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 a класса cl_derivative предназначен для работа с родительским и производным классами;
- cout - стандартный объект потока вывода данных;
- cin - стандартный объект потока ввода данных;
- if - условный оператор.

Класс cl_parent:

- свойства/поля:
  - поле Закрытое свойство:
    - наименование — x_closed;
    - тип — int;
    - модификатор доступа — private;
  - поле Открытое свойство:
    - наименование — x_open;
    - тип — int;
    - модификатор доступа — public;
- функционал:
  - метод cl_parent — конструктор;
  - метод closed — изменение значений полей;
  - метод open — присвоение значений полям;
  - метод out — вывод значений полей;
  - метод ~cl_parent — деструктор.

Класс cl_derivative:

- свойства/поля:
  - поле Закрытое свойство:

        - наименование — x_closed;
        - тип — int;
        - модификатор доступа — private;
    - поле Открытое свойство:
        - наименование — x_open;
        - тип — int;
        - модификатор доступа — public;
- функционал:
    - метод cl_deriavative — конструктор;
    - метод open — присвоение значений полям;
    - метод out — вывод значений полей.

Таблица 1 – Иерархия наследования классов

№	Имя класса	Классы-наследники	Модификатор доступа при наследовании	Описание	Номер
1	cl_parent			Родительский класс	
		cl_derivative	virtual public		2
2	cl_derivative			Производный класс	

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 конструктора класса cl_parent

Функционал: конструктор.

Параметры: int y, z - присвоение значений полям.

Алгоритм конструктора представлен в таблице 2.

Таблица 2 – Алгоритм конструктора класса cl_parent

№	Предикат	Действия	№ перехода
1		вызов метода closed с параметром y	2
2		присвоение x_closed значения z	∅

3.2 Алгоритм метода closed класса cl_parent

Функционал: изменение значений полей.

Параметры: int y, - изменене закрытого поля.

Возвращаемое значение: void - не возвращает значений.

Алгоритм метода представлен в таблице 3.

Таблица 3 – Алгоритм метода closed класса cl_parent

№	Предикат	Действия	№ перехода
1		присвоить x_closed значение y*2	∅

3.3 Алгоритм метода open класса cl_parent

Функционал: присвоение значений полям.

Параметры: int y, z - присвоение значений полям.

Возвращаемое значение: void - не возвращает значений.

Алгоритм метода представлен в таблице 4.

Таблица 4 – Алгоритм метода open класса cl_parent

№	Предикат	Действия	№ перехода
1		присвоение x_closed значения y	2
2		присвоение x_open значения z	∅

3.4 Алгоритм метода out класса cl_parent

Функционал: вывод значений полей.

Параметры: нет.

Возвращаемое значение: void - не возвращает значений.

Алгоритм метода представлен в таблице 5.

Таблица 5 – Алгоритм метода out класса cl_parent

№	Предикат	Действия	№ перехода
1		вывод с новой строки x_closed " " x_open	∅

3.5 Алгоритм деструктора класса cl_parent

Функционал: деструктор.

Параметры: нет.

Алгоритм деструктора представлен в таблице 6.

Таблица 6 – Алгоритм деструктора класса cl_parent

№	Предикат	Действия	№ перехода
1		уничтожение объекта	∅

3.6 Алгоритм метода open класса cl_derivative

Функционал: присвоение значений полям.

Параметры: int y, z - присвоение значений полям.

Возвращаемое значение: void - не возвращает значений.

Алгоритм метода представлен в таблице 7.

Таблица 7 – Алгоритм метода open класса cl_derivative

№	Предикат	Действия	№ перехода
1		присвоение x_closed значения y	2
2		присвоение x_open значения z	∅

3.7 Алгоритм метода out класса cl_derivative

Функционал: вывод значений полей.

Параметры: нет.

Возвращаемое значение: void - не возвращает значений.

Алгоритм метода представлен в таблице 8.

Таблица 8 – Алгоритм метода out класса cl_derivative

№	Предикат	Действия	№ перехода
1		вывод с новой строки x_closed " " x_open	∅

3.8 Алгоритм функции main

Функционал: основной алгоритм программы.

Параметры: нет.

Возвращаемое значение: целочисленное - индикатор корректности завершения алгоритма.

Алгоритм функции представлен в таблице 9.

Таблица 9 – Алгоритм функции main

№	Предикат	Действия	№ перехода
1		объявление целочисленных переменных y и z	2
2		ввод y с клавиатуры	3
3		ввод z с клавиатуры	4
4		создание производного объекта a класса cl_derivative с параметрами y и z	5
5		вызов родительского метода out для объекта a	6
6		вызов производного метода out для объекта a	7
7	y>0	вызов производного метода open для объекта a с параметрами y+1, z+1	8
		вызов родительского метода open для объекта a с параметрами y+1, z+1	11
8		вызов родительского метода open для объекта a с параметрами y-1, z-1	9
9		вызов производного метода out для объекта a	10
10		вызов родительского метода out для объекта a	∅
11		вызов производного метода open для объекта a с параметрами y-1, z-1	12
12		вызов родительского метода out для объекта a	13
13		вызов производного метода out для объекта a	∅

3.9 Алгоритм метода cl_deriavative класса cl_derivative

Функционал: конструктор.

Параметры: int y, z - присвоение значений полям.

Возвращаемое значение: конструктор, не возвращает значений.

Алгоритм метода представлен в таблице 10.

Таблица 10 – Алгоритм метода cl_deriavative класса cl_derivative

№	Предикат	Действия	№ перехода
1		присвоение x_closed значения y	2
2		присвоение x_open значения z	∅

4 БЛОК-СХЕМЫ АЛГОРИТМОВ

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

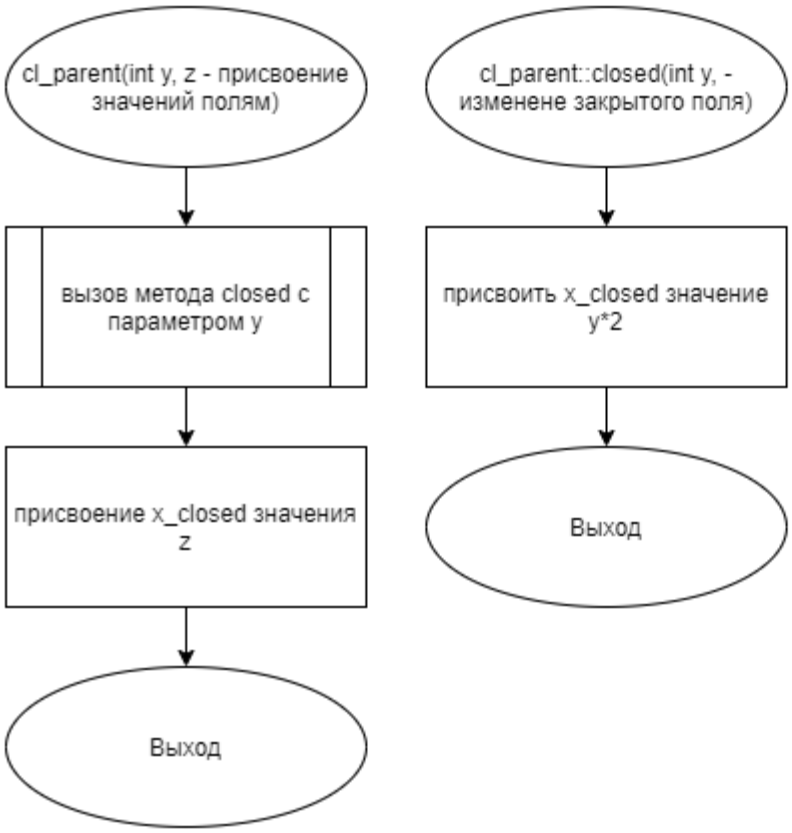

Рисунок 1 – Блок-схема алгоритма

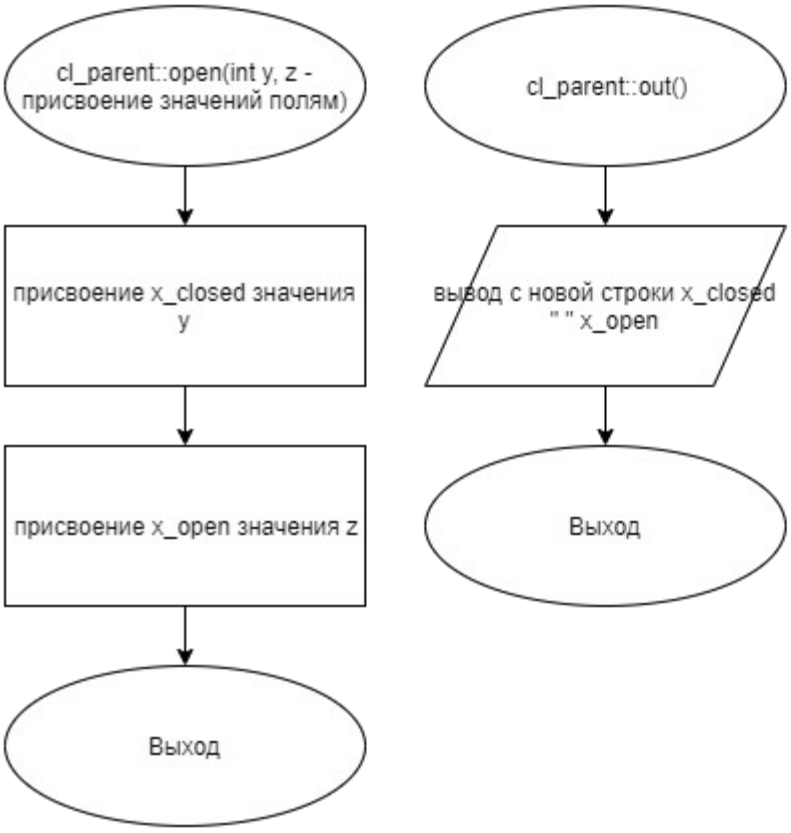

Рисунок 2 – Блок-схема алгоритма

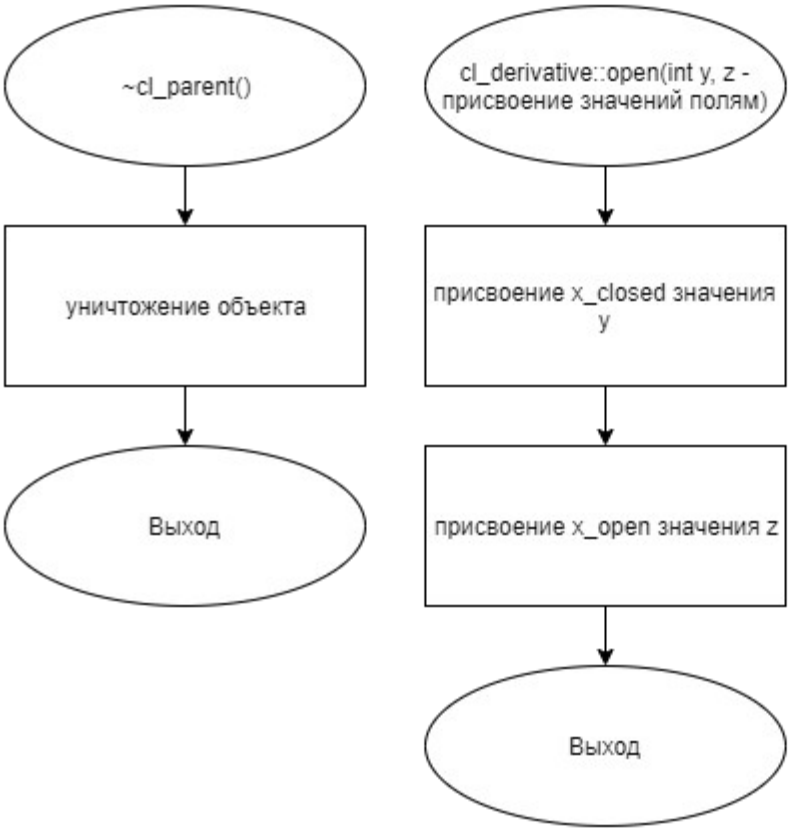

Рисунок 3 – Блок-схема алгоритма

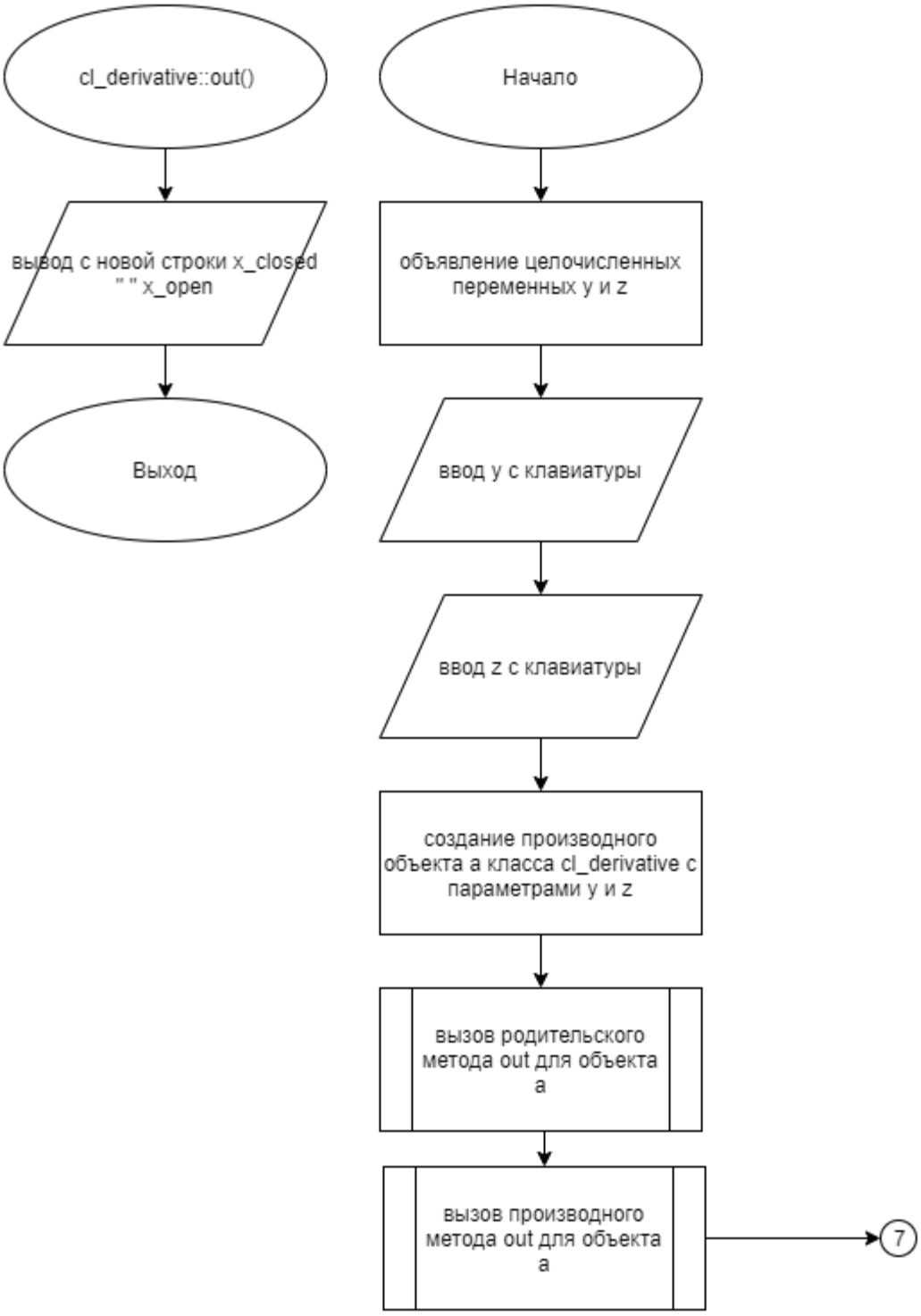

Рисунок 4 – Блок-схема алгоритма

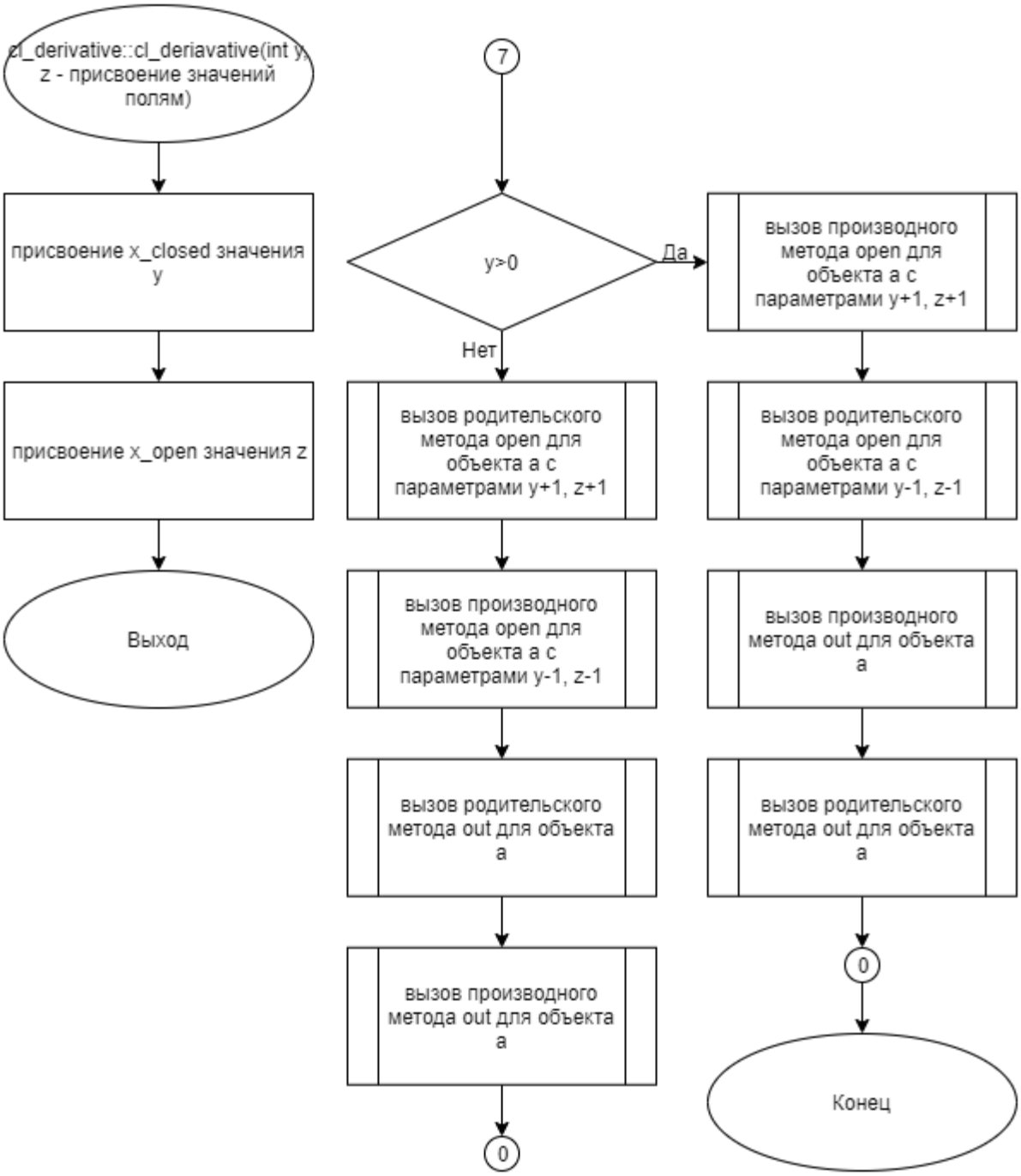

Рисунок 5 – Блок-схема алгоритма

5 КОД ПРОГРАММЫ

Программная реализация алгоритмов для решения задачи представлена ниже.

5.1 Файл cl_derivative.cpp

Листинг 1 – cl_derivative.cpp

```
#include "cl_derivative.h"
#include "cl_parent.h"
#include <iostream>
using namespace std;
cl_derivative::cl_derivative(int y, int z):cl_parent(y, z){
   x_closed=y;
   x_open = z;
}
void cl_derivative::open(int y, int z){
   x_closed = y;
   x_open = z;
}
void cl_derivative::out(){
   cout<<x_closed<<"    "<<x_open<<endl;
}
cl_derivative::~cl_derivative(){}
```

5.2 Файл cl_derivative.h

Листинг 2 – cl_derivative.h

```
#ifndef __CL_DERIVATIVE__H
#define __CL_DERIVATIVE__H
#include "cl_parent.h"
class cl_derivative:public cl_parent{
   private:
   int x_closed;
   public:
   int x_open;
   cl_derivative(int y, int z);
   void open(int y, int z);
   void out();
   ~cl_derivative();
};
```

```
#endif
```

5.3 Файл cl_parent.cpp

Листинг 3 – cl_parent.cpp

```
#include "cl_parent.h"
#include <iostream>
using namespace std;
void cl_parent::closed(int y){
   x_closed = y*2;
}
cl_parent::cl_parent(int y, int z){
   closed(y);
   x_open = z;
}
void cl_parent::open(int y, int z){
   closed(y);
   x_open = z;
}
void cl_parent::out(){
   cout<<x_closed<<"    "<<x_open<<endl;
}
cl_parent::~cl_parent(){}
```

5.4 Файл cl_parent.h

Листинг 4 – cl_parent.h

```
#ifndef __CL_PARENT__H
#define __CL_PARENT__H
class cl_parent{
   private:
   int x_closed;
   void closed(int y);
   public:
   int x_open;
   cl_parent(int y, int z);
   void open(int y, int z);
   void out();
   ~cl_parent();

};
#endif
```

5.5 Файл main.cpp

Листинг 5 – main.cpp

```
#include <stdlib.h>
#include <stdio.h>
#include <iostream>
#include "cl_parent.h"
#include "cl_derivative.h"
using namespace std;
int main()
{
   int y,z;
   cin>>y;
   cin>>z;
   cl_derivative a(y,z);
   a.cl_parent::out();
   a.out();
   if(y>0){
      a.open(y+1,z+1);
      a.cl_parent::open(y-1,z-1);
      a.out();
      a.cl_parent::out();
   }
   else{
      a.cl_parent::open(y+1,z+1);
      a.open(y-1,z-1);
      a.cl_parent::out();
      a.out();
   }
   return(0);
}
```

6 ТЕСТИРОВАНИЕ

Результат тестирования программы представлен в таблице 11.

Таблица 11 – 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).