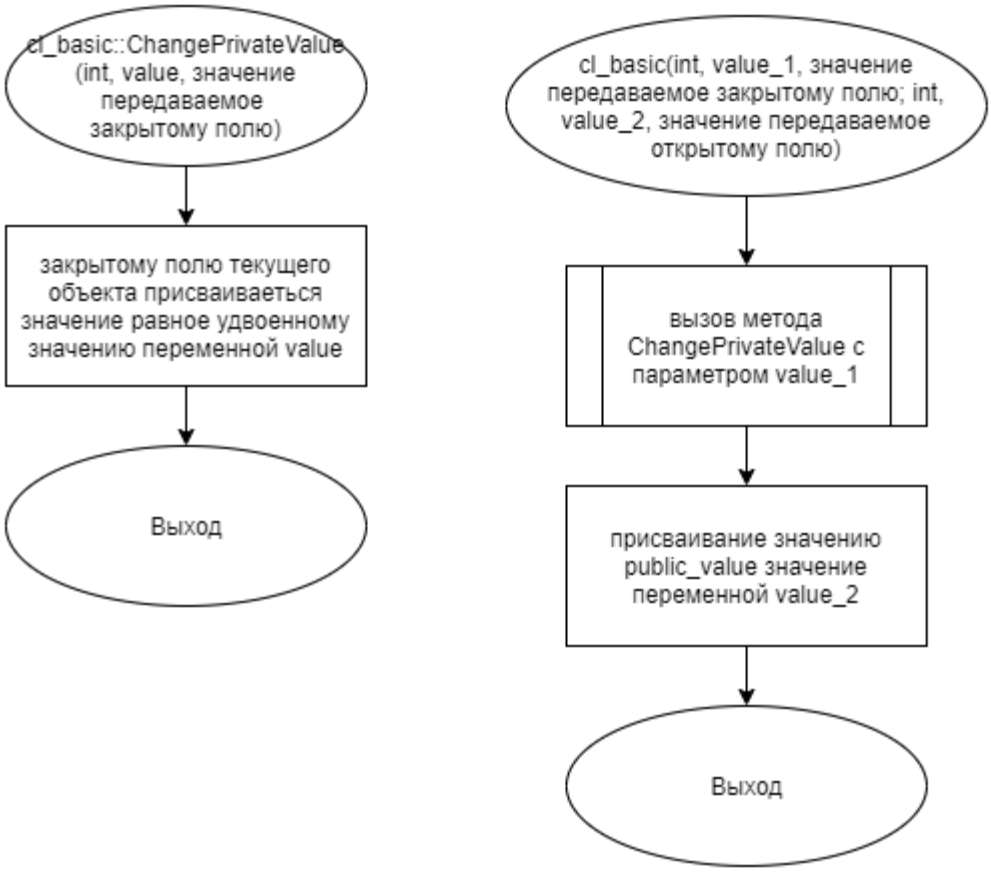

Рисунок 2 – Блок-схема алгоритма

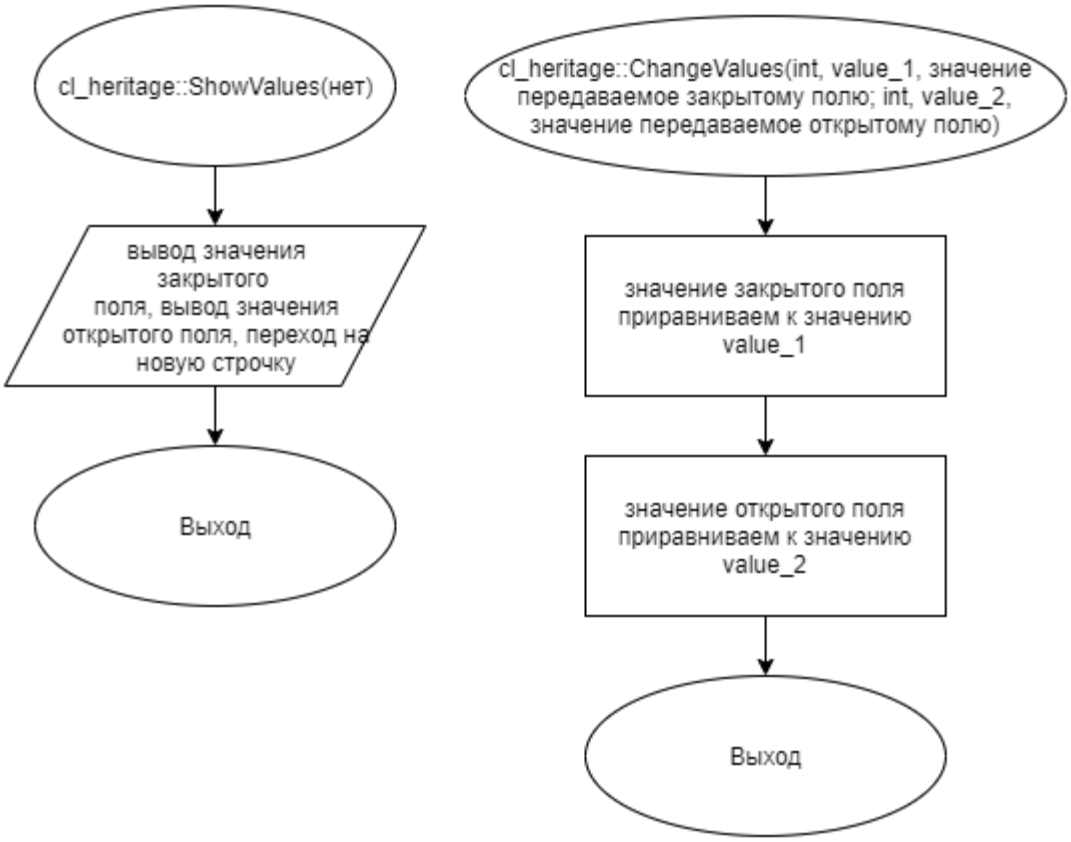

Рисунок 3 – Блок-схема алгоритма

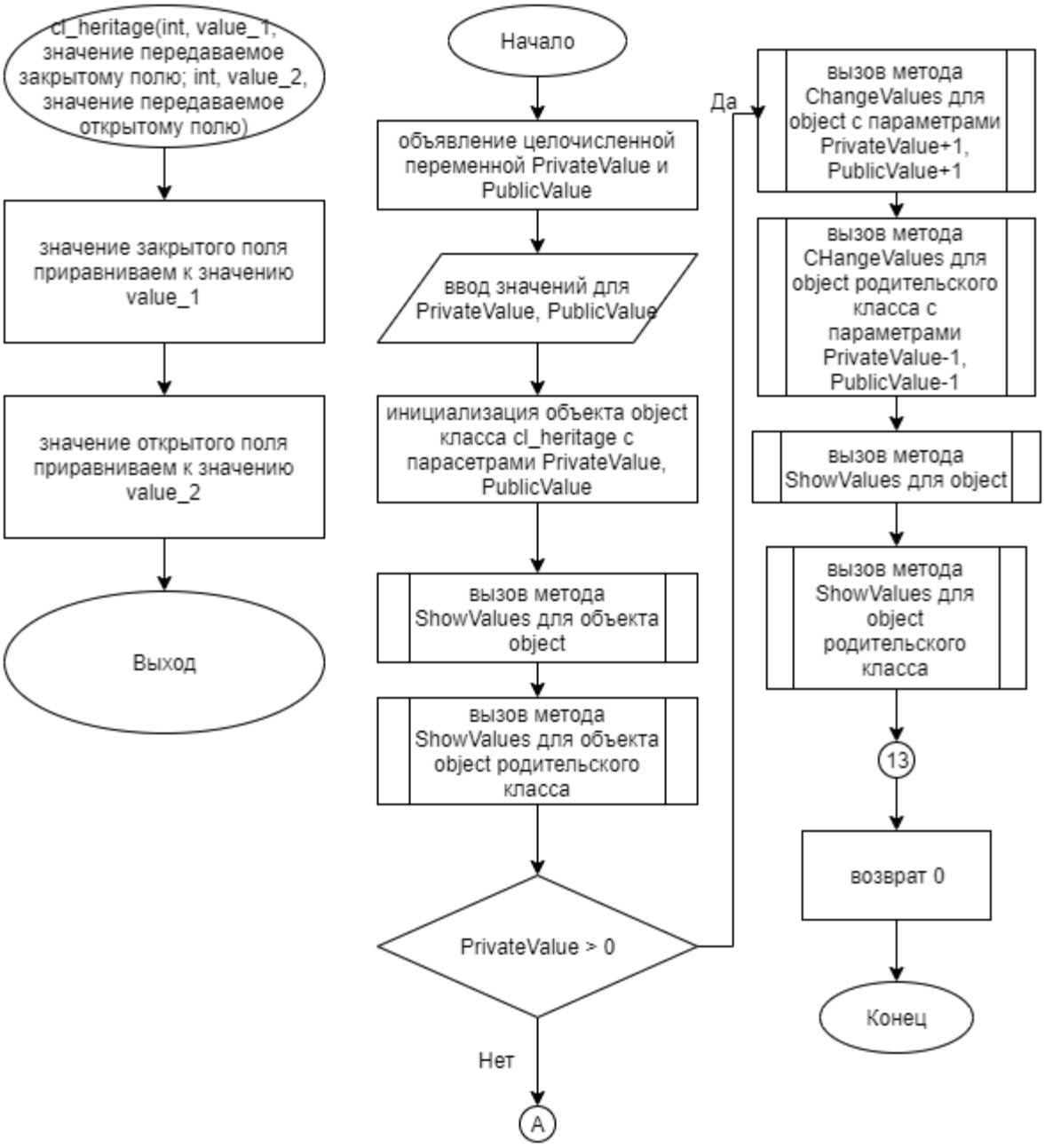

Рисунок 4 – Блок-схема алгоритма

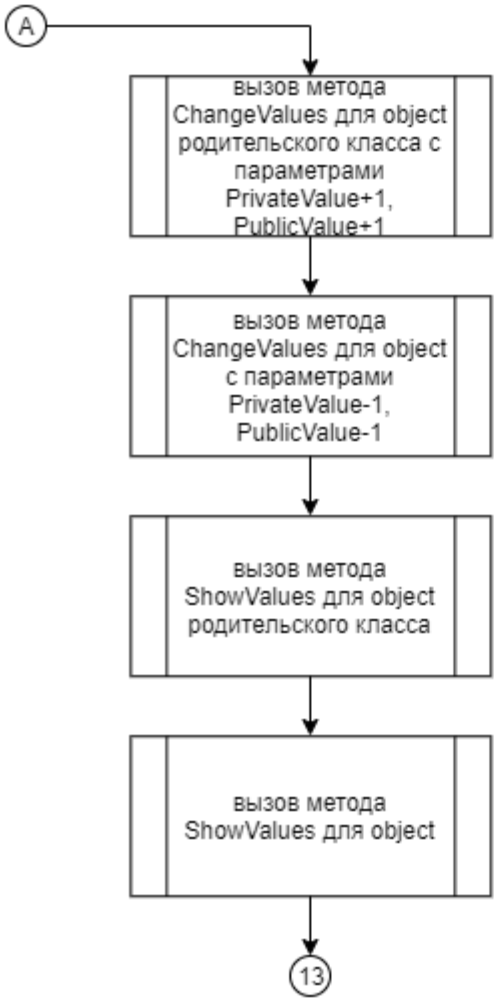

Рисунок 5 – Блок-схема алгоритма

5 КОД ПРОГРАММЫ

Программная реализация алгоритмов для решения задачи представлена ниже.

5.1 Файл cl_basic.cpp

Листинг 1 – cl_basic.cpp

```
#include "cl_basic.h"

cl_basic::cl_basic(int value_1, int value_2){
   ChangePrivateValue(value_1);
   public_value = value_2;
}
void cl_basic::ShowValues(){
   std::cout << private_value << "    " << public_value << "\n";
}
void cl_basic::ChangeValues(int value_1,int value_2){
   ChangePrivateValue(value_1);
   public_value = value_2;
}
void cl_basic::ChangePrivateValue(int value){
   this->private_value = value * 2;
}
```

5.2 Файл cl_basic.h

Листинг 2 – cl_basic.h

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


class cl_basic{
public:
   cl_basic(int value_1, int value_2);

   void ShowValues();
   void ChangeValues(int value_1, int value_2);

   int public_value;
```

```
private:
   void ChangePrivateValue(int value);

   int private_value;
};



#endif
```

5.3 Файл cl_heritage.cpp

Листинг 3 – cl_heritage.cpp

```
#include "cl_heritage.h"

cl_heritage::cl_heritage(int  value_1,  int  value_2  )  :  cl_basic(value_1,
value_2){
   private_value = value_1;
   public_value = value_2;
}
void cl_heritage::ChangeValues(int value_1, int value_2){
   private_value = value_1;
   public_value = value_2;
}
void cl_heritage::ShowValues(){
   std::cout << private_value << "    " << public_value << "\n";
}
```

5.4 Файл cl_heritage.h

Листинг 4 – cl_heritage.h

```
#ifndef __CL_HERITAGE__H
#define __CL_HERITAGE__H
#include "cl_basic.h"

class cl_heritage : public cl_basic{
public:
   cl_heritage(int value_1, int value_2);

   void ShowValues();
   void ChangeValues(int value_1, int value_2);
```

```
   int public_value;
private:
   int private_value;
};

#endif
```

5.5 Файл main.cpp

Листинг 5 – main.cpp

```
#include <stdlib.h>
#include <stdio.h>
#include "cl_heritage.h"

int main()
{
   int PrivateValue, PublicValue;
   std::cin >> PrivateValue >> PublicValue;
   cl_heritage object(PrivateValue, PublicValue);
   object.cl_basic::ShowValues();
   object.ShowValues();
   if (PrivateValue > 0){
      object.ChangeValues(PrivateValue + 1, PublicValue + 1);
      object.cl_basic::ChangeValues(PrivateValue - 1, PublicValue - 1);
      object.ShowValues();
      object.cl_basic::ShowValues();
   }
   else{
      object.ChangeValues(PrivateValue - 1, PublicValue - 1);
      object.cl_basic::ChangeValues(PrivateValue + 1, PublicValue + 1);
      object.cl_basic::ShowValues();
      object.ShowValues();
   }
   return 0;
}
```

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

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).