МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

« МИРЭА Российский технологический университет»

РТУ МИРЭА

Институт Информационных технологий

Кафедра Вычислительной техники

УЧЕБНОЕ ЗАДАНИЕ

по дисциплине

« Объектно-ориентированное программирование»

Наименование задачи:

« Задание 4_1_1 »

С тудент группы	ИКБО-33-21	Евсеев Г.Е.
Руководитель практики	Ассистент	Асадова Ю.С.
Работа представлена	«___»_____ 2022 г.	_____
		(подпись студента)
Оценка		_____
		(подпись руководителя)

Москва 2022

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Постановка задачи

Для организации иерархического построения объектов необходимо разработать базовый класс, который содержит функционал и свойства для построения иерархии объектов.
В последующем, в приложениях использовать этот класс как базовый для всех создаваемых классов. Это позволит включать любой объект в состав дерева иерархии объектов.

Создать базовый класс со следующими элементами:

Свойства:

- наименование объекта (строкового типа);
- указатель на головной объект для текущего объекта (для корневого объекта значение указателя равно 0);
- массив указателей на объекты, подчиненные к текущему объекту в дереве иерархии.

Функционал:

- параметризированный конструктор с параметрами: указатель на головной объект в дереве иерархии и наименование объекта (имеет значение по умолчанию);
- метод определения имени объекта;
- метод получения имени объекта;
- метод вывода наименований объектов в дереве иерархии слева направо и сверху вниз;
- метод переопределения головного объекта для текущего в дереве иерархии;
- метод получения указателя на головной объект текущего объекта.

Для построения дерева иерархии объектов в качестве корневого объекта используется объект приложение. Класс объекта приложения наследуется от базового класса. Объект приложение реализует следующий функционал:
− метод построения исходного дерева иерархии объектов (конструирования программы-системы, изделия);
− метод запуска приложения (начало функционирования системы, выполнение алгоритма решения задачи).

Написать программу, которая последовательно строит дерево иерархии объектов, слева направо и сверху вниз.
Переход на новый уровень происходит только от правого (последнего) объекта предыдущего уровня.
Для построения дерева использовать объекты двух производных классов, наследуемых от базового. Каждый объект имеет уникальное имя.

Построчно, по уровням вывести наименования объектов построенного иерархического дерева.

Основная функция должна иметь следующий вид:

```
int main()
{
   cl_application  ob_cl_application ( nullptr );
   ob_cl_application.bild_tree_objects ( );  // построение дерева объектов
   return ob_cl_application.exec_app ( );    // запуск системы
}
```

Наименование класса cl_application и идентификатора корневого объекта ob_cl_application могут быть изменены разработчиком.

Описание входных данных

Первая строка:

«имя корневого объекта»

Вторая строка и последующие строки:

«имя головного объекта»«имя подчиненного объекта»

Создается подчиненный объект и добавляется в иерархическое дерево.

Если «имя головного объекта» равняется «имени подчиненного объекта», то новый объект не создается и построение дерева объектов завершается.

Примерввода

Object_root

Object_root Object_1

Object_root Object_2

Object_root Object_3

Object_3 Object_4

Object_3 Object_5

Object_6 Object_6

Дерево объектов, которое будет построено по данному примеру:

Object_root

    Object_1

    Object_2

    Object_3

        Object_4

Object_5

Описание выходных данных

Первая строка:

«имя корневого объекта»

Вторая строка и последующие строки имена головного и подчиненных объектов очередного уровня разделенных двумя пробелами.

«имя головного объекта»«имя подчиненного объекта»[[«имя подчиненного объекта»] …….]

Пример вывода

Object_root

Object_root Object_1 Object_2 Object_3

Object_3 Object_4 Object_5

Метод решения

Для решения задачи используются:

- Для решения задачи используются:
- Объекты стандартного потока ввода-вывода
- Условный оператор
- Оператор цикла
- Объект класса vector стандартной библиотеки vector
- Объект класса string стандартной библиотеки string
- Объект класса cl_base
- Объект класса cl_1
- Объект класса cl_2
- Объект класса cl_application
- Класс cl_base:
    - Свойства/поля:
        - Поле, отвечающее за наименование объекта:
            - Наименование - name;
            - Тип - строковый;
            - Модификатор доступа - закрытый;
        - Ссылка на элемент-родитель:
            - Наименование - parent;
            - Тип - указатель на объект класса cl_base
            - Модификатор доступа - закрытый;
        - Список ссылок на дочерние элементы:
            - Наименование - children;
            - Тип - вектор указателей на объекты класса cl_base;
            - Модификатор доступа - закрытый;
    - Методы:

- Конструктор cl_base:
  - Функционал - параметризированный конструктор с 9 параметрами имени и ссылки на родительский элемент
- Деструктор ~cl_base:
  - Функционал - деструктор, уничтожающий данный объект и его потомков;
- Метод get_name:
  - Функционал: возвращает имя объекта;
- Метод set_name:
  - Функционал - устанавливает имя объекта
- Метод get_parent:
  - Функционал - возвращает указатель на родителя данного объекта;
- Метод set_parent:
  - Функционал - устанавливает нового родителя данному объекту
- Метод print:
  - Функционал - выводит на экран иерархию объектов класса, считая данный объект корневым

- Класс cl_application:
  - Методы:
    - Конструктор cl_application:
      - Функционал - параметризированный конструктор с параметром имени корневого объекта, унаследованный от конструктора класса Base;

- Метод build_tree_objects:
  - Функционал - строит дерево иерархии, используя 10 данные введенные данные
- Метод exec_app:
  - Функционал - запускает выполнение программы;

- Класс cl_1:
  - Методы:
    - Конструктор cl_1:
      - Параметризированный конструктор с параметрами имени объекта и указателя на его родителя, унаследованный от конструктора класса Base;

- Класс cl_2:
  - Методы:
    - Конструктор cl_2:
      - Параметризированный конструктор с параметрами имени объекта и указателя на его родителя, унаследованный от конструктора класса Base;

№	Имя класса	Классы-наследники	Модификатор доступа при наследовании	Описание	Номер	Комментарий
1	cl_base	cl_application	public	Базовый класс в иерархии классов. Содержит основные поля и методы.	3	
		cl_2	public		2	
		cl_1	public		1	
2	cl_application			Класс корневого объекта (приложения		

				)		
3	cl_1			Класс объектов, подчиненных корневому объекту класса Application		
	cl_2			Класс объектов, подчиненных корневому объекту класса Application		

Описание алгоритма

Согласно этапам разработки, после определения необходимого инструментария в разделе «Метод», составляются подробные описания алгоритмов для методов классов и функций.

Конструктор класса: cl_base_cl_application_cl_1_cl_2

Модификатор доступа: public

Функционал: конструктор класса

Параметры: name - строкового типа, parent - указатель на родителя

Алгоритм конструктора представлен в таблице 2.

Таблица 2. Алгоритм конструктора класса cl_base_cl_application_cl_1_cl_2

№	Предикат	Действия	№ перехода	Комментарий
1		присваивани имени текущего объекта = name	2	
2		присваивание указателю на родителя текущего объекта = parent	3	
3	указатель на родителя существует	добавление текущего объекта в состав детей от указателя parent	∅	
			∅	

Деструктор класса: cl_base

Модификатор доступа: public

Функционал: деструктор класса

Параметры: отсутствуют

Алгоритм деструктора представлен в таблице 3.

Таблица 3. Алгоритм деструктора класса cl_base

№	Предикат	Действия	№ перехода	Комментарий
1	i < размер массива children	удаляем ячейку i из children	1	
			∅	

Класс объекта: cl_base

Модификатор доступа: public

Метод: get_name

Функционал: возвращает имя объекта

Параметры: отсутствуют

Возвращаемое значение: name - строкового типа

Алгоритм метода представлен в таблице 4.

Таблица 4. Алгоритм метода get_name класса cl_base

№	Предикат	Действия	№ перехода	Комментарий
1		вернуть name	∅	

Класс объекта: cl_base

Модификатор доступа: public

Метод: set_name

Функционал: устанавливает имя объекта

Параметры: name - строкового типа

Возвращаемое значение: отсутствуют

Алгоритм метода представлен в таблице 5.

Таблица 5. Алгоритм метода set_name класса cl_base

№	Предикат	Действия	№ перехода	Комментарий
1		присвоить имени текущего обхекта = name	∅	

Класс объекта: cl_base

Модификатор доступа: public

Метод: set_parent

Функционал: устанавливает нового родителя для данного объекта

Параметры: new_parent - указатель на нового родителя

Возвращаемое значение: отсутствуют

Алгоритм метода представлен в таблице 6.

Таблица 6. Алгоритм метода set_parent класса cl_base

№	Предикат	Действия	№ перехода	Комментарий
1	указатель на родителя не пуст		2	
			5	
2	i < размер родительского массива children		3	
			5	
3	значение ячейки родительского массива children[i] =	удалить из родительского children элемент под индексом i	4	

	текущий объект			
			4	
4		увеличить значение i на единицу	2	
5		присвоить parent значение new_parent	6	
6	указатель на нового родителя не пуст	добавить текущий объект в массив детей нового родителя	∅	
			∅	

Класс объекта: cl_base

Модификатор доступа: public

Метод: get_parent

Функционал: возвращает указатель на родителя объекта

Параметры: отсутствуют

Возвращаемое значение: значение: указатель на объект класса cl_base

Алгоритм метода представлен в таблице 7.

Таблица 7. Алгоритм метода get_parent класса cl_base

№	Предикат	Действия	№ перехода	Комментарий
1		вернуть parent текущего объекта	∅	

Класс объекта: cl_base

Модификатор доступа: public

Метод: print

Функционал: выводит известных родителей и детей

Параметры: отсутствуют

Возвращаемое значение: отсутствуют

Алгоритм метода представлен в таблице 8.

Таблица 8. Алгоритм метода print класса cl_base

№	Предикат	Действия	№ перехода	Комментарий
1	размер массива children != 0	вывод get_name()	2	
			∅	
2	i < размер массива children	вывод children[i]->get_name()	3	
			4	
3		увеличить i на 1	2	
4	j < размер массива children	вывод children[i]->print()	5	
			∅	
5		увеличить j на 1	4	

Класс объекта: cl_application

Модификатор доступа: public

Метод: build_tree_objects

Функционал: строит дерево иерархии, используя введенные данные

Параметры: отсутствуют

Возвращаемое значение: отсутствуют

Алгоритм метода представлен в таблице 9.

Таблица 9. Алгоритм метода build_tree_objects класса cl_application

№	Предикат	Действия	№ перехода	Комментарий
1		объявление строковых переменных parent_name child_name	2	
2		объявление переменных-указателей на объекты класса cl_base parent_ptr child_ptr	3	
3		ввод с клавиатуры значение переменной parent_name	4	
4		вызов метода set_name для parent_name	5	
5		присвоить parent_ptr = значение текущего объекта	6	
6		опустошение child_ptr	7	
7	истина	ввод с клавиатуры значений переменных parent_name, child_name	8	
			∅	
8	parent_name = child_name		∅	
			9	
9	child_ptr != пустой указатель && child_ptr->get_name() == parent_name	присвоить parent_ptr значение child_ptr	10	
			10	
10	остаток от деления на 2 псевдослучайного числа равен 0	выделяем память для объекта child_ptr класса cl_1 с именем child_name и указателем	7	

		на родителя parent_ptr		
		выделяем память для объекта child_ptr класса cl_2 с именем child_name и указателем на родителя parent_ptr	7	

Класс объекта: cl_application

Модификатор доступа: public

Метод: exec_app

Функционал: запускает выполнение программы

Параметры: отсутствуют

Возвращаемое значение: целого типа

Алгоритм метода представлен в таблице 10.

Таблица 10. Алгоритм метода exec_app класса cl_application

№	Предикат	Действия	№ перехода	Комментарий
1		вызов и вывод метода get_name()	2	
2		вызов метода print()	∅	

Функция: main

Функционал: основная функция

Параметры: отсутствуют

Возвращаемое значение: целого типа

Алгоритм функции представлен в таблице 11.

Таблица 11. Алгоритм функции main

№	Предикат	Действия	№ перехода	Комментарий
1		создаение объекта app класса cl_application	2	
2		вызов метода build_tree_objects() для объекта app класса cl_application	∅	

Блок-схема алгоритма

Представим описание алгоритмов в графическом виде на рисунках ниже.

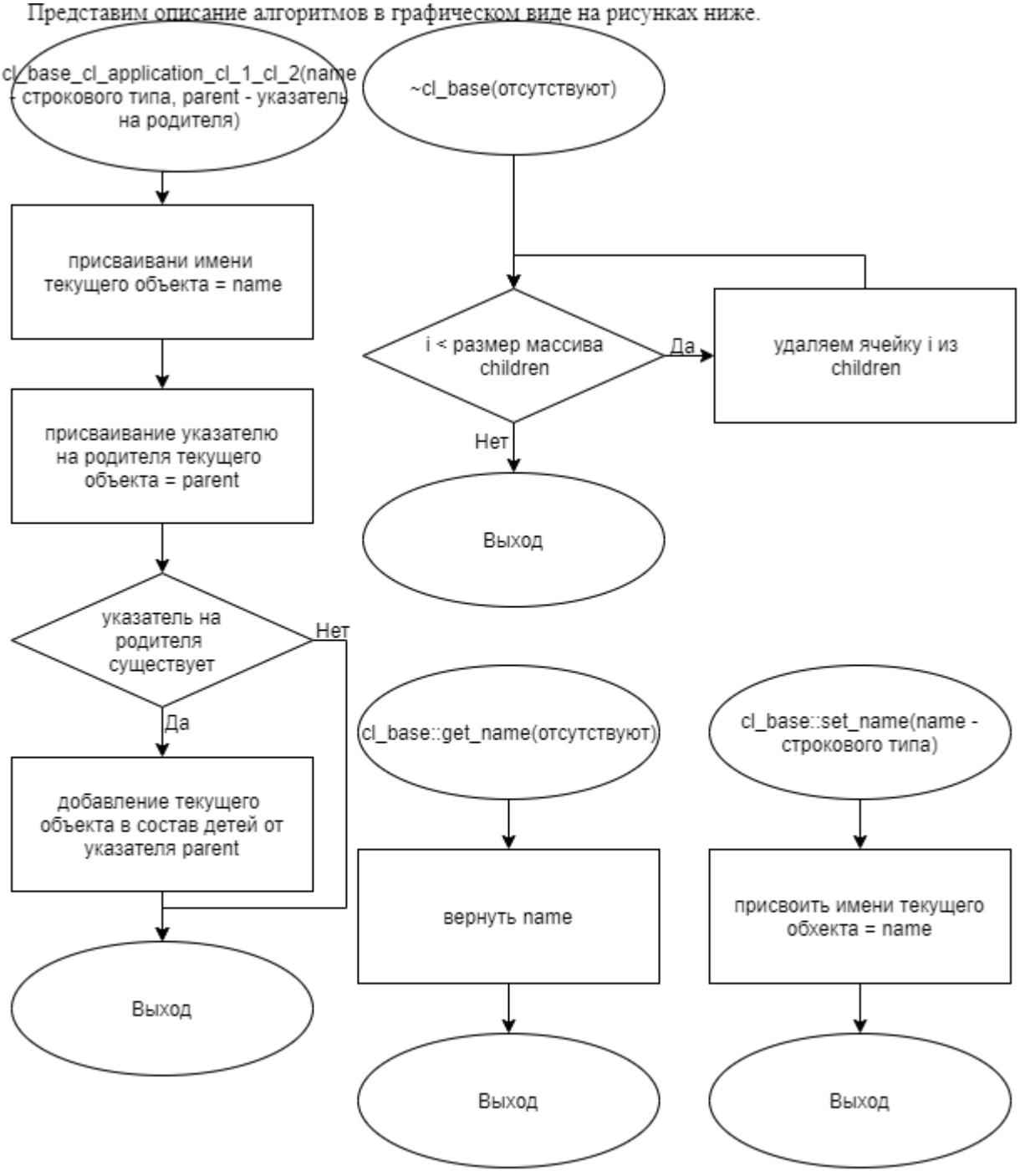

Рис. 1. Блок-схема алгоритма.

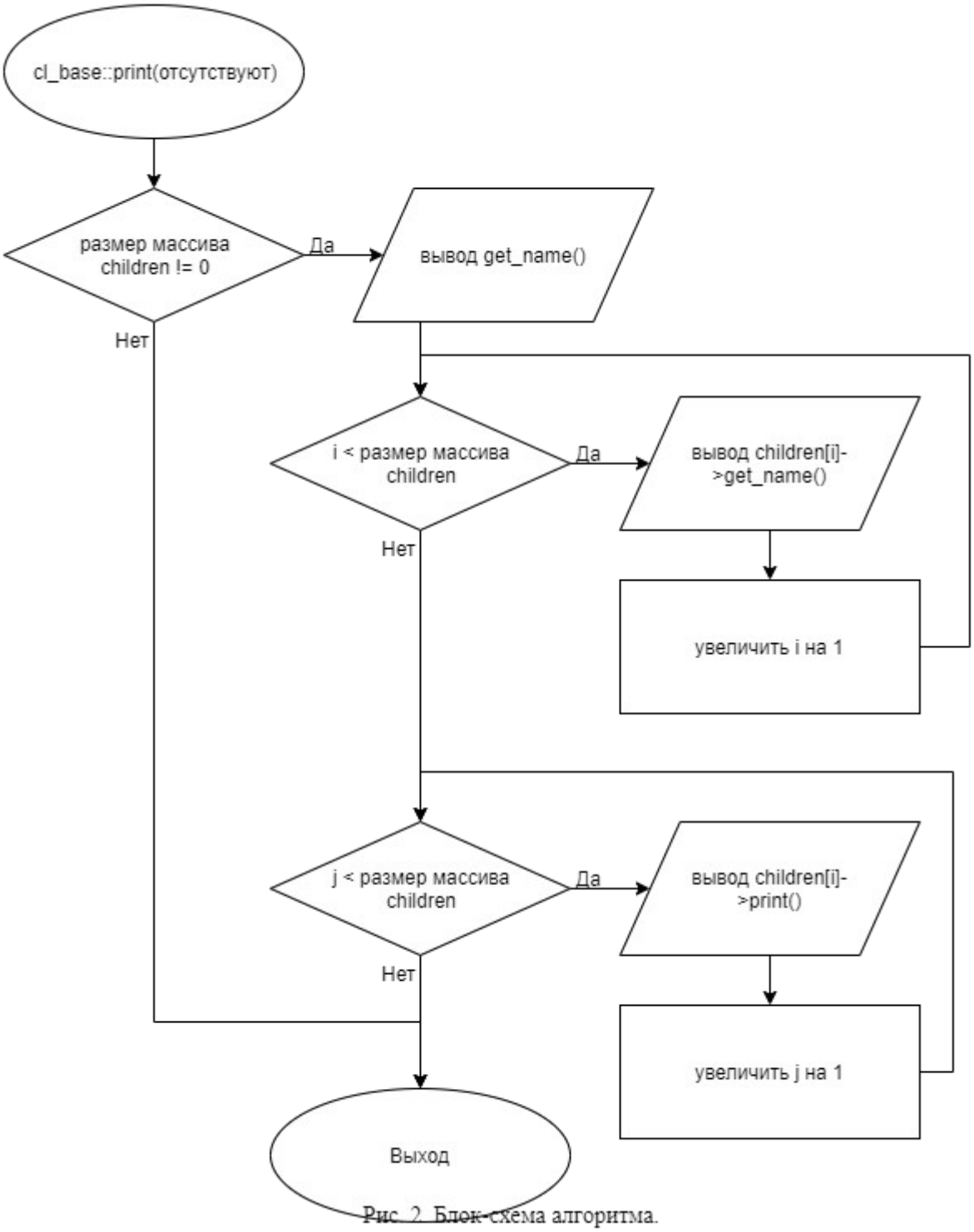

Рис. 2. Блок-схема алгоритма.

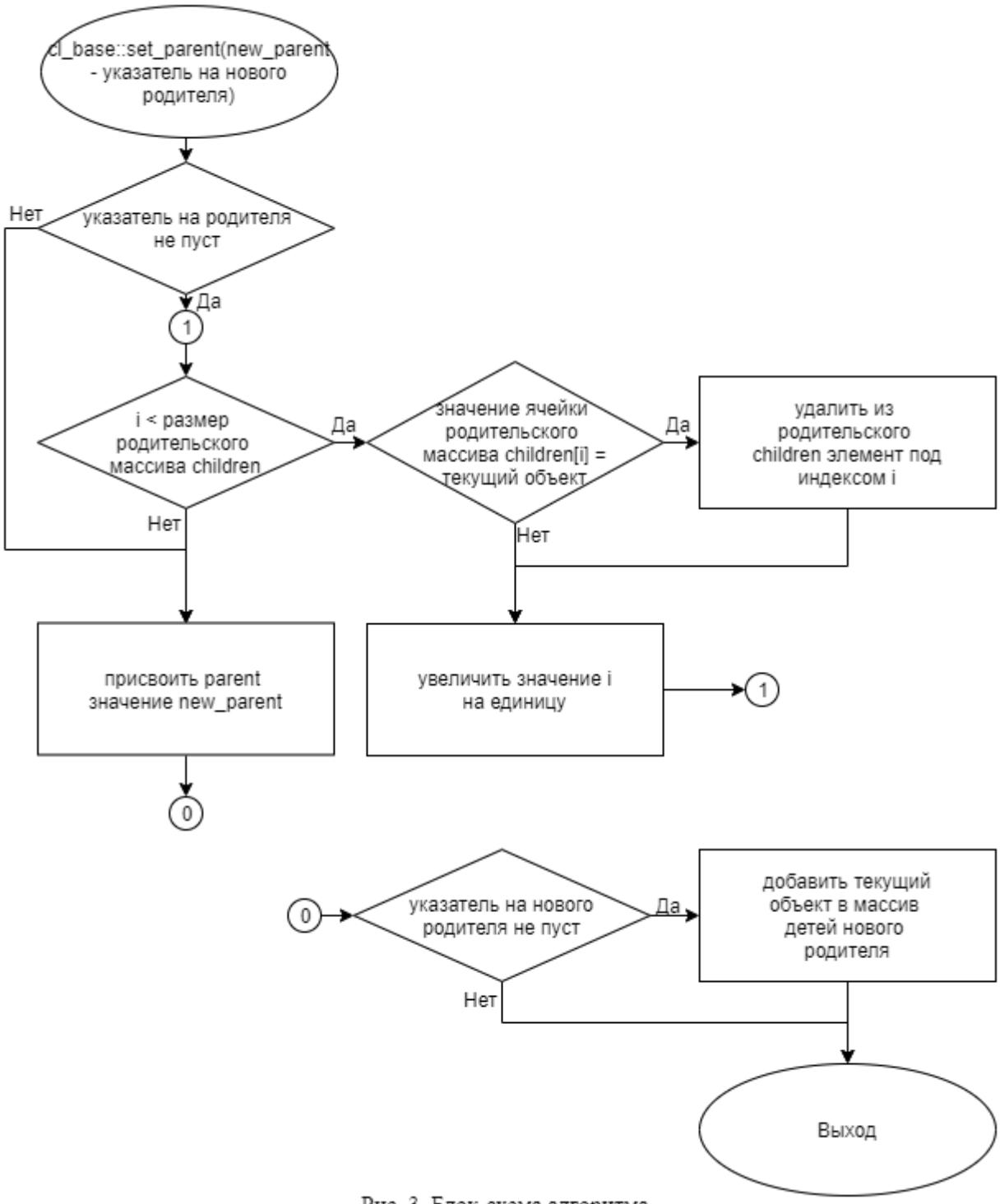

Рис. 3. Блок-схема алгоритма.

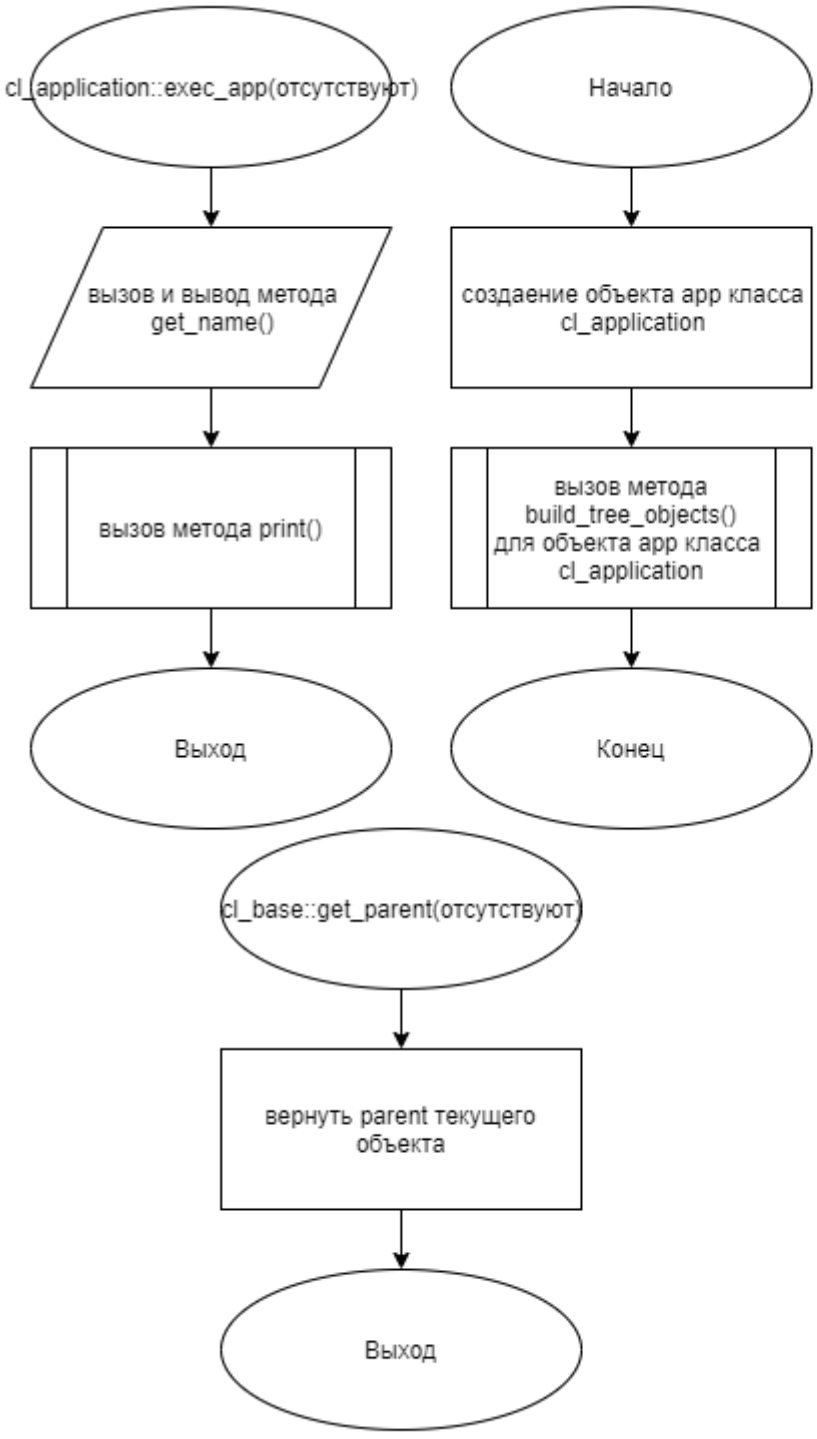

Рис. 4. Блок-схема алгоритма.

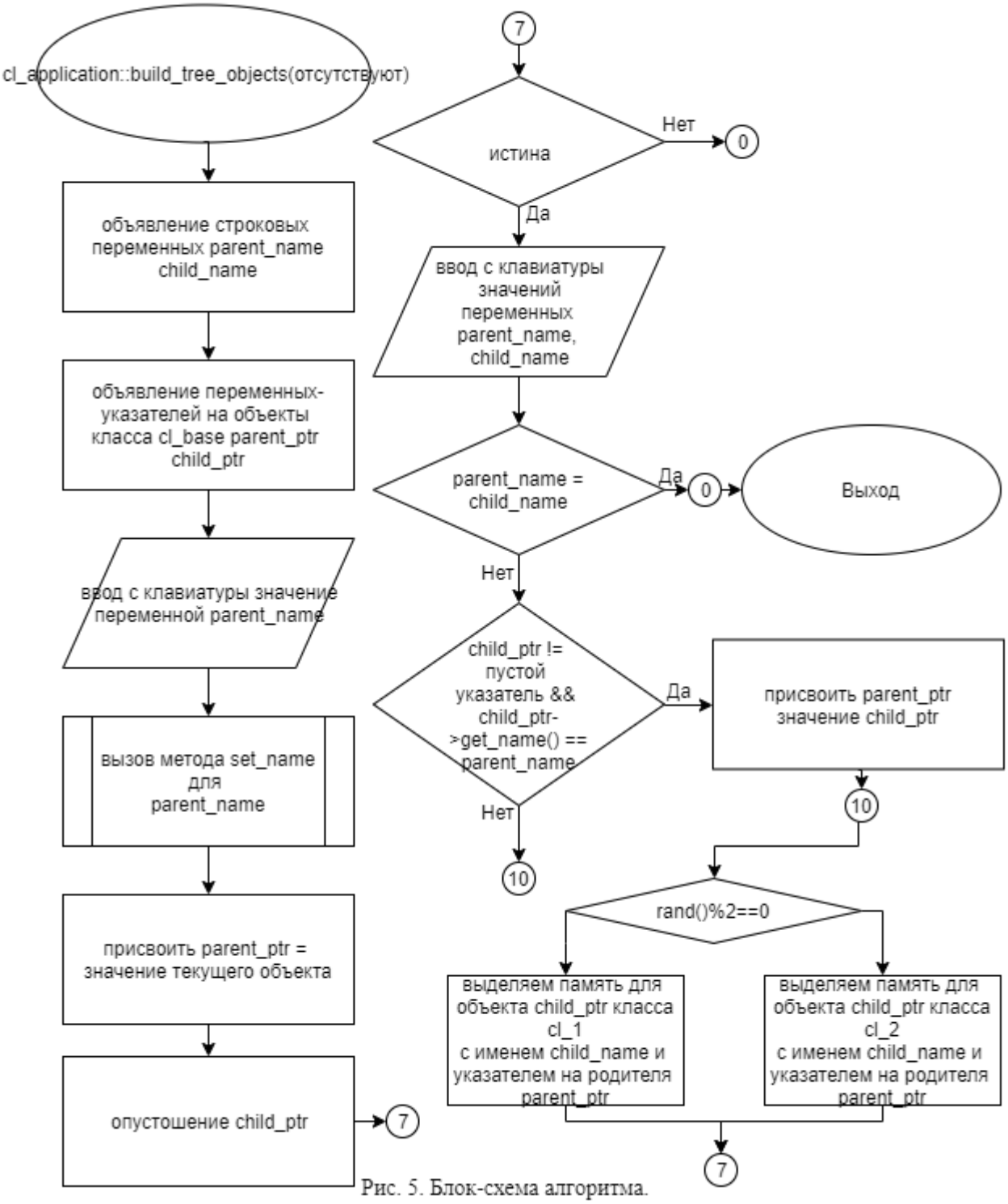

Рис. 5. Блок-схема алгоритма.

Код программы

Программная реализация алгоритмов для решения задачи представлена ниже.

Файл cl_1.h

```
#ifndef __CL_1_H__
#define __CL_1_H__
#include "cl_base.h"
        class cl_1 : public cl_base
{
public:
        cl_1(string name, cl_base* parent) : cl_base(name, parent) {};
};
#endif
```

Файл cl_2.h

```
#ifndef __CL_2_H__
#define __CL_2_H__
#include "cl_base.h"
        class cl_2 : public cl_base
{
public:
        cl_2(string name, cl_base* parent) : cl_base(name, parent) {};
};
#endif
```

Файл cl_application.cpp

```
#include "cl_application.h"
void cl_application::build_tree_objects()
{
        string parent_name, child_name;
        cl_base* parent_ptr, * child_ptr;
        cin >> parent_name;
        set_name(parent_name);
        parent_ptr = this;
        child_ptr = nullptr;
        while (true)
        {
```

```
                cin >> parent_name >> child_name;
                if (parent_name == child_name)
                        break;
                if (child_ptr != nullptr && child_ptr->get_name() ==
parent_name)
                        parent_ptr = child_ptr;
                if (rand()%2==0)
                        child_ptr = new cl_1(child_name, parent_ptr);
                else
                        child_ptr = new cl_2(child_name, parent_ptr);
        }
}
int cl_application::exec_app()
{
        if (get_name() != " " and get_name() != "")
                cout << get_name();
        print();
        return 0;
}
```

Файл cl_application.h

```
#ifndef __CL_APPLICATION_H__
#define __CL_APPLICATION_H__
#include "cl_base.h"
#include "cl_1.h"
#include "cl_2.h"
class cl_application : public cl_base
{
public:
        cl_application(string name = "") : cl_base(name) {};
        void build_tree_objects();
        int exec_app();
};
#endif
```

Файл cl_base.cpp

```
#include "cl_base.h"
cl_base::cl_base(string name, cl_base* parent)
{
        this->name = name;
        this->parent = parent;
        if (parent)
                parent->children.push_back(this);
        return;
}
cl_base::~cl_base()
{
        for (int i = 0; i < children.size(); i++)
                delete children[i];
```

```
}
string cl_base::get_name()
{
        return name;
}
void cl_base::set_name(string& name)
{
        this->name = name;
        return;
}
void cl_base::set_parent(cl_base* new_parent)
{
        if (parent)
                for (int i = 0; i < parent->children.size(); i++)
                        if (parent->children[i] == this)
                                parent->children.erase(parent ->
children.begin() + i);
        parent = new_parent;
        if (new_parent)
                new_parent->children.push_back(this);
        return;
}
cl_base* cl_base::get_parent()
{
        return this->parent;
}
void cl_base::print()
{
        if (children.size())
        {
                if (get_name() != "" and get_name() != " ")
                        cout << endl << get_name();
                for (int i = 0; i < children.size(); i++)
                        if (children[i]->get_name() != "" and children[i] and
                                children[i]->get_name() != " ")
                                cout << "  " << children[i]->get_name();
                for (int i = 0; i < children.size(); i++)
                        children[i]->print();
        }
        return;
}
```

Файл cl_base.h

```
#ifndef __CL_BASE_H__
#define __CL_BASE_H__
#include <iostream>
#include <vector>
#include <string>
using namespace std;
class cl_base
```

```
{
        string name;
        cl_base* parent;
        vector<cl_base*> children;
public:
        cl_base(string name = "", cl_base* parent = nullptr);
        ~cl_base();
        string get_name();
        void set_name(string& name);
        cl_base* get_parent();
        void set_parent(cl_base* new_parent);
        void print();
};
#endif
```

Файл main.cpp

```
#include "cl_application.h"
int main()
{
        cl_application application;
        application.build_tree_objects();
        return application.exec_app();
}
```

Тестирование

Результат тестирования программы представлен в следующей таблице.

Входные данные	Ожидаемые выходные данные	Фактические выходные данные
Object_root Object_root Object_1 Object_root Object_2 Object_root Object_3 Object_3 Object_4 Object_3 Object_5 Object_6 Object_6	Object_root Object_root Object_1 Object_2 Object_3 Object_3 Object_4 Object_5	Object_root Object_root Object_1 Object_2 Object_3 Object_3 Object_4 Object_5
Object_root Object_root Object_1 Object_root Object_2 Object_root Object_3 Object_3 Object_4 Object_3 Object_5 Object_6 Object_6	Object_root Object_root Object_1 Object_2 Object_3 Object_3 Object_4 Object_5	Object_root Object_root Object_1 Object_2 Object_3 Object_3 Object_4 Object_5
Object_root Object_root Object_1 Object_root Object_2 Object_root Object_3 Object_3 Object_4 Object_3 Object_5 Object_6 Object_6	Object_root Object_root Object_1 Object_2 Object_3 Object_3 Object_4 Object_5	Object_root Object_root Object_1 Object_2 Object_3 Object_3 Object_4 Object_5

ЗАКЛЮЧЕНИЕ

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ (ИСТОЧНИКОВ)

1. Васильев А.Н. Объектно-ориентированное программирование на C++. Издательство: Наука и Техника. Санкт-Петербург, 2016г. 543 стр.

2. Шилдт Г. C++: базовый курс. 3-е изд. Пер. с англ.. — М.: Вильямс, 2017. — 624 с.

3. Методическое пособие для проведения 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).

4. 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).

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).