Здесь будет титульник, листай ниже

СОДЕРЖАНИЕ

1 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 (не допускается задание головного объекта для корневого и появление второго корневого объекта);

- 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.

1.1 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

1.2 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

2 МЕТОД РЕШЕНИЯ

В данной программе используется:

cin/cout - объекты потоков ввода и вывода;

Объект root класса Application

Объекты класса Cobject, количество задается пользователем

класс CBase

- поля:
- методы:

класс Application настедован от CBase:

- поля

1. CBase* temp_paren - переменная для временного хранения указателя на головной объект;
2. CBase* temp_child - переменная для временного хранения указателя на подчиненный объект;

- методы:

1. конструктор наследованный от конструктора класса CBase;
2. void buildTee() - метод постройки дерева объектов (базового класса);
3. int execute - метод вывода головного объекта и его дерева наследников (за счет метода базового класса printTree);

класс Cobject

методы:

конструктор наследованный от конструктора класса CBase;

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 функции main

Функционал: основная, функция точка входа.

Параметры: -.

Возвращаемое значение: целое число - код ошибки.

Алгоритм функции представлен в таблице 1.

Таблица 1 – Алгоритм функции main

№	Предикат	Действия	№ перехода
1		создание обькта root класса Application с параметром nullptr	2
2		вызов метода buildTree() для объекта root	3
3		вернуть результат вызова метода execute() для объекта root	∅

3.2 Алгоритм метода buildTree класса Application

Функционал: Построение дерева объектов.

Параметры: -.

Возвращаемое значение: -.

Алгоритм метода представлен в таблице 2.

Таблица 2 – Алгоритм метода buildTree класса Application

№	Предикат	Действия	№ перехода
1		объявление переменных parent_name и child_name	2

№	Предикат	Действия	№ перехода
		строчного типа - имена	
2		ввод значения с клавиатуры в переменною parent_name	3
3		вызов метода setName() с параметром parent_name	4
4		присвоение переменной temp_parent указателя на текущий объект	5
5		ввод значения с клавиатуры в переменные parent_name и child_name	6
6	значение переменной parent_name равно значению переменной child_name	выйти	∅
			7
7	значение переменной parent_name не равно результату вызова метода getName() для поля-указателя temp_parent		8
			9
8	значение переменной parent_name равно результату вызова метода getName() для поля-указателя temp_child	присвоение значения поля-указателя temp_child полю-указателю temp_parent	9
			6
9		присвоение объекта класса Cobject, созданного в динамической памяти, с параметрами temp_parent и child_name полю temp_child	6

3.3 Алгоритм конструктора класса CBase

Функционал: конструктор объектов класса CBase.

Параметры: указатель на CBase parent - головной объект , строка name - имя.

Алгоритм конструктора представлен в таблице 3.

Таблица 3 – Алгоритм конструктора класса CBase

№	Предикат	Действия	№ перехода
1		присвоение значение переменной заданной из параметров parent полю parent - указатель на родителя	2
2		присвоение значение переменной заданной из параметров name полю name - имя объкта	3
3	значенмие поля panen не равно нулевому указателю nullptr	добавить указатель на текущщий объект в конец вектора children родительского объекта поля parent	∅
			∅

3.4 Алгоритм метода setName класса CBase

Функционал: установить имя объекта.

Параметры: строка name - имя объекта.

Возвращаемое значение: -.

Алгоритм метода представлен в таблице 4.

Таблица 4 – Алгоритм метода setName класса CBase

№	Предикат	Действия	№ перехода
1		присвоение значения параметра name полю name	∅

3.5 Алгоритм метода getName класса CBase

Функционал: получить имя объекта.

Параметры: -.

Возвращаемое значение: строка - имя объекта.

Алгоритм метода представлен в таблице 5.

Таблица 5 – Алгоритм метода getName класса CBase

№	Предикат	Действия	№ перехода
1		вернуть значение поля name	∅

3.6 Алгоритм метода setState класса CBase

Функционал: установить состояние объекта.

Параметры: целое число state - состояние объекта.

Возвращаемое значение: -.

Алгоритм метода представлен в таблице 6.

Таблица 6 – Алгоритм метода setState класса CBase

№	Предикат	Действия	№ перехода
1		присвоение значения параметра state полю state	∅

3.7 Алгоритм метода getState класса CBase

Функционал: получить значение состояния объекта.

Параметры: -.

Возвращаемое значение: цулое число - состояние объекта.

Алгоритм метода представлен в таблице 7.

Таблица 7 – Алгоритм метода getState класса CBase

№	Предикат	Действия	№ перехода
1		вернуть значение поля state	∅

3.8 Алгоритм метода setParent класса CBase

Функционал: установить головной объект объекта.

Параметры: указатель на CBase newParent - головной объект.

Возвращаемое значение: -.

Алгоритм метода представлен в таблице 8.

Таблица 8 – Алгоритм метода setParent класса CBase

№	Предикат	Действия	№ перехода
1		инициализация указателя на CBase oldParent равному значению поля parent	2
2	указатель oldParent = nulptr или значение указателя заданного из параметров newParent = nullptr	выход	∅
			3
3		инициализация вектора указателей на CBase oldPatherChildren полем children объекта oldParent	4
4		инициализация переменной i = 0 - счетчик цикла	5
5	i < размер вектора oldPatherChildren		6
			7
6	i элемент вектора oldPatherChildren равен указателю на текущий объект	удалить i-й объет из вектора oldPatherChildren	7

№	Предикат	Действия	№ перехода
		инкремент i	5
7		присвоение значения переменной newParent полю parent текущего объекта	8
8		добавить в конец вектора children объекта, на который указывает parent указатель на текущий объект	∅

3.9 Алгоритм метода printNames класса CBase

Функционал: Вывод наименований объектов в древе.

Параметры: -.

Возвращаемое значение: -.

Алгоритм метода представлен в таблице 9.

Таблица 9 – Алгоритм метода printNames класса CBase

№	Предикат	Действия	№ перехода
1	поле children - пустой вектрор	выход	∅
			2
2		вывод перенос строки и имя текущего объекта	3
3		присвоение указателя на начало поля-вектора children полю children_iterator	4
4	children_iterator не равен указателю на следующее значение после последнего элемента поля-вектора children (конец children)	вывод: два пробела	5
			7
5		вывод результата вызова метода getName() для	6

№	Предикат	Действия	№ перехода
		поля-указателя children_iterator	
6		инкремент поля-указателя children_iterator	7
7		декримент поля-указателя children_iterator	8
8		вызов метода printNames() для поля-указателя children_iterator	∅

3.10 Алгоритм метода execute класса Application

Функционал: Выводит дерево объектов.

Параметры: -.

Возвращаемое значение: целое число - код ошибки для возврата в функцию main.

Алгоритм метода представлен в таблице 10.

Таблица 10 – Алгоритм метода execute класса Application

№	Предикат	Действия	№ перехода
1		вывод результата вызова метода getName()	2
2		вызов метода printNames()	3
3		вернуть 0	∅

3.11 Алгоритм деструктора класса CBase

Функционал: деструктор класса CBase.

Параметры: -.

Алгоритм деструктора представлен в таблице 11.

Таблица 11 – Алгоритм деструктора класса CBase

№	Предикат	Действия	№ перехода
1		инициализация переменной целого типа i = 0	2

№	Предикат	Действия	№ перехода
2	i < размера вектора-поля children	удалить из динамической памяти children[i]	3
			∅
3		инкремент переменной i	2

4 БЛОК-СХЕМЫ АЛГОРИТМОВ

Представим описание алгоритмов в графическом виде на рисунках 1-14.

Рисунок 1 – Блок-схема алгоритма

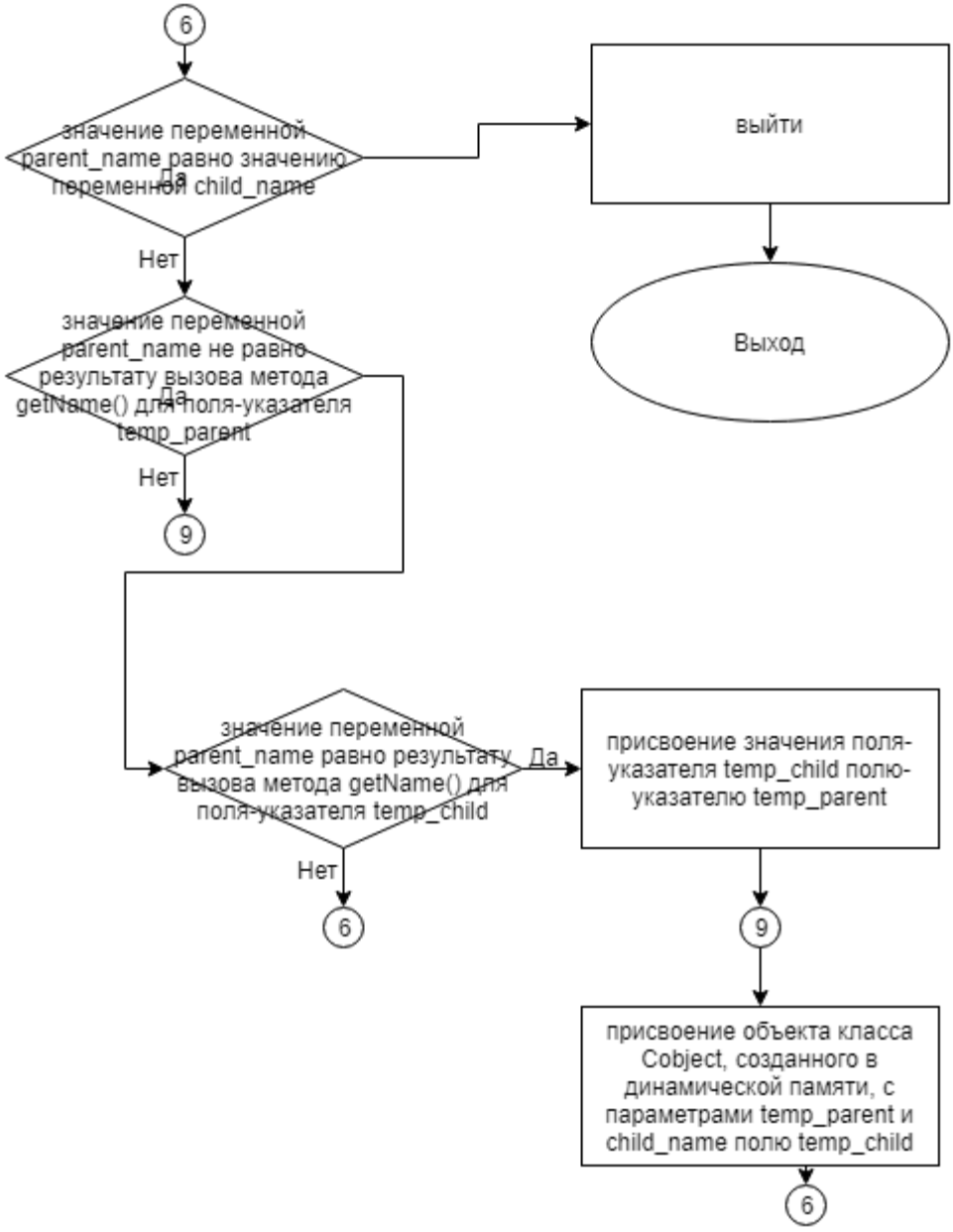

Рисунок 2 – Блок-схема алгоритма

Рисунок 3 – Блок-схема алгоритма

Рисунок 4 – Блок-схема алгоритма

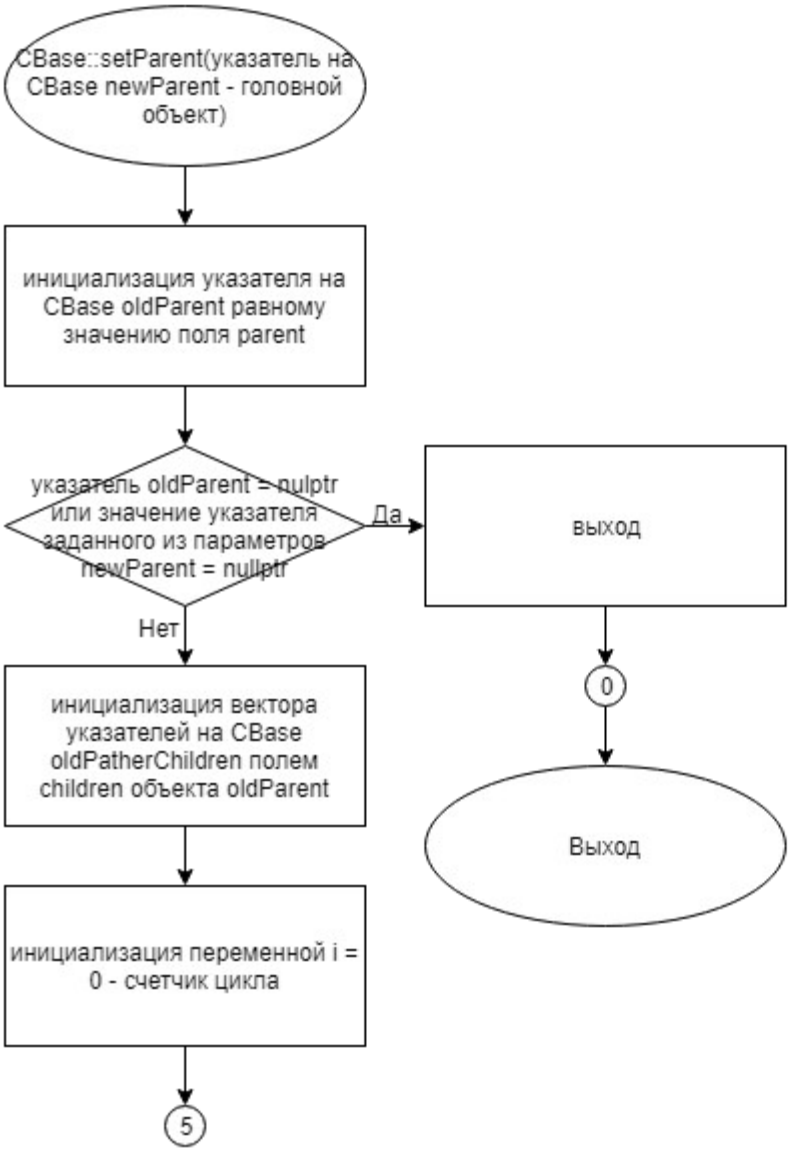

Рисунок 5 – Блок-схема алгоритма

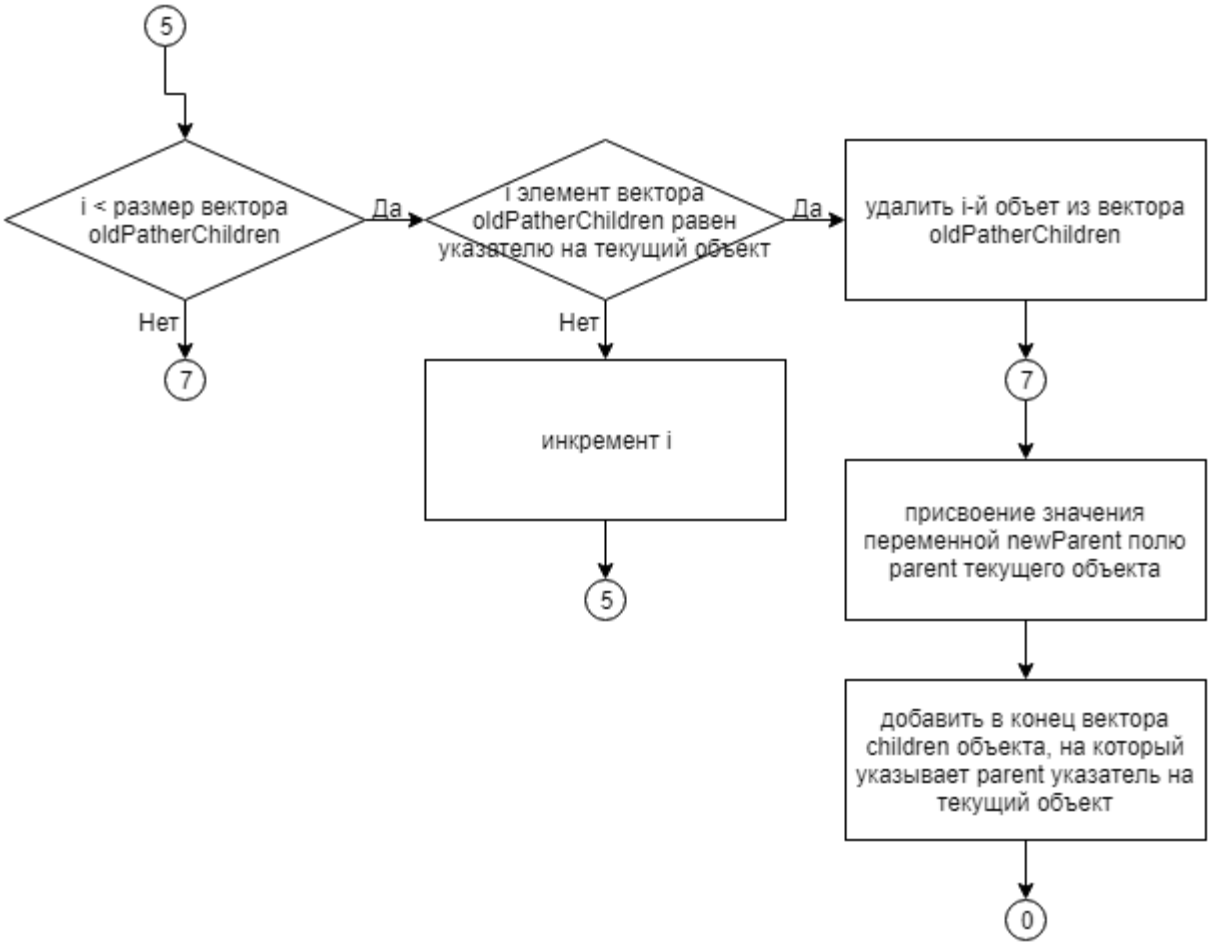

Рисунок 6 – Блок-схема алгоритма

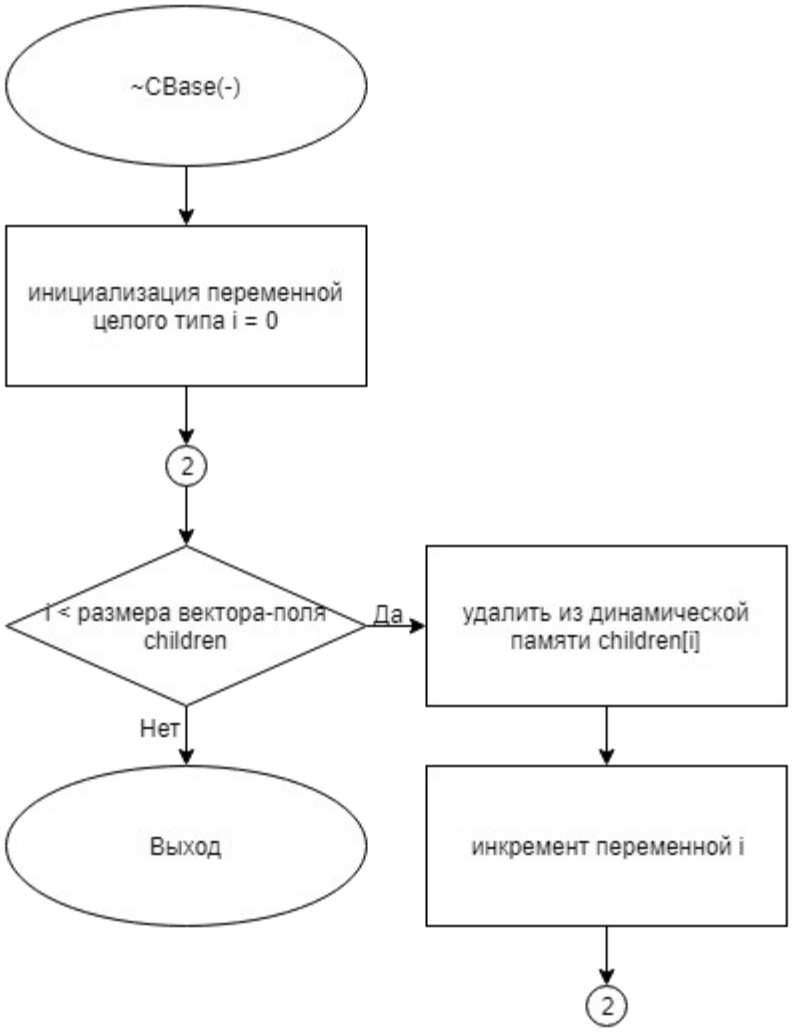

Рисунок 7 – Блок-схема алгоритма

Рисунок 8 – Блок-схема алгоритма

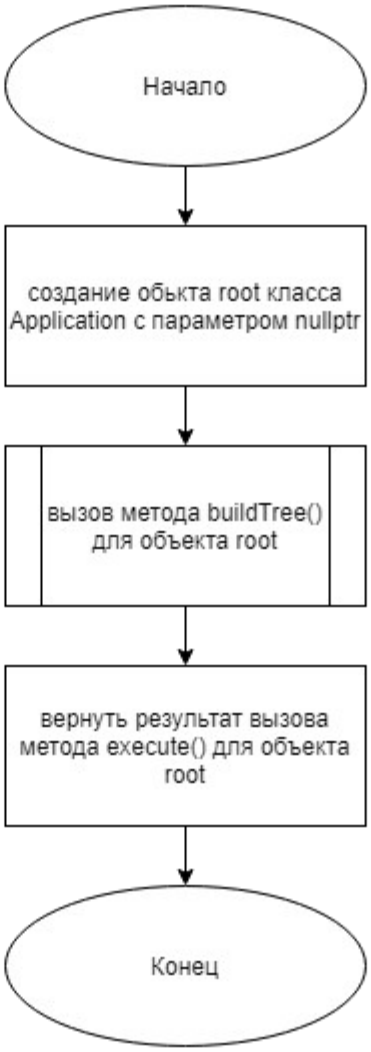

Рисунок 9 – Блок-схема алгоритма

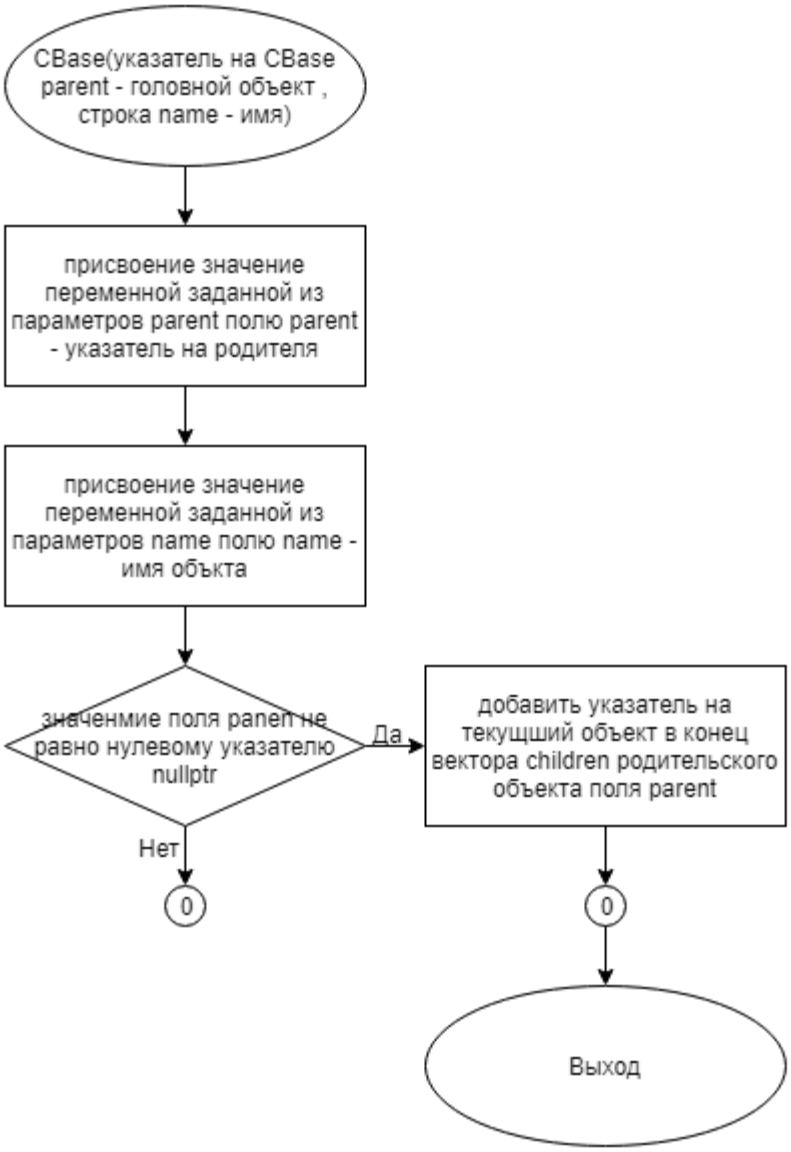

Рисунок 10 – Блок-схема алгоритма

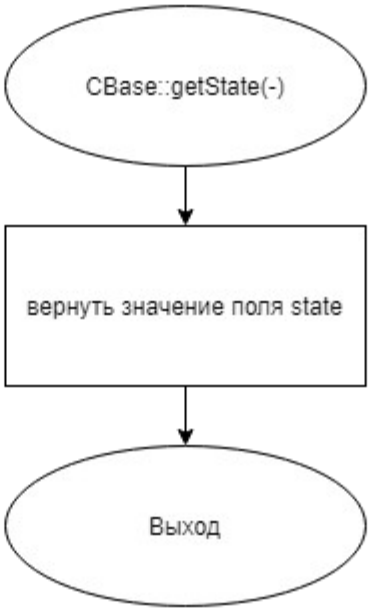

Рисунок 11 – Блок-схема алгоритма

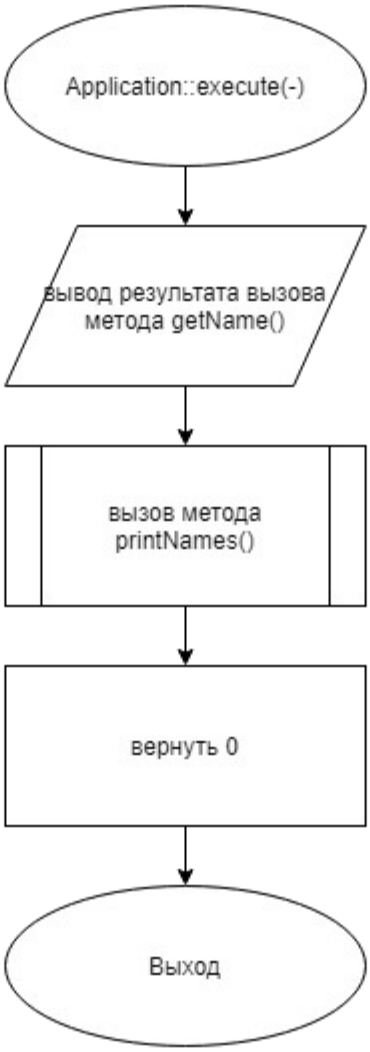

Рисунок 12 – Блок-схема алгоритма

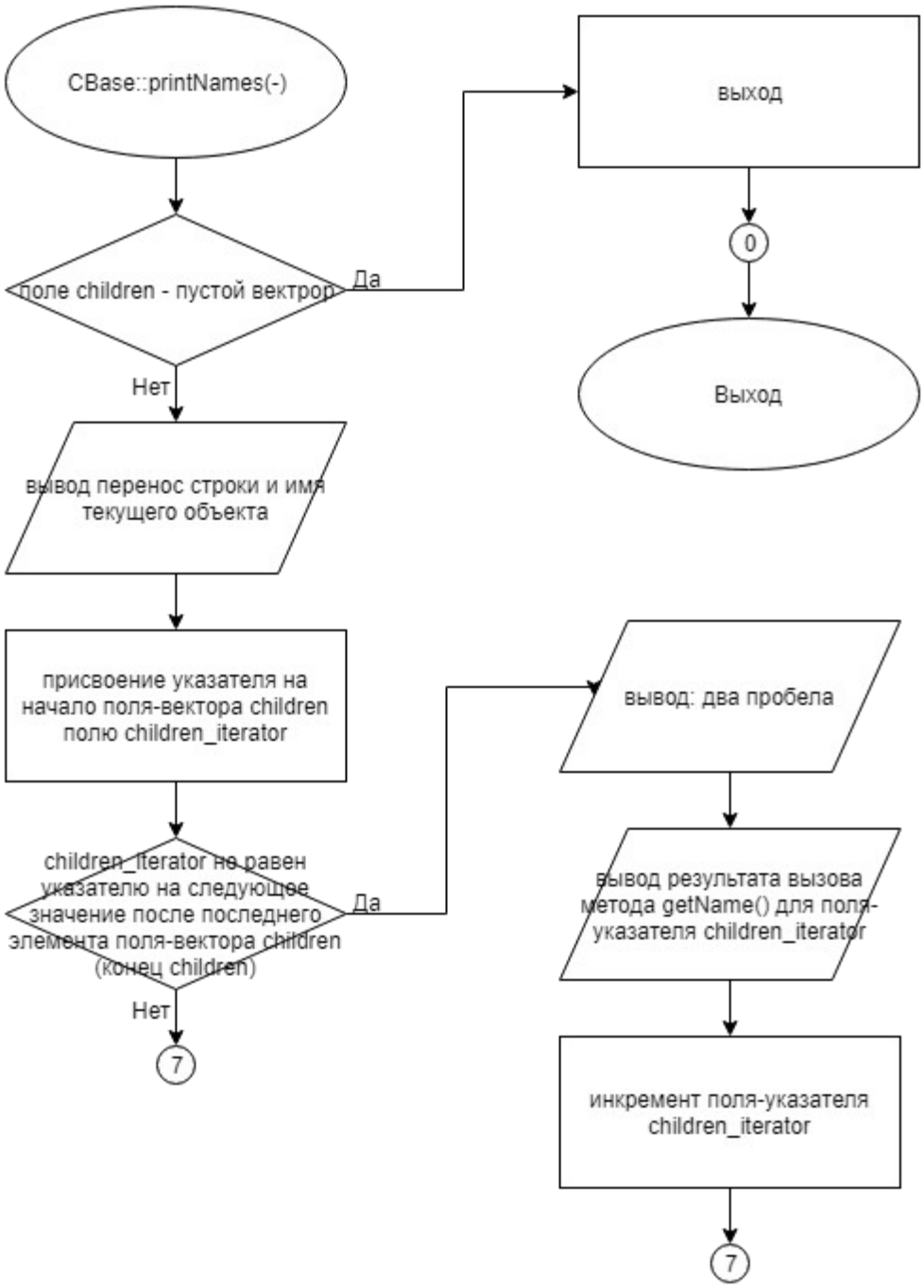

Рисунок 13 – Блок-схема алгоритма

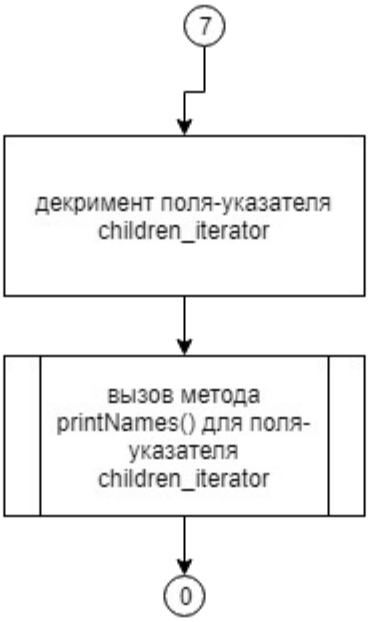

Рисунок 14 – Блок-схема алгоритма

5 КОД ПРОГРАММЫ

Программная реализация алгоритмов для решения задачи представлена ниже.

5.1 Файл application.cpp

Листинг 1 – application.cpp

```
#include "application.h"
#include "cobject.h"
#include <iostream>

void Application::buildTree() {
	std::string parentName, childName;
	std::cin >> parentName;
	setName(parentName);
	tempParent = this;
	do {
	std::cin >> parentName >> childName;
	if(parentName == childName) return;
	if(parentName != tempParent -> getName()) {
		if(parentName == tempChild -> getName())
			tempParent = tempChild;
		else continue;
	}
	tempChild = new Cobject(tempParent, childName);
	}
	while(true);
}

int Application::execute(){
	std::cout << getName();
	printNames();
	return 0;
}
```

5.2 Файл application.h

Листинг 2 – application.h

```
#ifndef APPLICATION_H
#define APPLICATION_H

#include "cbase.h"
```

```
class Application: public CBase {
	CBase* tempParent;
	CBase* tempChild;
public:
	using CBase::CBase; // наследование конструктора
	void buildTree();
	int execute();
};

#endif // APPLICATION_H
```

5.3 Файл cbase.cpp

Листинг 3 – cbase.cpp

```
#include "cbase.h"
#include <iostream>

CBase::CBase(CBase* parent, std::string name) {
	this->parent = parent;
	this->name = name;
	if(parent != nullptr)
		this->parent->children.push_back(this);
}

void CBase::setName(std::string name) {
	this->name = name;
}

std::string CBase::getName() {
	return this->name;
}

void CBase::setState(int state) {
	this->state = state;
}

int CBase::getState() {
	return this->state;
}

void CBase::setParent(CBase* newParent) {
	CBase* oldParent = this->parent;
	if (oldParent == nullptr || newParent == nullptr) return;
	std::vector<CBase*> oldPatherChildren = oldParent->children;
	for (size_t i=0; i < oldPatherChildren.size(); ++i)
		if(oldPatherChildren[i] == this) {
			oldPatherChildren.erase(oldPatherChildren.begin()+i);
			break;
		}
	this->parent = newParent;
	parent->children.push_back(this);
}
```

```
void CBase::printNames() {
	if(children.empty()) return;
	std::cout << "\n" << name;
	childrenIterator = children.begin();
	while(childrenIterator != children.end()) {
	std::cout << "  " << (*(*childrenIterator)).getName();
	childrenIterator++;
	}
	childrenIterator--;
	(*childrenIterator)->printNames();
}

CBase::~CBase() {
	for(size_t i=0; i < children.size(); ++i)
		delete children[i];
}
```

5.4 Файл cbase.h

Листинг 4 – cbase.h

```
#ifndef CBASE_H
#define CBASE_H

#include <string>
#include <vector>

class CBase {
private:
	std::string name;
	CBase* parent;
	int state;
	std::vector<CBase*> children;
	std::vector <CBase*>::iterator childrenIterator;
public:
	CBase(CBase* parent, std::string name="Base");
	void setName(std::string name);
	std::string getName();
	void setState(int state);
	int getState();
	void setParent(CBase* parent);
	CBase* getParent();
	void printNames();
	~CBase();
};

#endif // CBASE_H
```

5.5 Файл cobject.cpp

Листинг 5 – cobject.cpp

```
#include "cobject.h"
```

5.6 Файл cobject.h

Листинг 6 – cobject.h

```
#ifndef COBJECT_H
#define COBJECT_H

#include "cbase.h"

class Cobject: public CBase {
public:
	using CBase::CBase; // наследование конструктора
};

#endif // COBJECT_H
```

5.7 Файл main.cpp

Листинг 7 – main.cpp

```
#include "application.h"


int main() {
	Application root(nullptr);
	root.buildTree();
	return root.execute();
}
```

6 ТЕСТИРОВАНИЕ

Результат тестирования программы представлен в таблице 12.

Таблица 12 – Результат тестирования программы

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
A A B A C C D C V V V	A A B C C D V	A A B C C D V

СПИСОК ИСПОЛЬЗОВАННЫХ 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_laboratorny h_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).