Здесь будет титульник, листай ниже

СОДЕРЖАНИЕ

1 ПОСТАНОВКА ЗАДАЧИ

Разработать систему, которая демонстрирует возможность использования объекта дружественного класса.

Система состоит из множества однотипных объектов. Объекты содержат характеристику, на которую влияет внешнее возмущение, что приводит к отклонению части значений за пределы допустимой нормы. В составе системы входит инструмент, который периодически анализирует состояние каждого объекта и при обнаружении недопустимых отклонений проводит необходимые корректировки, для восстановления режима нормального функционирования объектов.

Спроектировать объект, с свойствами в закрытом доступе:

- целого типа, для хранения размерности массива;
- целого типа, для хранения величины допустимого отклонения;
- указатель на объект целого типа.

С параметризированным конструктором. У конструктора есть параметр целого типа. Параметр передает (содержит) значение размерности целочисленного массива. У конструктора есть параметр целого типа. Параметр передает (содержит) значение величины допустимого отклонения. В конструкторе: фиксируются значения размерности массива и величины допустимого отклонения; создается целочисленный массив заданной размерности; вводится значения элементов массива. Класс данного объекта является родительским. Имеется функционал (методы) в открытом доступе:

- реализует возмущение значений элементов целочисленного массива. Метод имеет одни целочисленный параметр, который содержит величину возмущения для элементов массива. Каждому элементу массива добавляется величина возмущения;

- реализует вывод значений элементов массива.

В деструкторе освобождается память, выделенная для массива.

Спроектировать производный объект, на базе родительского объекта. У объекта в закрытом доступе имеется свойство:

- строкового типа, для хранения наименования объекта.

С параметризированным конструктором. У конструктора есть параметр целого типа. Параметр передает (содержит) значение размерности целочисленного массива. У конструктора есть параметр целого типа. Параметр передает (содержит) значение величины допустимого отклонения. В реализации конструктора вводится значение наименования объекта.

Имеется функционал (метод) в открытом доступе, который с новой строки выводит наименование объекта и значения элементов массива.

Спроектировать объект, который анализирует состояние производных объектов. Проверяет значения всех элементов массива. Если значение выходит за рамки интервала допустимости, то значение элемента меняет на ноль. Для этого у объекта имеется соответствующий метод. Допустимым интервалом является отрицательное и положительного значение величины отклонения. Например: [-5, 5].

Алгоритм конструирования и отработки системы:

1. Объявляются целочисленные переменные для хранения значений: количества объектов; размерности массива; величины допустимого отклонения; количество итераций.

2. Объявляется строковая переменная, для хранения наименования объекта.

3. Могут быть другие объявления.

4. Вводятся значения: количество объектов; размерность массива; величины допустимого отклонения; количество итераций.

5. В цикле создаются производные объекты, согласно введенному

количеству.

6. Запускается цикл согласно количеству итераций.

6.1. С новой строки выводится: Iteration = «номер итерации»

6.2. С новой строки выводится: The original

6.3. Для каждого объекта выводиться значения элементов массива.

6.4. Вводится значение величины возмущения.

6.5. Для каждого объекта отрабатывает метод возмущения.

6.6. С новой строки выводится: After the outrage

6.7. Посредством объекта анализа и корректировки состояния производного объекта, проверяется и приводиться в нормальный режим работы каждый производный объект.

6.8. С новой строки выводится: The result

6.9. Для каждого объекта выводиться значения элементов массива.

7. После завершения цикла итераций, созданные производные объекты удаляются (уничтожаются).

1.1 Описание входных данных

Первая строка:

`«целое число, количество объектов»`

Вторая строка:

`«целое число, размерность массива»`

Третья строка:

`«целое число, величина допустимого отклонения»`

Четвертая строка:

`«целое число, количество итераций»`

Начиная с пятой строки, значения элементов массивов и имя очередного

объекта, согласно количеству объектов:

```
«целое число» «целое число» . . . «целое число» «строка»
```

Далее, построчно значения величины возмущения, согласно количества итерации.

```
«целое число, значение величины возмущения»
```

Пример ввода

```
2
5
6
4
1 -1 1 -1 1 object_1
-2 2 -2 2 -2 object_2
3
4
2
-7
```

1.2 Описание выходных данных

В процессе каждой итерации производится вывод.

```
Iteration = «номер итерации»
The original
«имя объекта»  «целое число»  «целое число» . . . «целое число»
```

Наименование объекта и значения элементов массива, согласно последовательности создания объектов.

```
After the outrage
«имя объекта»  «целое число»  «целое число» . . . «целое число»
```

Наименование объекта и значения элементов массива, согласно последовательности создания объектов.

```
The result
```

«имя объекта» «целое число» «целое число» . . . «целое число»

Наименование объекта и значения элементов массива, согласно последовательности создания объектов.

Пример вывода

```
Iteration = 1
The original
object_1  1  -1  1  -1  1
object_2  -2  2  -2  2  -2
After the outrage
object_1  4  2  4  2  4
object_2  1  5  1  5  1
The result
object_1  4  2  4  2  4
object_2  1  5  1  5  1
Iteration = 2
The original
object_1  4  2  4  2  4
object_2  1  5  1  5  1
After the outrage
object_1  8  6  8  6  8
object_2  5  9  5  9  5
The result
object_1  0  6  0  6  0
object_2  5  0  5  0  5
Iteration = 3
The original
object_1  0  6  0  6  0
object_2  5  0  5  0  5
After the outrage
object_1  2  8  2  8  2
object_2  7  2  7  2  7
The result
object_1  2  0  2  0  2
object_2  0  2  0  2  0
Iteration = 4
The original
object_1  2  0  2  0  2
object_2  0  2  0  2  0
After the outrage
object_1  -5  -7  -5  -7  -5
object_2  -7  -5  -7  -5  -7
The result
object_1  -5  0  -5  0  -5
object_2  0  -5  0  -5  0
```

2 МЕТОД РЕШЕНИЯ

Для решения задачи используется:

- объект analyzer класса Analyzer предназначен для анализа производных объектов;
- функция main для определения входной точки программы;
- объекты без наименования класса ChildObj для работы предназначены для построения системы;
- стандартная библиотека ввода-вывода;
- указатели;
- заголовочные файлы;
- динамический массив.

Класс ParentObj:

- свойства/поля:
    - поле хранит размерность массива:
        - наименование — m_size;
        - тип — int;
        - модификатор доступа — private;
    - поле хранит величину допустимого отклонения:
        - наименование — m_permittedDeviation;
        - тип — int;
        - модификатор доступа — private;
    - поле хранит массив:
        - наименование — m_array;
        - тип — int*;
        - модификатор доступа — private;
- функционал:

- метод ParentObj — параметризированный конструктор;
- метод ~ParentObj — деструктор;
- метод Perturbation — добавляет величину возмущения к каждому элементу массива;
- метод PrintArray — выводит элементы массива.

Класс ChildObj:

- свойства/поля:
  - поле хранит наименование объекта:
    - наименование — m_name;
    - тип — std::string;
    - модификатор доступа — private;
- функционал:
  - метод ChildObj — параметризированный конструктор;
  - метод PrintArray — выводит наименование объекта и значения элементов массива.

Класс Analyzer:

- функционал:
  - метод Sanitize — анализирует состояние производных объектов.

Таблица 1 – Иерархия наследования классов

№	Имя класса	Классы-наследники	Модификатор доступа при наследовании	Описание	Номер
1	ParentObj				
		ChildObj	public		2
2	ChildObj				
3	Analyzer				

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 ParentObj

Функционал: Параметризированный конструктор.

Параметры: int size, int deviation.

Алгоритм конструктора представлен в таблице 2.

Таблица 2 – Алгоритм конструктора класса ParentObj

№	Предикат	Действия	№ перехода
1		Выделение памяти под динамический массив m_array размерностью m_size с помощью оператора new	2
2		Инициализация целочисленной переменной i = 0	3
3	i < m_size	Ввод m_array[i]	3
			∅

3.2 Алгоритм деструктора класса ParentObj

Функционал: Деструктор.

Параметры: нет.

Алгоритм деструктора представлен в таблице 3.

Таблица 3 – Алгоритм деструктора класса ParentObj

№	Предикат	Действия	№ перехода
1		Освобождение памяти, выделенной под целочисленной массив	∅

№	Предикат	Действия	№ перехода
		m_array с помощью оператора delete	

3.3 Алгоритм метода Perturbation класса ParentObj

Функционал: Добавляет величину возмущения к каждому элементу массива.

Параметры: int perturbation.

Возвращаемое значение: void.

Алгоритм метода представлен в таблице 4.

Таблица 4 – Алгоритм метода Perturbation класса ParentObj

№	Предикат	Действия	№ перехода
1		Инициализация целочисленной переменной i = 0	2
2	i < m_size	Добавление значения отклонения к m_array[i]	2
			∅

3.4 Алгоритм метода PrintArray класса ParentObj

Функционал: Выводит элементы массива.

Параметры: нет.

Возвращаемое значение: void.

Алгоритм метода представлен в таблице 5.

Таблица 5 – Алгоритм метода PrintArray класса ParentObj

№	Предикат	Действия	№ перехода
1		Инициализация целочисленной переменной i = 0	2
2	i < m_size	Вывод m_array[i]	2
			∅

3.5 Алгоритм конструктора класса ChildObj

Функционал: Параметризированный конструктор.

Параметры: int size, int deviation.

Алгоритм конструктора представлен в таблице 6.

Таблица 6 – Алгоритм конструктора класса ChildObj

№	Предикат	Действия	№ перехода
1		Ввод значения наименования объекта m_name	∅

3.6 Алгоритм метода PrintArray класса ChildObj

Функционал: Выводит наименование объекта и значения элементов массива.

Параметры: нет.

Возвращаемое значение: void.

Алгоритм метода представлен в таблице 7.

Таблица 7 – Алгоритм метода PrintArray класса ChildObj

№	Предикат	Действия	№ перехода
1		Вывод наименования объекта m_name	2
2		Инициализация целочисленной переменной i = 0	3
3	i < m_size	Вывод двух пробельных символов и m_array[i]	3
			4
4		Вывод символа переноса строки	∅

3.7 Алгоритм метода Sanitize класса Analyzer

Функционал: Анализирует состояние производных объектов.

Параметры: Child* obj.

Возвращаемое значение: void.

Алгоритм метода представлен в таблице 8.

Таблица 8 – Алгоритм метода Sanitize класса Analyzer

№	Предикат	Действия	№ перехода
1		Инициализация целочисленной переменной max значением obj->m_permittedDeviation	2
2		Инициализация целочисленной переменной min значением -max	3
3		Инициализация целочисленной переменной i = 0	4
4	i < obj->m_size		5
			∅
5	obj->m_array[i] < min || obj->m_array[i] > max	Присвоение obj->m_array[i] = 0	4
			4

3.8 Алгоритм функции main

Функционал: Основная функция.

Параметры: нет.

Возвращаемое значение: int.

Алгоритм функции представлен в таблице 9.

Таблица 9 – Алгоритм функции main

№	Предикат	Действия	№ перехода
1		Объявление целочисленных переменных objCount, size, deviation, iterations, perturbation	2
2		Объявление строковой переменной name	3
3		Ввод objCount, size, deviation, iterations	4
4		Создание объекта analyzer класса Analyzer с помощью оператора new с выделением памяти	5

№	Предикат	Действия	№ перехода
		указателя	
5		Выделение памяти под динамический массив objects указателей на объекты класса ChildObj размерностью, равной значению целочисленной переменной size	6
6		Инициализация целочисленной переменной i = 0	7
7	i < objCount	Создание объекта класса ChildObj с помощью оператора new и присваивание его objects[i]	7
			8
8		Инициализация целочисленной переменной i = 1	9
9	i < iterations + 1	Вывод "Iteration = ", значения целочисленной переменной i, символа переноса строки	10
			22
10		Вывод "The original" и символа переноса строки	11
11		Ввод perturbation	12
12		Инициализация целочисленной переменной i = 0	13
13	i < objCount	Вызов objects[i]->PrintArray()	14
			15
14		Вызов objects[i]->Perturbation(perturbation)	13
15		Вывод "After the outrage", символа переноса строки	16
16		Инициализация целочисленной переменной i = 0	17
17	i < objCount	Вызов objects[i]->PrintArray()	18
			19
18		Вызов analyzer->Sanitize(objects[i])	17
19		Вывод "The result", символа переноса строки	20
20		Инициализация целочисленной переменной i = 0	21
21	i < objCount	Вызов objects[i]->PrintArray()	21
			9

№	Предикат	Действия	№ перехода
22		Инициализация целочисленной переменной i = 0	23
23	i < objCount	Удаление объекта objects[i] с помощью оператора delete с освобождением памяти указателя	23
			∅

4 БЛОК-СХЕМЫ АЛГОРИТМОВ

Представим описание алгоритмов в графическом виде на рисунках 1-7.

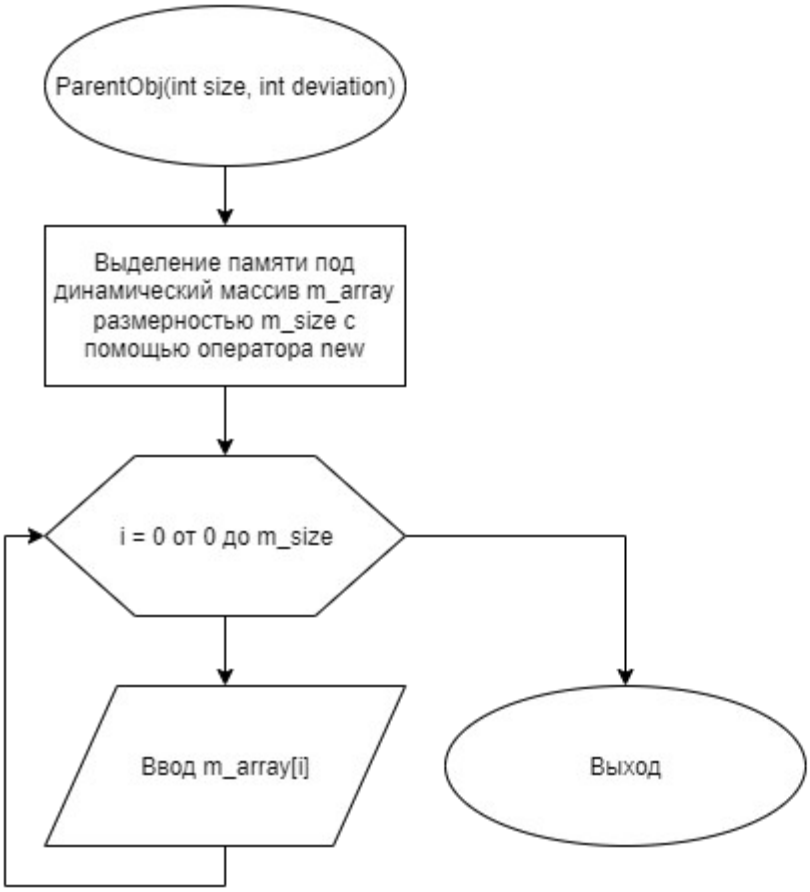

Рисунок 1 – Блок-схема алгоритма

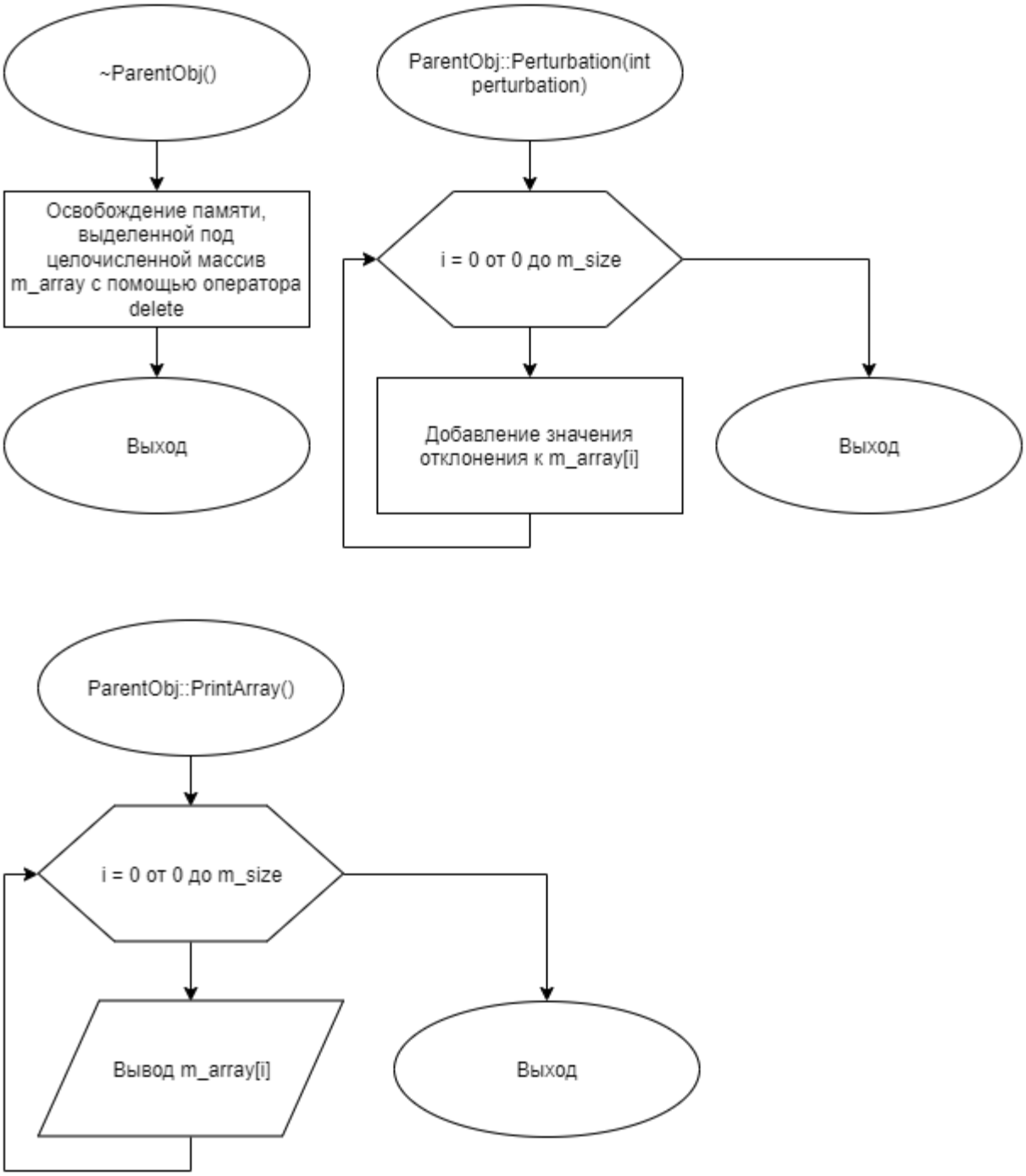

Рисунок 2 – Блок-схема алгоритма

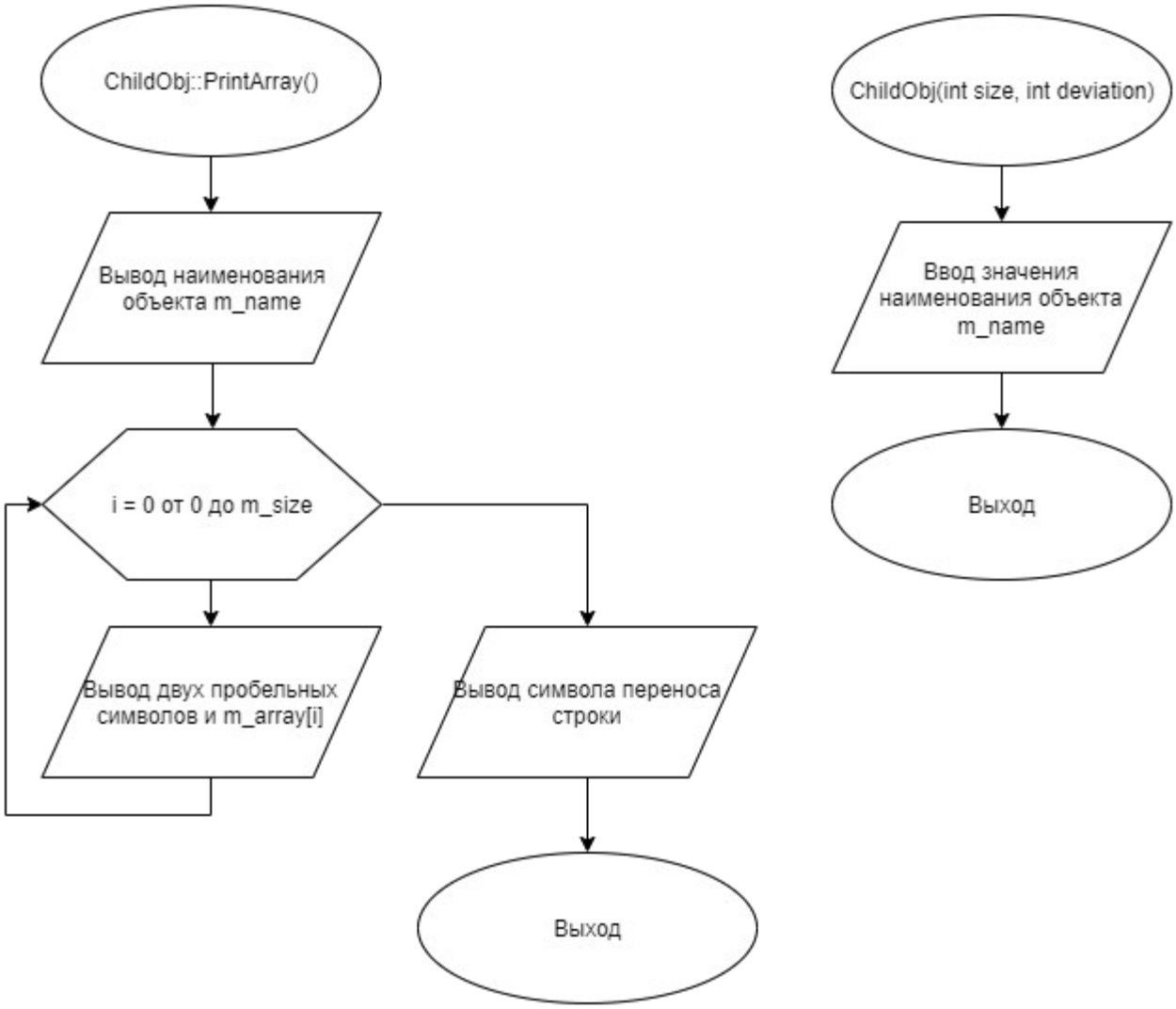

Рисунок 3 – Блок-схема алгоритма

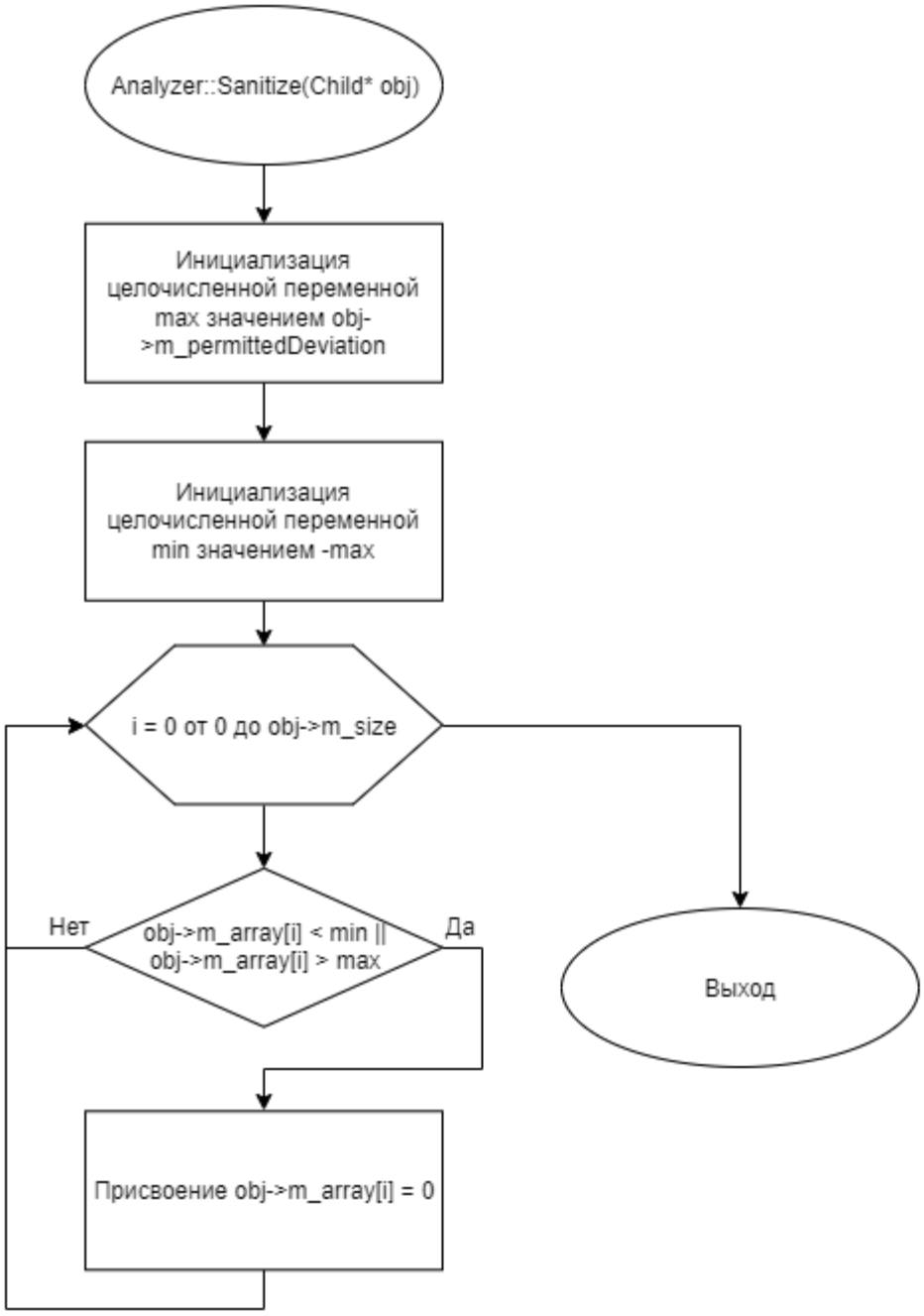

Рисунок 4 – Блок-схема алгоритма

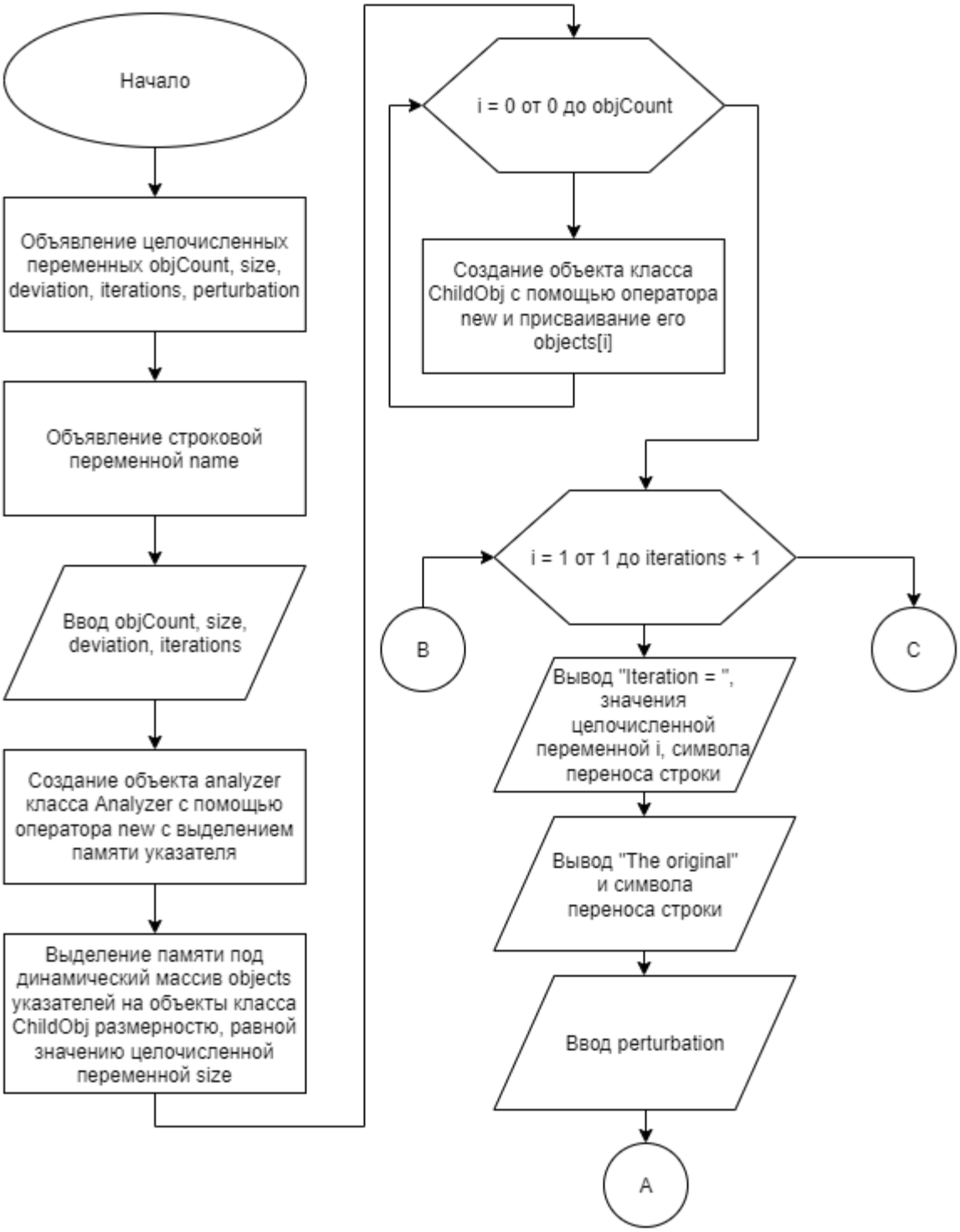

Рисунок 5 – Блок-схема алгоритма

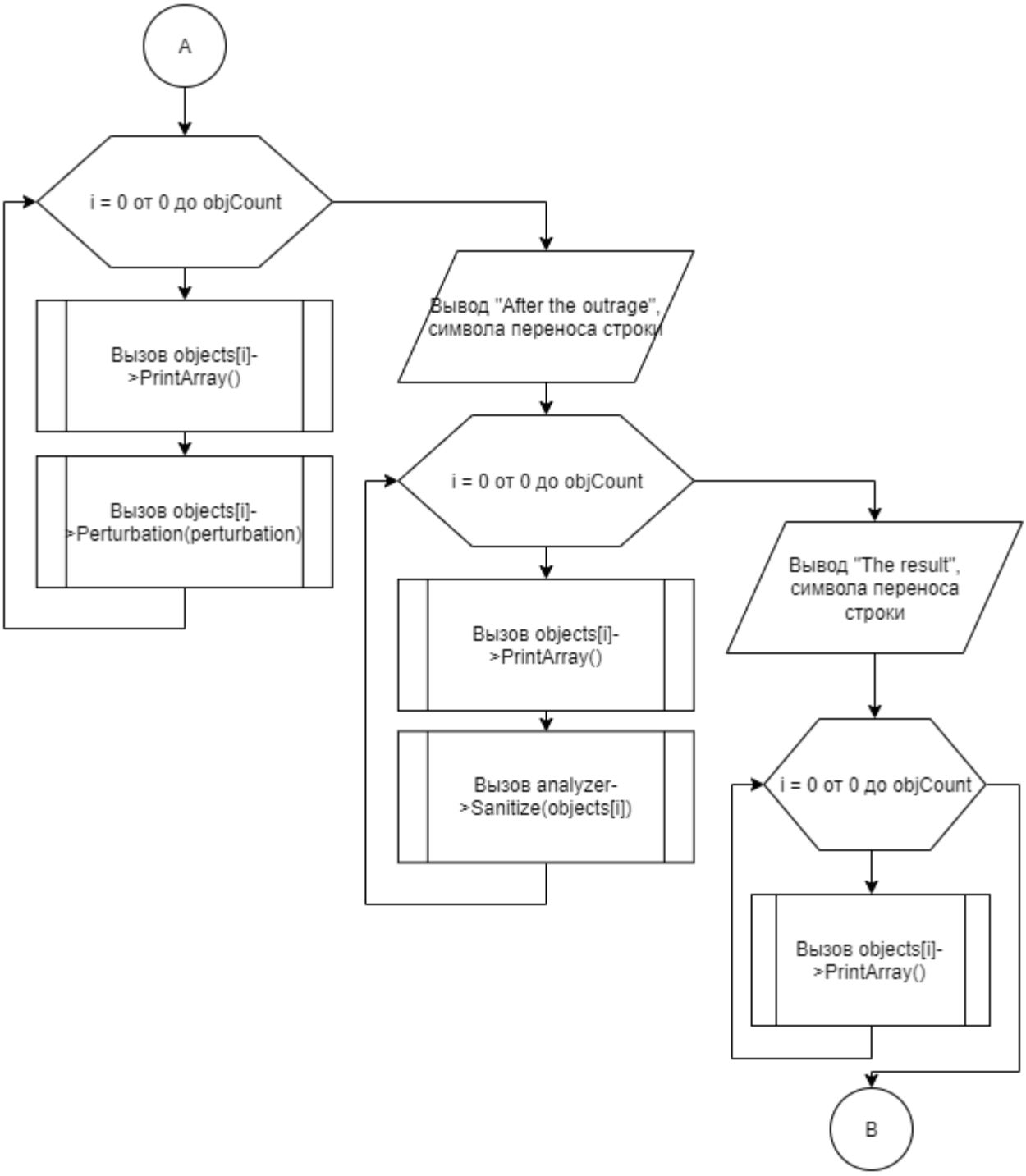

Рисунок 6 – Блок-схема алгоритма

Рисунок 7 – Блок-схема алгоритма

5 КОД ПРОГРАММЫ

Программная реализация алгоритмов для решения задачи представлена ниже.

5.1 Файл Analyzer.cpp

Листинг 1 – Analyzer.cpp

```
#include "Analyzer.h"

void Analyzer::Sanitize(ChildObj* obj)
{
   int max = obj->m_permittedDeviation;
   int min = -max;

   for (int i = 0; i < obj->m_size; i++)
   {
      if (obj->m_array[i] < min || obj->m_array[i] > max)
         obj->m_array[i] = 0;
   }
}
```

5.2 Файл Analyzer.h

Листинг 2 – Analyzer.h

```
#ifndef __ANALYZER__H
#define __ANALYZER__H

#include "ChildObj.h"

class Analyzer
{
public:
   void Sanitize(ChildObj* obj);
};

#endif
```

5.3 Файл ChildObj.cpp

Листинг 3 – ChildObj.cpp

```
#include "ChildObj.h"

ChildObj::ChildObj(int size, int deviation) : ParentObj(size, deviation)
{
   std::cin >> m_name;
}

void ChildObj::PrintArray()
{
   std::cout << m_name;

   for (int i = 0; i < m_size; i++)
   {
      std::cout << "  " << m_array[i];
   }
   std::cout << std::endl;
}
```

5.4 Файл ChildObj.h

Листинг 4 – ChildObj.h

```
#ifndef __CHILDOBJ__H
#define __CHILDOBJ__H

#include "ParentObj.h"
#include <string>

class ChildObj : public ParentObj
{
   friend class Analyzer;

   std::string m_name;
public:
   ChildObj(int size, int deviation);

   void PrintArray();
};

#endif
```

5.5 Файл main.cpp

Листинг 5 – main.cpp

```
#include "ParentObj.h"
#include "ChildObj.h"
#include "Analyzer.h"

int main()
{
   int objCount, size, deviation, iterations, perturbation;
   std::string name;

   std::cin >> objCount >> size >> deviation >> iterations;

   Analyzer* analyzer = new Analyzer();
   ChildObj** objects = new ChildObj*[objCount];

   for (int i = 0; i < objCount; i++)
   {
      objects[i] = new ChildObj(size, deviation);
   }

   for (int i = 1; i < iterations + 1; i++)
   {
      std::cout << "Iteration = " << i << std::endl;
      std::cout << "The original" << std::endl;

      std::cin >> perturbation;

      for (int i = 0; i < objCount; i++)
      {
         objects[i]->PrintArray();
         objects[i]->Perturbation(perturbation);
      }

      std::cout << "After the outrage" << std::endl;

      for (int i = 0; i < objCount; i++)
      {
         objects[i]->PrintArray();
         analyzer->Sanitize(objects[i]);
      }

      std::cout << "The result" << std::endl;

      for (int i = 0; i < objCount; i++)
      {
         objects[i]->PrintArray();
      }
   }

   for (int i = 0; i < objCount; i++)
   {
```

```
      delete objects[i];
   }

   return 0;
}
```

5.6 Файл ParentObj.cpp

Листинг 6 – ParentObj.cpp

```
#include "ParentObj.h"

ParentObj::ParentObj(int size, int deviation) : m_size(size),
m_permittedDeviation(deviation)
{
   m_array = new int[m_size];

   for (int i = 0; i < m_size; i++)
   {
      std::cin >> m_array[i];
   }
}

ParentObj::~ParentObj()
{
   delete m_array;
}

void ParentObj::Perturbation(int perturbation)
{
   for (int i = 0; i < m_size; i++)
   {
      m_array[i] += perturbation;
   }
}

void ParentObj::PrintArray()
{
   for (int i = 0; i < m_size; i++)
   {
      std::cout << m_array[i] << std::endl;
   }
}
```

5.7 Файл ParentObj.h

Листинг 7 – ParentObj.h

```
#ifndef __PARENTOBJ__H
#define __PARENTOBJ__H

#include <iostream>

class ParentObj
{
   friend class ChildObj;
   friend class Analyzer;

   int m_size;
   int m_permittedDeviation;
   int* m_array;
public:
   ParentObj(int size, int deviation);
   ~ParentObj();

   void Perturbation(int perturbation);
   void PrintArray();
};

#endif
```

6 ТЕСТИРОВАНИЕ

Результат тестирования программы представлен в таблице 10.

Таблица 10 – Результат тестирования программы

Входные данные	Ожидаемые выходные данные	Фактические выходные данные
2 5 6 4 1 -1 1 -1 1 object_1 -2 2 -2 2 -2 object_2 3 4 2 -7	Iteration = 1 The original object_1 1 -1 1 -1 1 object_2 -2 2 -2 2 -2 After the outrage object_1 4 2 4 2 4 object_2 1 5 1 5 1 The result object_1 4 2 4 2 4 object_2 1 5 1 5 1 Iteration = 2 The original object_1 4 2 4 2 4 object_2 1 5 1 5 1 After the outrage object_1 8 6 8 6 8 object_2 5 9 5 9 5 The result object_1 0 6 0 6 0 object_2 5 0 5 0 5 Iteration = 3 The original object_1 0 6 0 6 0 object_2 5 0 5 0 5 After the outrage object_1 2 8 2 8 2 object_2 7 2 7 2 7 The result object_1 2 0 2 0 2	Iteration = 1 The original object_1 1 -1 1 -1 1 object_2 -2 2 -2 2 -2 After the outrage object_1 4 2 4 2 4 object_2 1 5 1 5 1 The result object_1 4 2 4 2 4 object_2 1 5 1 5 1 Iteration = 2 The original object_1 4 2 4 2 4 object_2 1 5 1 5 1 After the outrage object_1 8 6 8 6 8 object_2 5 9 5 9 5 The result object_1 0 6 0 6 0 object_2 5 0 5 0 5 Iteration = 3 The original object_1 0 6 0 6 0 object_2 5 0 5 0 5 After the outrage object_1 2 8 2 8 2 object_2 7 2 7 2 7 The result object_1 2 0 2 0

Входные данные	Ожидаемые выходные данные	Фактические выходные данные
	object_2 0 2 0 2 0 Iteration = 4 The original object_1 2 0 2 0 2 object_2 0 2 0 2 0 After the outrage object_1 -5 -7 -5 -7 -5 object_2 -7 -5 -7 -5 -7 The result object_1 -5 0 -5 0 -5 object_2 0 -5 0 -5 0	2 object_2 0 2 0 2 0 Iteration = 4 The original object_1 2 0 2 0 2 object_2 0 2 0 2 0 After the outrage object_1 -5 -7 -5 -7 -5 object_2 -7 -5 -7 -5 -7 The result object_1 -5 0 -5 0 -5 object_2 0 -5 0 -5 0

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).