Здесь будет титульник, листай ниже

СОДЕРЖАНИЕ

1 ПОСТАНОВКА ЗАДАЧИ

Разработать систему, которая демонстрирует работу (использование) конструктора копии.

Первоначально, в процессе конструирования, система состоит из одного объекта (исходного) и из одной функции. Далее, ее работа управляется посредством команд. Команды вводятся с стандартного потока данных. По команде создаются копии объектов от исходного или от других копии. Копия так же создается при передаче объекта в функцию по значению.

Спроектировать объект, с свойствами в закрытом доступе:

- строкового типа, для хранения наименования объекта;
- целого типа, для хранения номера копии объекта;
- целого типа, для хранения размерности массива;
- указатель на объект целого типа, для хранения адреса динамически созданного целочисленного массива;
- свойство, для хранения целочисленного значения, который доступен всем копиям (инструкцию static не использовать);
- еще из дополнительных свойств, для реализации требований, заданных в постановке задачи.

С параметризированным конструктором. У конструктора есть параметр строкового типа. Параметр передает (содержит) значение наименования объекта. В реализации метода фиксируется имя объекта. Свойству номера копии присваивается значение нуль. Общедоступному свойству тоже присваивается нуль. Есть фрагмент, в котором вводится значение размерности целочисленного массива, динамический создается целочисленный массив согласно значению размерности и вводятся значения элементов массива.

С конструктором копии. В конструкторе копии из переданному в качестве

параметра объекту в текущем:

- копируется наименование объекта;
- определяется номер новой копии и присваивается соответствующему свойству;
- определяется значение свойства с общим доступом;
- копируется размерность массива:
- динамический создается целочисленный массив согласно значению размерности. Элементам массива присваиваются значения элементов исходного массива увеличенные на значение номера копии текущего объекта;
- выводиться наименование объекта и номер копии.

Имеется функционал (методы) в открытом доступе:

- с одним целочисленным параметром, значением которого редактируется значение свойства с общим доступом;
- возвращает значение номера копии объекта;
- выводит состояние объекта; наименование объекта; номер копии; значение свойства с общим доступом; значения элементов массива;
- деструктор, который сообщает объект с каким именем и с каким номер копии уничтожается, освобождается память, выделенная для целочисленного массива.

В составе системы определена функция с одним параметром. По этому параметру передается объект по значению. В функции для объекта, переданного по параметру, вызывается метод вывода состояния объекта.

Система работает по следующим командам:

```
copy «целое число, номер копии»
```

По данной команде ищется объект с заданным номером копии, создается новый объект посредством конструктора копии, в котором в качестве аргумента

передается найденный объект. Выводиться состояние нового объекта.

```
function «целое число, номер копии»
```

По данной команде ищется объект с заданным номером копии, вызывается функция и в качестве аргумента передается найденный объект.

```
state «целое число, номер копии»
```

По данной команде ищется объект с заданным номером копии и выводится его состояние.

```
shared «целое число, номер копии» «целое число, значение свойства с общим доступом»
```

По данной команде ищется объект с заданным номером копии. Вводиться значение свойства с общим доступом. Для найденного объекта вызывается метод редактирования значения свойства с общим доступом.

```
end
```

По данной команде система завершает работу.

Если не удается найти объект с данным номер копии, то выдается соответствующее сообщение.

Алгоритм конструирования и отработки системы:

1. Объявляются строковые переменные, для хранения наименования объекта и значения команд системы.
2. Объявляются целочисленные переменные для хранения значений: номера копии объекта и значения свойства с общим доступом.
3. Могут быть другие объявления.
4. Вводится значение наименования объекта.
5. Выводится значение наименования объекта.
6. Создается объект посредством параметризированного конструктора.
7. Для созданного объекта вызывается метод вывода состояния объекта.
8. Могут быть другие операторы.

9. Организуется цикл по командам.

1.1 Описание входных данных

Первая строка:

```
«строка, наименование объекта»
```

Вторая строка:

```
«целое число, размерность массива»
```

Третья строка:

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

Значения элементов массива.

Начиная с четвертой строки, построчно, вводятся команды системы.

```
команда «целое число, номер копии» [«целое число, значение свойства с общим доступом»]
```

Пример ввода

```
Document
5
1 2 3 4 5
copy 0
copy 1
shared 8 99
copy 1
shared 2 77
copy 0
function 1
state 5
state 1
end
```

1.2 Описание выходных данных

Каждый вывод производится с новой строки.

Вывод наименования объекта:

```
Object name: «наименование объекта»
```

Вывод состояния объекта занимает две строки. Первая строка:

```
Object name: «наименование объекта»      Copy: «номер копии»      Shared: «значение свойства с общим доступом»
```

Вторая строка:

```
Array:  «целое число»  «целое число» . . . «целое число»
```

Сообщение конструктора копии имеет следующий шаблон:

```
Construcrot copy Object name: «наименование объекта»    Copy: «номер копии»
```

Если объект с заданным номером копии не найден, то выводиться сообщение:

```
A copy of object number «номер копии» was not found.
```

Сообщение деструктора копии имеет следующий шаблон:

```
Destructor Object name: «наименование объекта»    Copy: «номер копии»
```

Пример вывода

```
Object name: Document
Object name: Document    Copy: 0    Shared: 0
Array:  1  2  3  4  5
Construcrot copy Object name: Document    Copy: 1
Object name: Document    Copy: 1    Shared: 0
Array:  2  3  4  5  6
Construcrot copy Object name: Document    Copy: 2
Object name: Document    Copy: 2    Shared: 0
Array:  4  5  6  7  8
A copy of object number 8 was not found.
Construcrot copy Object name: Document    Copy: 3
Object name: Document    Copy: 3    Shared: 0
Array:  5  6  7  8  9
Construcrot copy Object name: Document    Copy: 4
Object name: Document    Copy: 4    Shared: 77
Array:  5  6  7  8  9
Construcrot copy Object name: Document    Copy: 5
Object name: Document    Copy: 5    Shared: 77
Array:  7  8  9  10  11
Destructor Object name: Document    Copy: 5
A copy of object number 5 was not found.
Object name: Document    Copy: 1    Shared: 77
Array:  2  3  4  5  6
```

2 МЕТОД РЕШЕНИЯ

Для решения задачи используется:

- объект origin класса Obj предназначен для демонстрации работы программы;
- функция main для определения входной точки программы;
- функция process для вызова метода вывода состояния объекта;
- функция findObj для нахождение объекта по номеру копии;
- объекты без наименования класса Obj предназначены для построения системы;
- стандартная библиотека ввода-вывода;
- библиотека string;
- библиотека vector;
- указатели;
- заголовочный файл.

Класс Obj:

- свойства/поля:
    - поле хранит наименование объекта:
        - наименование — m_name;
        - тип — std::string;
        - модификатор доступа — private;
    - поле хранит номер копии:
        - наименование — m_number;
        - тип — int;
        - модификатор доступа — private;
    - поле хранит размер массива:
        - наименование — m_size;

        - тип — int;
        - модификатор доступа — private;
    - поле динамический массив:
        - наименование — m_array;
        - тип — int*;
        - модификатор доступа — private;
    - поле свойство с общим доступом:
        - наименование — value;
        - тип — int*;
        - модификатор доступа — public;
- функционал:
    - метод Obj — параметризированный конструктор;
    - метод Obj — параметризированный конструктор;
    - метод ~Obj — деструктор;
    - метод GetNumber — возвращает номер копии объекта;
    - метод PrintState — выводит состояние объекта;
    - метод SetValue — устанавливает значение свойства с общим доступом.

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 Obj

Функционал: Параметризированный конструктор.

Параметры: std::string name.

Алгоритм конструктора представлен в таблице 1.

Таблица 1 – Алгоритм конструктора класса Obj

№	Предикат	Действия	№ перехода
1		Инициализация value нулём	2
2		Ввод m_size	3
3		Выделение памяти для целочисленного динамического массива m_array, размерностью значения m_size с помощью оператора new	4
4		Инициализация целочисленной переменной i = 0	5
5	i < m_size	Ввод m_array[i]	5
			∅

3.2 Алгоритм конструктора класса Obj

Функционал: Параметризированный конструктор.

Параметры: const Obj& other, int number.

Алгоритм конструктора представлен в таблице 2.

Таблица 2 – Алгоритм конструктора класса Obj

№	Предикат	Действия	№ перехода
1		Инициализация value значением other.value	2
2		Инициализация m_size значением other.m_size	3
3		Выделение памяти для целочисленного динамического массива m_array, размерностью значения m_size с помощью оператора new	4
4		Инициализация целочисленной переменной i = 0	5
5	i < m_size	Присвоение m_array[i] значения other.m_array[i], сложенного с m_number	5
			6
6		Вывод "Construcrot copy Object name: " , m_name, четырех пробельных символов, "Copy: ", m_number, символа переноса строки	∅

3.3 Алгоритм деструктора класса Obj

Функционал: Деструктор.

Параметры: нет.

Алгоритм деструктора представлен в таблице 3.

Таблица 3 – Алгоритм деструктора класса Obj

№	Предикат	Действия	№ перехода
1		Вывод наименования и номера копии объекта	2
2		Освобождение памяти, выделенной на динамический массив m_array, с помощью оператора delete	∅

3.4 Алгоритм метода GetNumber класса Obj

Функционал: Возвращает номер копии объекта.

Параметры: нет.

Возвращаемое значение: int.

Алгоритм метода представлен в таблице 4.

Таблица 4 – Алгоритм метода GetNumber класса Obj

№	Предикат	Действия	№ перехода
1		Возврат m_number	∅

3.5 Алгоритм метода PrintState класса Obj

Функционал: Выводит состояние объекта.

Параметры: нет.

Возвращаемое значение: void.

Алгоритм метода представлен в таблице 5.

Таблица 5 – Алгоритм метода PrintState класса Obj

№	Предикат	Действия	№ перехода
1		Вывод наименования объекта, номера копии, общедоступного свойства	2
2		Вывод "Array: "	3
3		Инициализация целочисленной переменной i = 0	4
4	i < m_size	Вывод двух пробельных символов и значение m_array[i]	4
			5
5		Вывод символа переноса строки	∅

3.6 Алгоритм метода SetValue класса Obj

Функционал: Устанавливает значение свойства с общим доступом.

Параметры: int newValue.

Возвращаемое значение: void.

Алгоритм метода представлен в таблице 6.

Таблица 6 – Алгоритм метода SetValue класса Obj

№	Предикат	Действия	№ перехода
1		Присвоение общедоступному свойству value значение параметра newValue	∅

3.7 Алгоритм функции process

Функционал: Вызов метода вывода состояния объекта.

Параметры: Obj obj.

Возвращаемое значение: void.

Алгоритм функции представлен в таблице 7.

Таблица 7 – Алгоритм функции process

№	Предикат	Действия	№ перехода
1		Вызов obj.PrintState()	∅

3.8 Алгоритм функции findObj

Функционал: Нахождение объекта по номеру копии.

Параметры: std::vector<Obj*>& objects, int number.

Возвращаемое значение: Obj*.

Алгоритм функции представлен в таблице 8.

Таблица 8 – Алгоритм функции findObj

№	Предикат	Действия	№ перехода
1		Инициализация целочисленной переменной i = 0	2
2	i < objects.size()		3

№	Предикат	Действия	№ перехода
			4
3	objects[i]->GetNumber() == number	Возврат objects[i]	∅
			2
4		Вывод "A copy of object number ", number, " was not found."	5
5		Возврат нулевого адреса	∅

3.9 Алгоритм функции main

Функционал: Основная функция программы.

Параметры: нет.

Возвращаемое значение: int.

Алгоритм функции представлен в таблице 9.

Таблица 9 – Алгоритм функции main

№	Предикат	Действия	№ перехода
1		Объявление двух переменных строкового типа name, command	2
2		Инициализация целочисленной переменной number значением 1, объявление целочисленной переменной value	3
3		Объявление вектора указателей objects на объекты класса Obj	4
4		Ввод name	5
5		Вывод "Object name: ", name, символа переноса строки	6
6		Создание объекта origin класса Obj с помощью оператора new	7

№	Предикат	Действия	№ перехода
7		Вызов origin->PrintState()	8
8		Добавление объекта origin в вектор objects	9
9	поток ввода не пустой	Инициализация целочисленной переменной copyNum значением 0	10
			∅
10	command это "end"		∅
	command это "copy"	Ввод copyNum	11
	command это "function"	Ввод copyNum	16
	command это "state"	Ввод copyNum	20
	command это "shared"	Ввод copyNum, value	23
11		Присвоение объекту origin класса Obj результата вызова функции findObj	12
12	origin указывает на нулевой адрес		9
			13
13		Создание объекта copy класса Obj с помощью оператора new с передачей разыменнованного указателя на объект origin, number, увеличенный на 1 в качестве аргументов	14
14		Добавление объекта copy в вектор objects	15
15		Вызов copy->PrintState()	9
16		Присвоение объекту origin класса Obj результата вызова функции findObj	17
17	origin указывает на нулевой адрес		9
			18
18		Создание объекта copy класса Obj с помощью оператора new с передачей разыменнованного указателя на объект copy, number, увеличенный на	19

№	Предикат	Действия	№ перехода
		1 в качестве аргументов	
19		process(*copy); Вызов функции process с передачей разыменнованного указателя на объект copy в качестве аргумента	9
20		Присвоение объекту origin класса Obj результата вызова функции findObj	21
21	obj указывает на нулевой адрес		9
			22
22		Вызов obj->PrintState()	9
23		Присвоение объекту origin класса Obj результата вызова функции findObj	24
24	obj указывает на нулевой адрес		9
			25
25		Вызов obj->SetValue(value)	9

4 БЛОК-СХЕМЫ АЛГОРИТМОВ

Представим описание алгоритмов в графическом виде на рисунках 1-10.

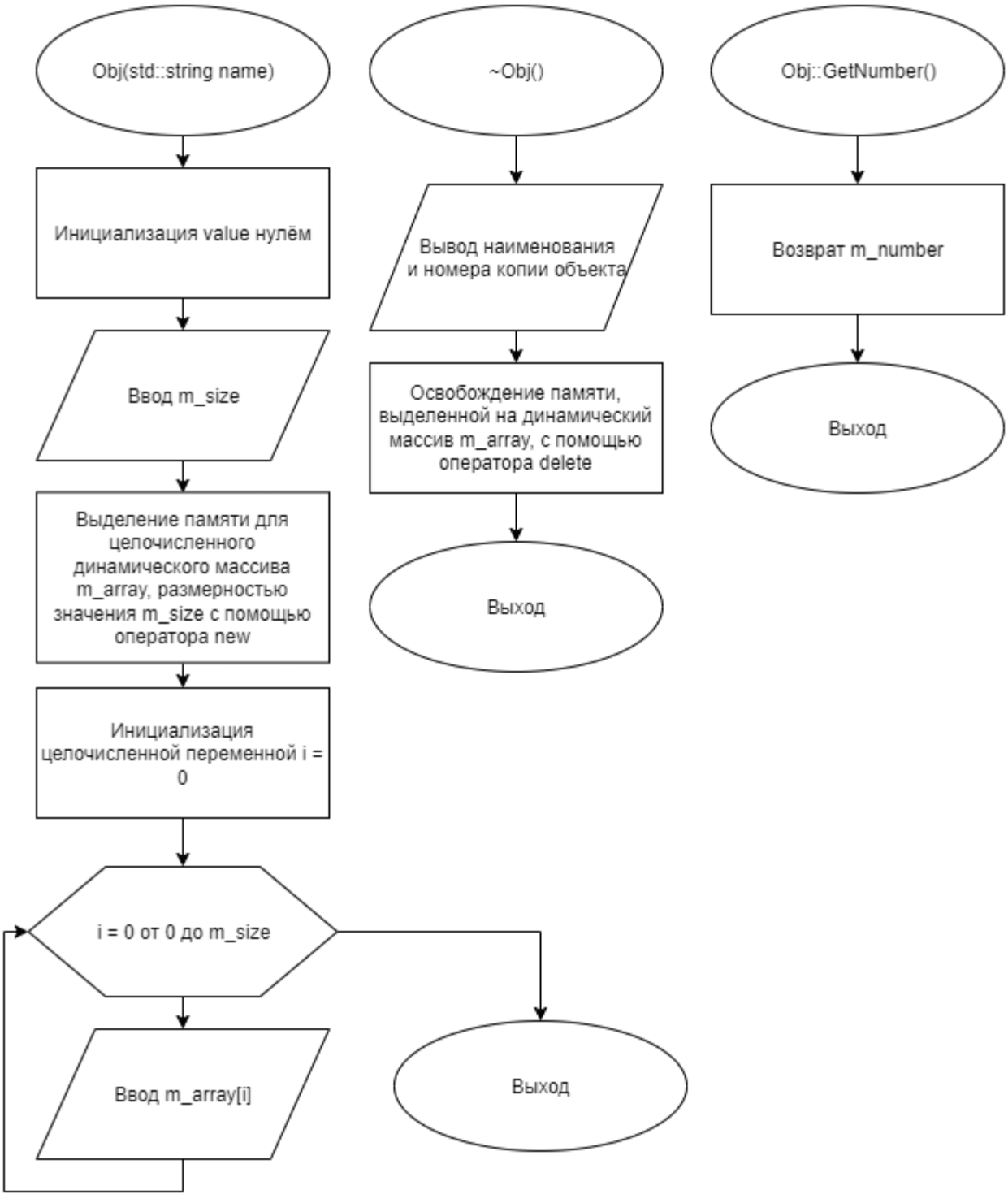

Рисунок 1 – Блок-схема алгоритма

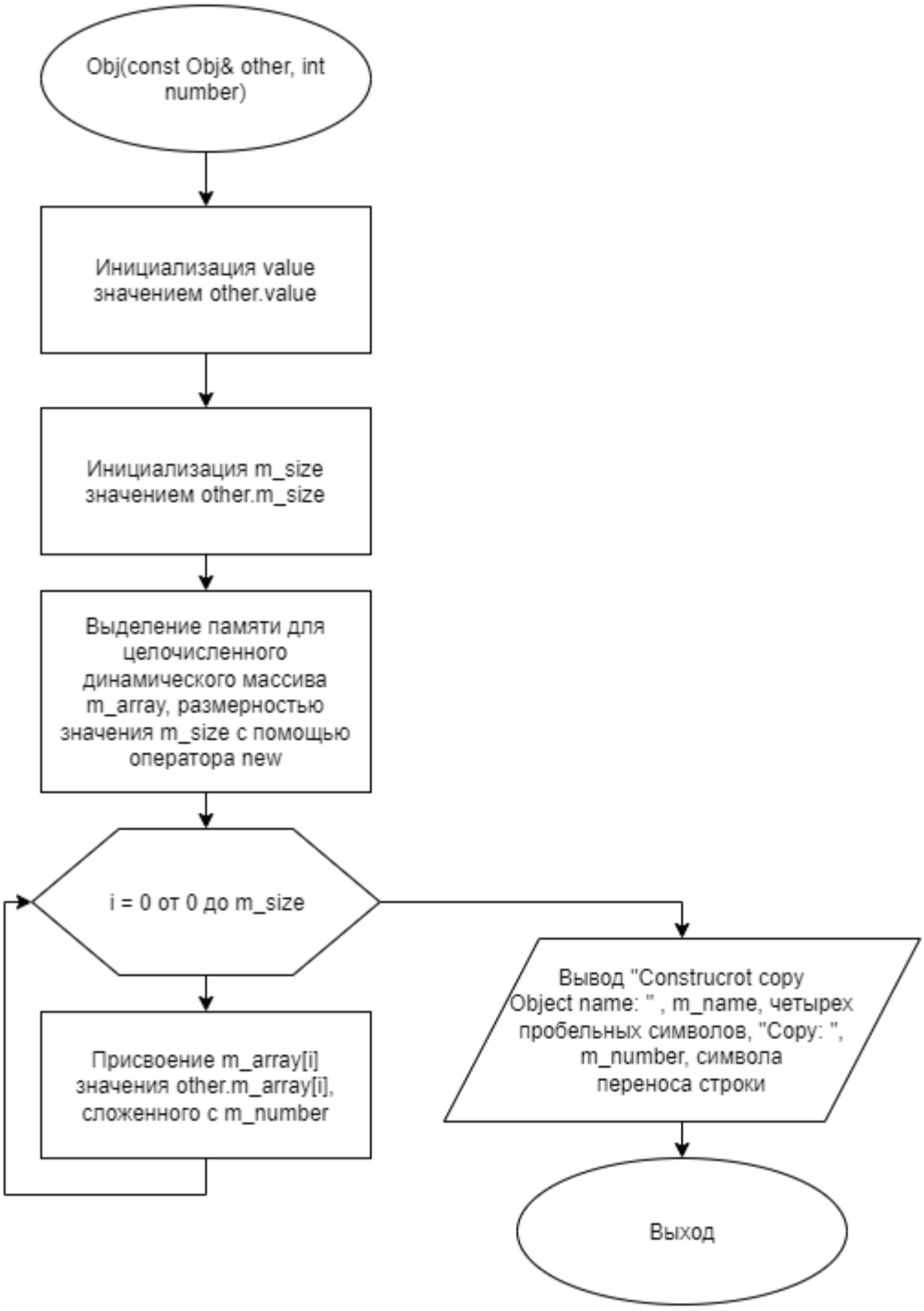

Рисунок 2 – Блок-схема алгоритма

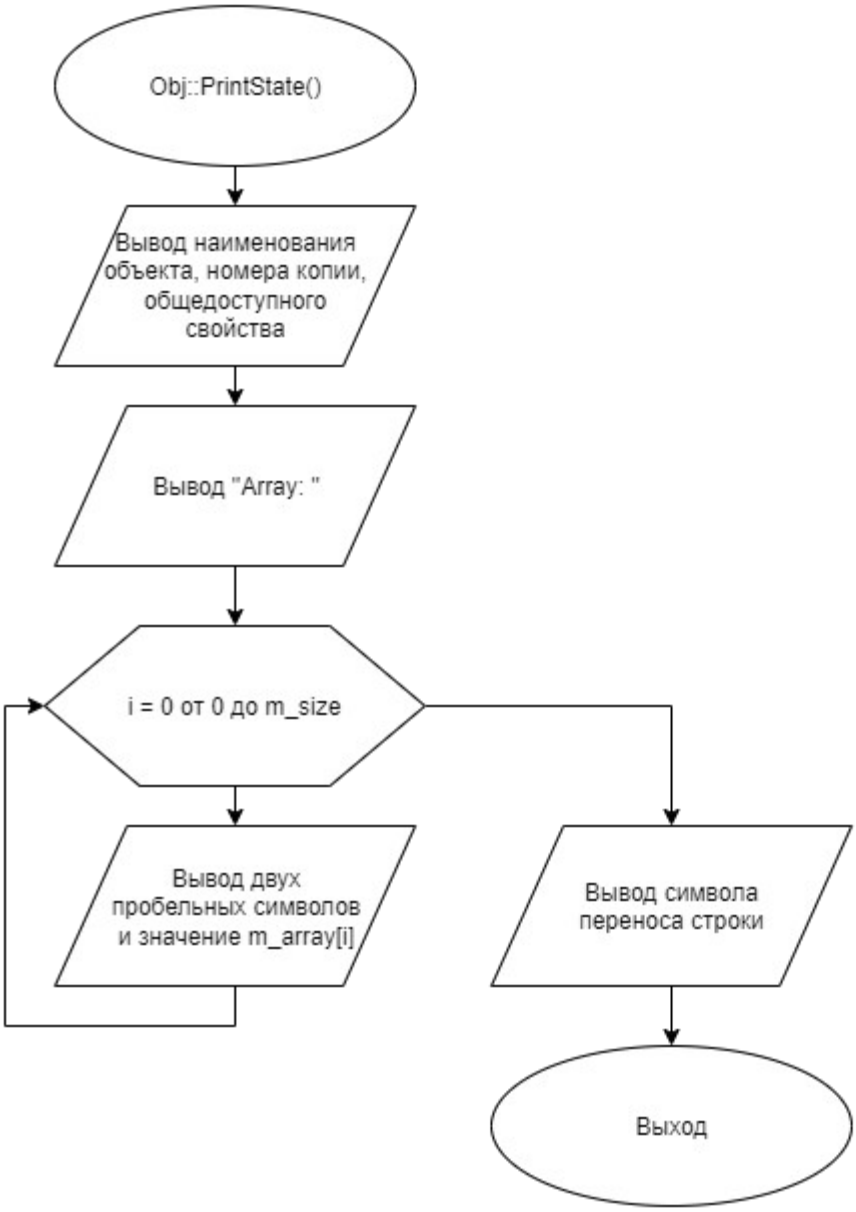

Рисунок 3 – Блок-схема алгоритма

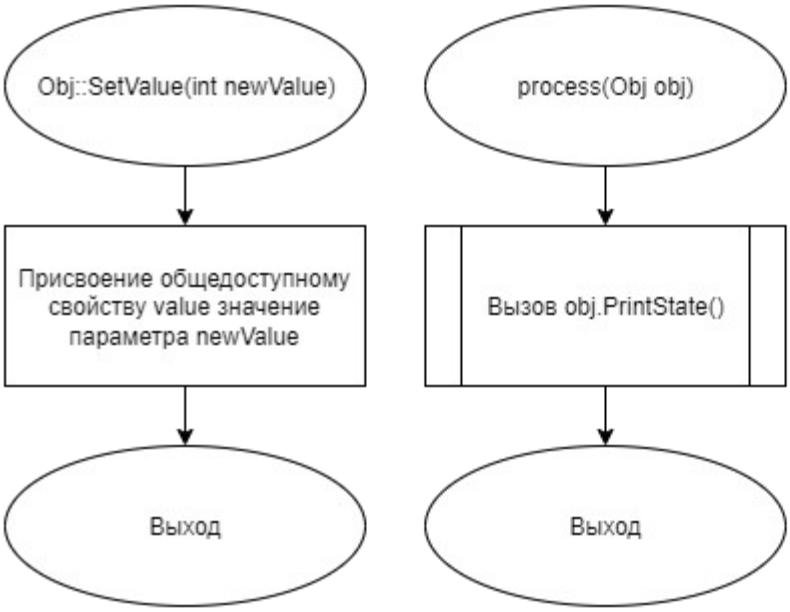

Рисунок 4 – Блок-схема алгоритма

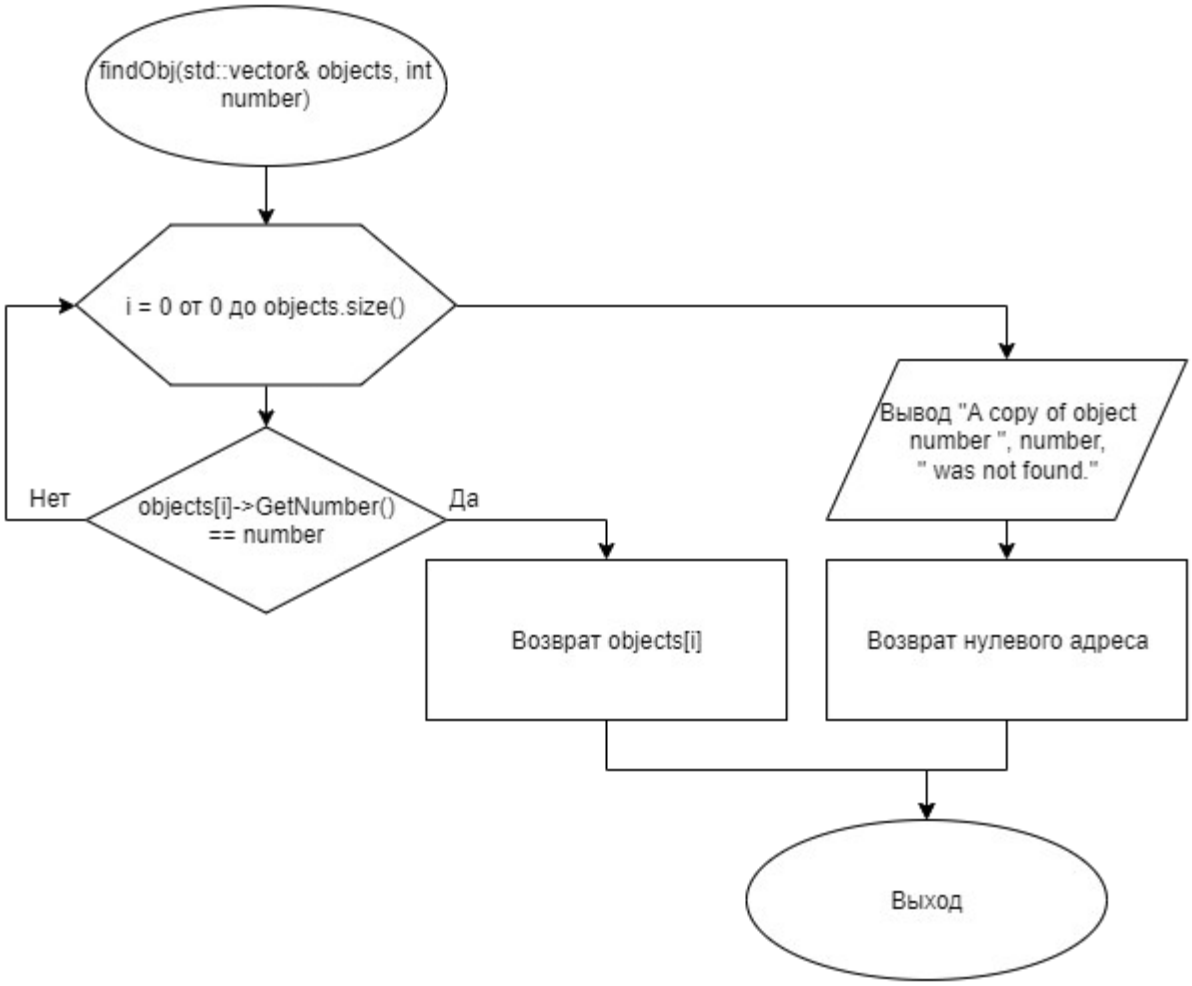

Рисунок 5 – Блок-схема алгоритма

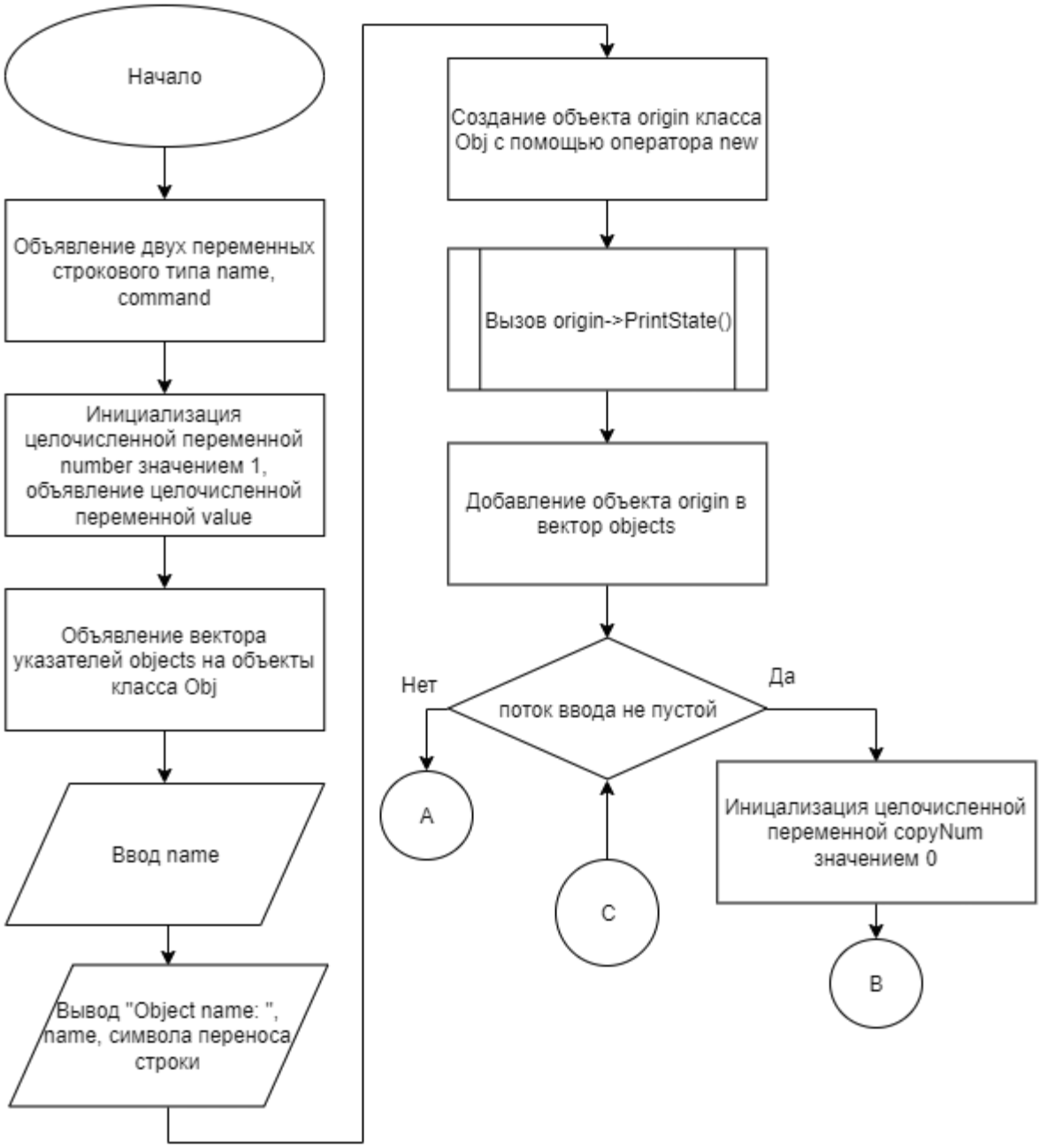

Рисунок 6 – Блок-схема алгоритма

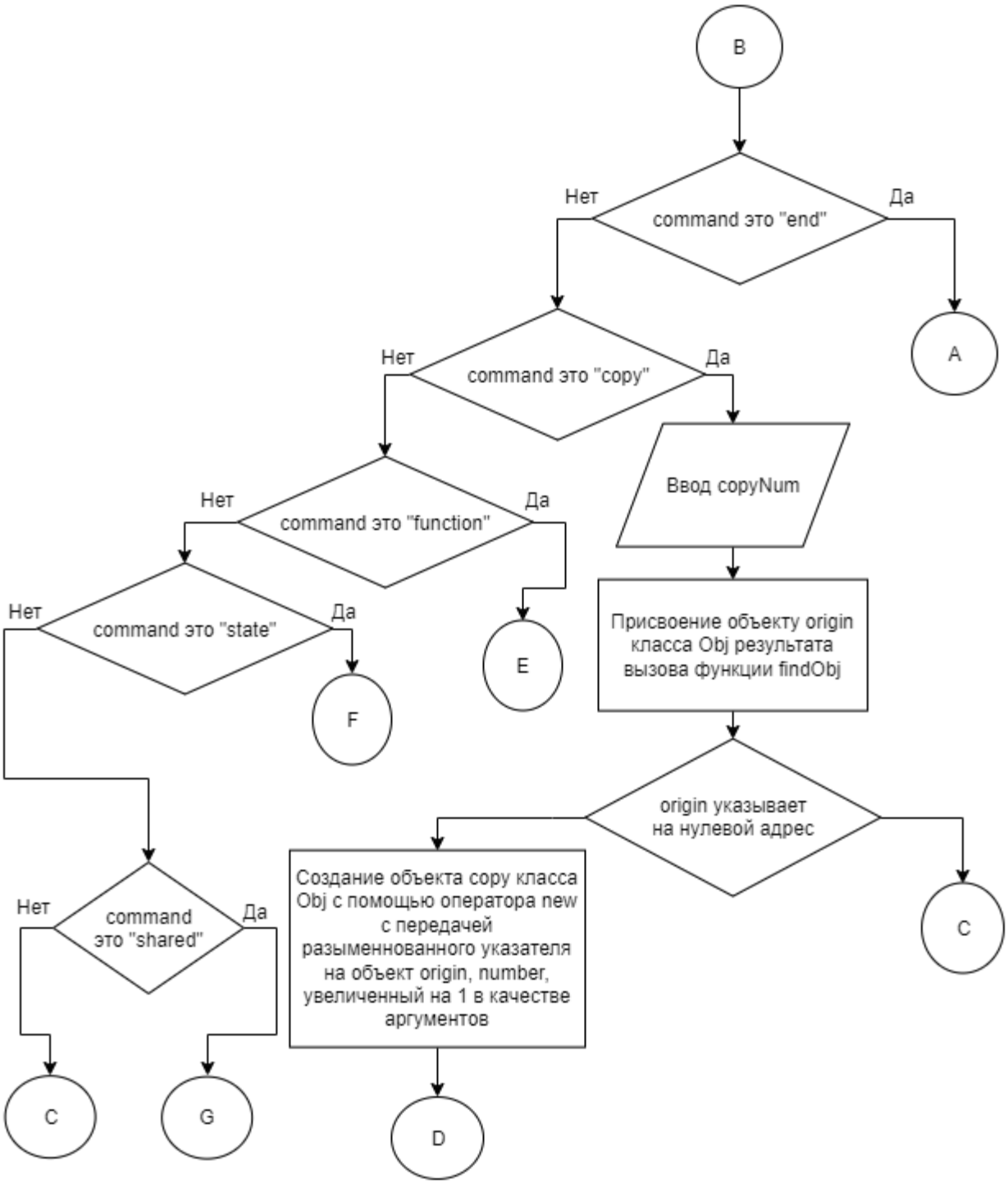

Рисунок 7 – Блок-схема алгоритма

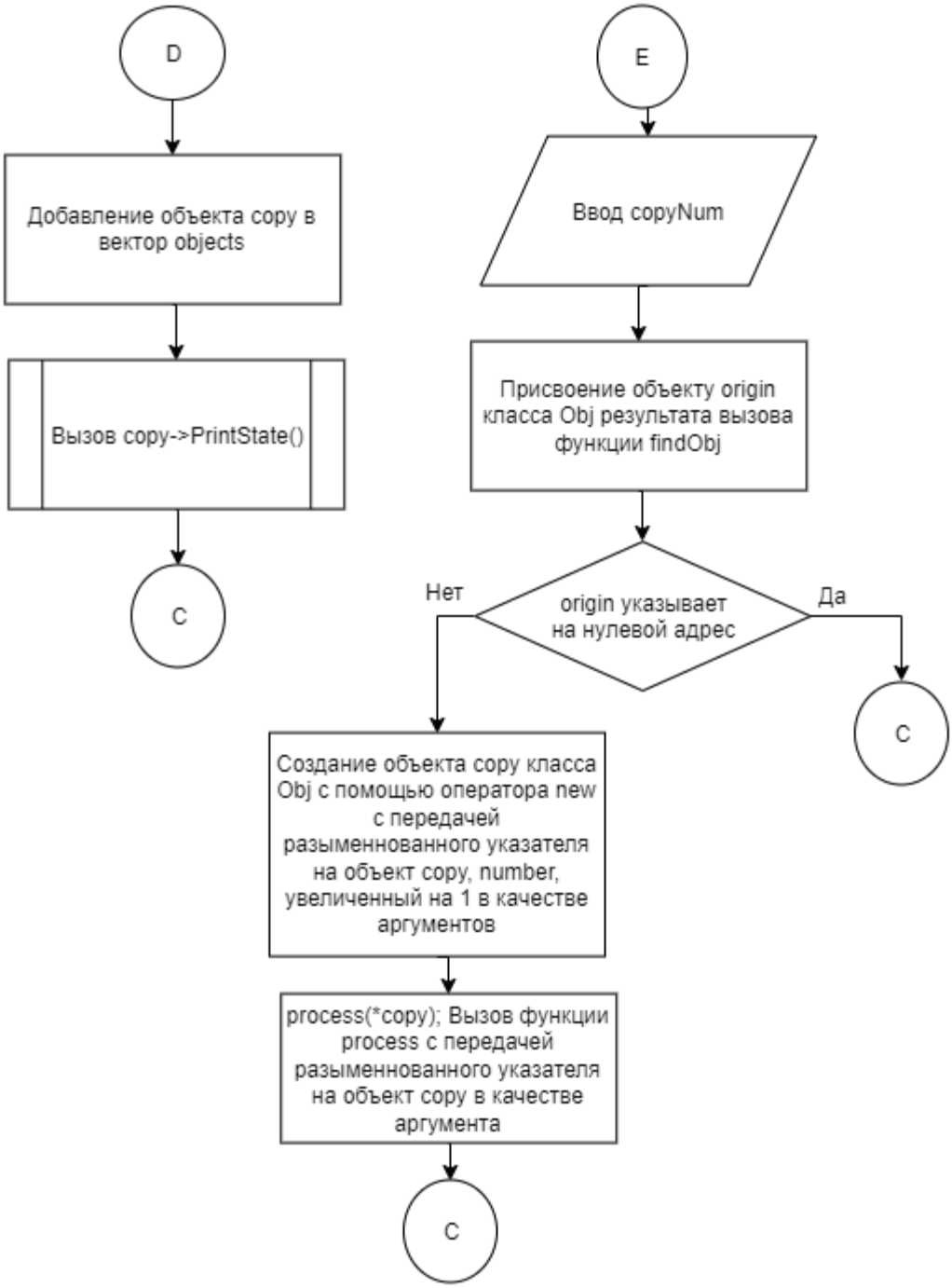

Рисунок 8 – Блок-схема алгоритма

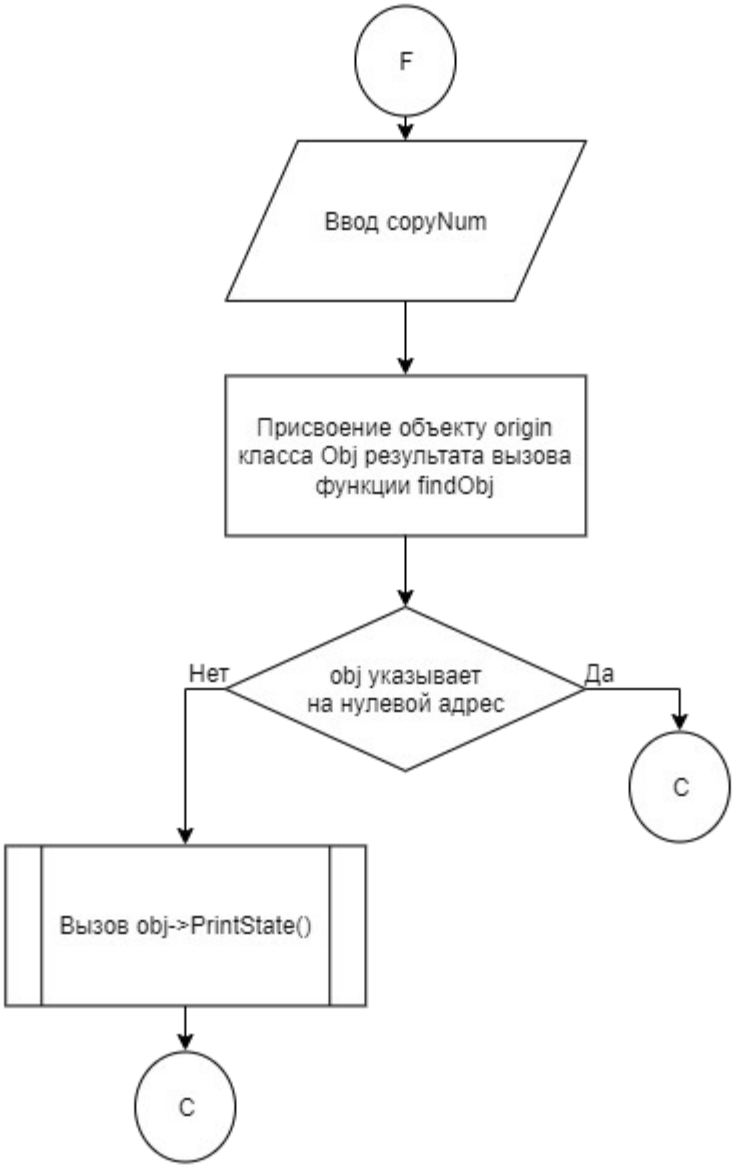

Рисунок 9 – Блок-схема алгоритма

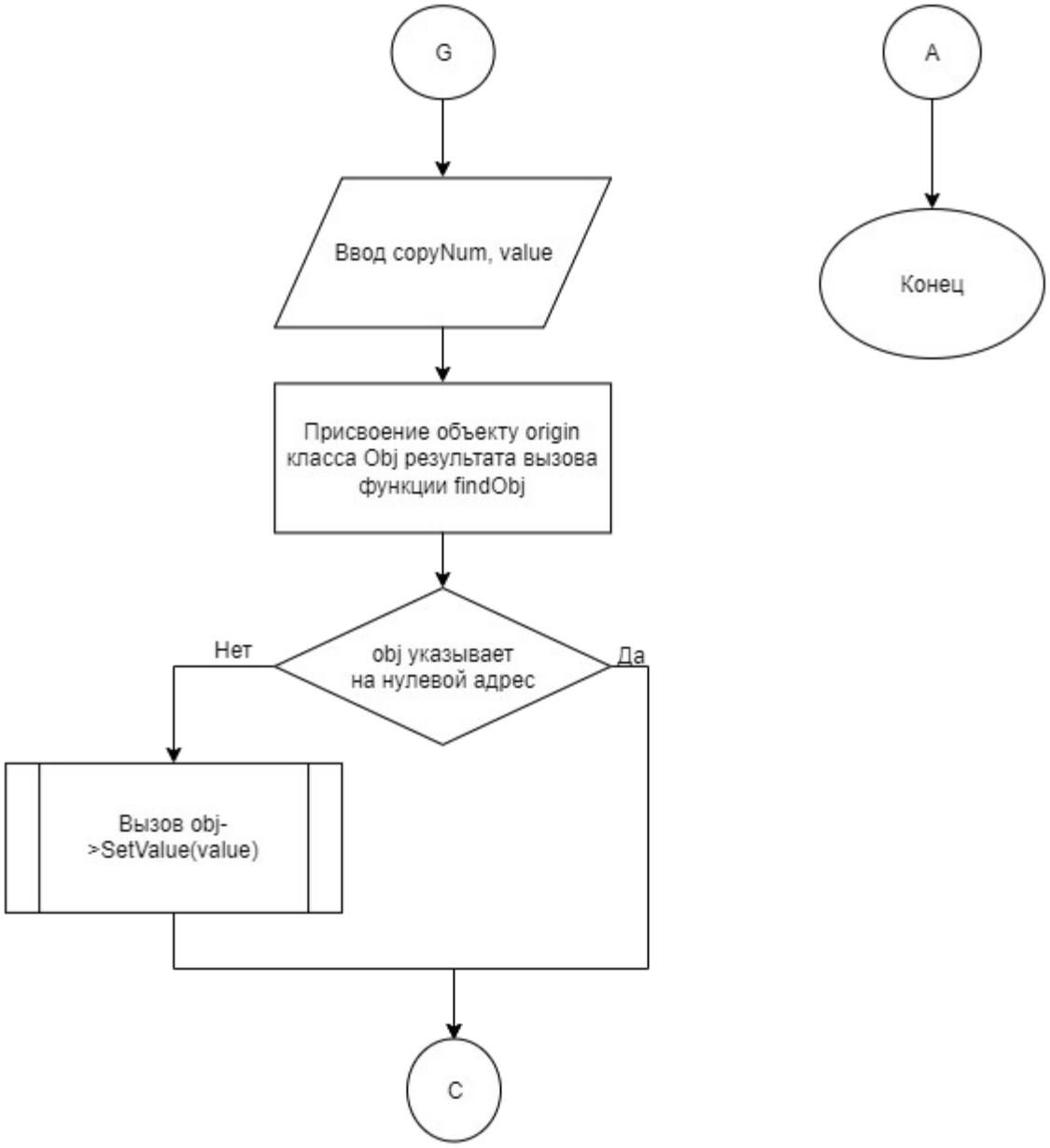

Рисунок 10 – Блок-схема алгоритма

5 КОД ПРОГРАММЫ

Программная реализация алгоритмов для решения задачи представлена ниже.

5.1 Файл main.cpp

Листинг 1 – main.cpp

```
#include "Obj.h"
#include <vector>

void process(Obj obj)
{
   obj.PrintState();
}

Obj* findObj(std::vector<Obj*>& objects, int number)
{
   for (int i = 0; i < objects.size(); i++)
   {
      if (objects[i]->GetNumber() == number)
         return objects[i];
   }

   std::cout << "A copy of object number " << number << " was not found." <<
std::endl;
   return nullptr;
}

int main()
{
   std::string name, command;
   int number = 1, value;
   std::vector<Obj*> objects;

   std::cin >> name;
   std::cout << "Object name: " << name << std::endl;
   Obj* origin = new Obj(name);
   origin->PrintState();

   objects.push_back(origin);

   while (std::cin >> command)
   {
      int copyNum = 0;

      if (command == "end")
```

```
            {
               break;
            }
            else if (command == "copy")
            {
               std::cin >> copyNum;
               Obj* origin = findObj(objects, copyNum);
               if (origin == nullptr) continue;

               Obj* copy = new Obj(*origin, number++);
               objects.push_back(copy);
               copy->PrintState();
            }
            else if (command == "function")
            {
               std::cin >> copyNum;
               Obj* origin = findObj(objects, copyNum);
               if (origin == nullptr) continue;

               Obj* copy = new Obj(*origin, number++);
               process(*copy);
            }
            else if (command == "state")
            {
               std::cin >> copyNum;
               Obj* obj = findObj(objects, copyNum);
               if (obj == nullptr) continue;

               obj->PrintState();
            }
            else if (command == "shared")
            {
               std::cin >> copyNum >> value;
               Obj* obj = findObj(objects, copyNum);
               if (obj == nullptr) continue;

               obj->SetValue(value);
            }
         }

         return 0;
      }
```

5.2 Файл Obj.cpp

Листинг 2 – Obj.cpp

```
#include "Obj.h"

Obj::Obj(std::string name) : m_name(name), m_number(0)
```

```
{
   value = new int(0);

   std::cin >> m_size;
   m_array = new int[m_size];

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

Obj::Obj(const   Obj&   other,   int   number)   :   m_name(other.m_name),
m_number(number)
{
   value = other.value;
   m_size = other.m_size;

   m_array = new int[m_size];
   for (int i = 0; i < m_size; i++)
   {
      m_array[i] = other.m_array[i] + m_number;
   }

   std::cout << "Construcrot copy Object name: " << m_name
      << "    " << "Copy: " << m_number << std::endl;
}

Obj::~Obj()
{
   std::cout << "Destructor Object name: " << m_name
      << "    Copy: " << m_number << std::endl;
   delete m_array;
}

int Obj::GetNumber()
{
   return m_number;
}

void Obj::PrintState()
{
   std::cout << "Object name: " << m_name << "    Copy: " <<
      m_number << "    Shared: " << *value << std::endl;

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

void Obj::SetValue(int newValue)
{
```

```
   *value = newValue;
}
```

5.3 Файл Obj.h

Листинг 3 – Obj.h

```
#ifndef __OBJ__H
#define __OBJ__H

#include <string>
#include <iostream>

class Obj
{
   std::string m_name;
   int m_number;
   int m_size;
   int* m_array;
public:
   int* value;

   Obj(std::string name);
   Obj(const Obj& other, int number);
   ~Obj();

   int GetNumber();
   void PrintState();
   void SetValue(int newValue);
};

#endif
```

6 ТЕСТИРОВАНИЕ

Результат тестирования программы представлен в таблице 10.

Таблица 10 – Результат тестирования программы

Входные данные	Ожидаемые выходные данные	Фактические выходные данные
Document 5 1 2 3 4 5 copy 0 copy 1 shared 8 99 copy 1 shared 2 77 copy 0 function 1 state 5 state 1 end	Object name: Document Object name: Document Copy: 0 Shared: 0 Array: 1 2 3 4 5 Construcrot copy Object name: Document Copy: 1 Object name: Document Copy: 1 Shared: 0 Array: 2 3 4 5 6 Construcrot copy Object name: Document Copy: 2 Object name: Document Copy: 2 Shared: 0 Array: 4 5 6 7 8 A copy of object number 8 was not found. Construcrot copy Object name: Document Copy: 3 Object name: Document Copy: 3 Shared: 0 Array: 5 6 7 8 9 Construcrot copy Object name: Document Copy: 4 Object name: Document Copy: 4 Shared: 77 Array: 5 6 7 8 9 Construcrot copy Object name: Document Copy: 5 Object name:	Object name: Document Object name: Document Copy: 0 Shared: 0 Array: 1 2 3 4 5 Construcrot copy Object name: Document Copy: 1 Object name: Document Copy: 1 Shared: 0 Array: 2 3 4 5 6 Construcrot copy Object name: Document Copy: 2 Object name: Document Copy: 2 Shared: 0 Array: 4 5 6 7 8 A copy of object number 8 was not found. Construcrot copy Object name: Document Copy: 3 Object name: Document Copy: 3 Shared: 0 Array: 5 6 7 8 9 Construcrot copy Object name: Document Copy: 4 Object name: Document Copy: 4 Shared: 77 Array: 5 6 7 8 9 Construcrot copy Object name: Document Copy: 5 Object name:

Входные данные	Ожидаемые выходные данные	Фактические выходные данные
	Document Copy: 5 Shared: 77 Array: 7 8 9 10 11 Destructor Object name: Document Copy: 5 A copy of object number 5 was not found. Object name: Document Copy: 1 Shared: 77 Array: 2 3 4 5 6	Document Copy: 5 Shared: 77 Array: 7 8 9 10 11 Destructor Object name: Document Copy: 5 A copy of object number 5 was not found. Object name: Document Copy: 1 Shared: 77 Array: 2 3 4 5 6

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).