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-13-21	Дамарад Д.В.
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 поставленной задачи используются:

1. Объекты стандартных потоков (cin и cout). Используются для ввода с клавиатуры и вывода на экран.
2. Объект app класса App.
3. Библиотека контейнеров vector для создания вектора, который будет хранить в себе указатели на дочерние объекта.
4. Библиотека string для создания переменных строкового типа.

Класс Base:

- Поля:
    - Поле, хранящее наименование объекта:
        - Наименование - name;
        - Тип - строковый;
        - Модификатор доступа - protected.
    - Поле, хранящее указатель на головной объект:
        - Наименование - head;
        - Тип - указатель на класс Base;
        - Модификатор доступа - protected.
    - Поле, хранящее массив указателей на объекты, подчиненные к текущему объекту в дереве иерархии:
        - Наименование - children;
        - Тип - контейнер с указателями на класс Base;
        - Модификатор доступа - public.
- Методы:
    - Конструктор Base:
        - Функционал - параметризированный конструктор.

- Метод SetName:
    - Функционал - метод определения имени объекта.
- Метод Get Name:
    - Функционал - метод получения имени объекта.
- Метод SetHead :
    - Функционал - метод переопределения головного объекта для текущего в дереве иерархии.
- Метод GetHead :
    - Функционал - метод получения указателя на головной объект текущего объекта.
- Метод PrintTree :
    - Функционал - метод вывода наименований объектов в дереве иерархии.

Класс ObjectTree:

- Поля:
    - Отсутсвуют.
- Методы:
    - Отсутсвуют.

Класс App:

- Поля:
    - Отсутсвуют.
- Методы:
    - Метод App:

        - Функционал - делегирующий конструктор.
    - Метод BuildTree:
        - Функционал - метод построения исходного дерева иерархии объектов.
    - Метод exec:
        - Функционал - метод запуска приложения.

№	Наименование класса	Классы наследники	Модификатор доступа при наследовании	Описание	Номер	Комментарий
1	Base	App	public	Базовый класс в дереве иерархии	2	
		ObjectTree	public		3	
2	App			Класс объектов, подчиненных классу Base		
3	ObjectTree			Класс объектов, подчиненных классу Base		

Описание алгоритма

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

Функция: main

Функционал: Основной алгоритм программы

Параметры: Отсутсвуют

Возвращаемое значение: Целочисленное значение - код возврата

Алгоритм функции представлен в таблице 2.

Таблица 2. Алгоритм функции main

№	Предикат	Действия	№ перехода	Комментарий
1		Инициализация главного объекта app типа App и передача в его конструктор указателя(nullptr)	2	
2		Вызов метода BuildTree у объекта app	3	
3		Возврат значения метода exec у объекта app	∅	

Конструктор класса: Base

Модификатор доступа: public

Функционал: Параметризированный конструктор

Параметры: Указатель на объект класса Base - родительский объект, строка -

название текущего объекта

Алгоритм конструктора представлен в таблице 3.

Таблица 3. Алгоритм конструктора класса Base

№	Предикат	Действия	№ перехода	Комментарий
1		Вызов метода SetName, передача параметра name в качестве аргумента	2	
2		Вызов метода SetHead, передача параметра head в качестве аргумента	3	
3	головной указатель на объект не равен 0	Вызов метода push_back у родительского объекта поля children и передача ему указателя на текущий объект	∅	
			∅	

Класс объекта: Base

Модификатор доступа: public

Метод: SetName

Функционал: Метод определения имени объекта

Параметры: строка name - название объекта

Возвращаемое значение: Отсутсвует

Алгоритм метода представлен в таблице 4.

Таблица 4. Алгоритм метода SetName класса Base

№	Предикат	Действия	№ перехода	Комментарий
1		Присвоение полю name объекта значения передаваемого	∅	

		параметра name		

Класс объекта: Base

Модификатор доступа: public

Метод: GetName

Функционал: Метод получения имени объекта

Параметры: Отсутсвуют

Возвращаемое значение: Строка - поле name

Алгоритм метода представлен в таблице 5.

Таблица 5. Алгоритм метода GetName класса Base

№	Предикат	Действия	№ перехода	Комментарий
1		Возврат поля name объекта	∅	

Класс объекта: Base

Модификатор доступа: public

Метод: SetHead

Функционал: Метод переопределения головного объекта для текущего в дереве иерархии;

Параметры: Указатель на объект класса Base

Возвращаемое значение: Отсутсвует

Алгоритм метода представлен в таблице 6.

Таблица 6. Алгоритм метода SetHead класса Base

№	Предикат	Действия	№ перехода	Комментарий
1		Присвоение полю head объекта значения передаваемого параметра head	∅	

Класс объекта: Base

Модификатор доступа: public

Метод: GetHead

Функционал: Метод получения указателя на головной объект текущего объекта.

Параметры: Отсутсвуют

Возвращаемое значение: Указатель на объект типа Base

Алгоритм метода представлен в таблице 7.

Таблица 7. Алгоритм метода GetHead класса Base

№	Предикат	Действия	№ перехода	Комментарий
1		Возврат поля head объекта	∅	

Класс объекта: Base

Модификатор доступа: public

Метод: PrintTree

Функционал: Метод вывода наименований объектов в дереве иерархии слева направо и сверху вниз;

Параметры: Отсутсвуют

Возвращаемое значение: Отсутсвует

Алгоритм метода представлен в таблице 8.

Таблица 8. Алгоритм метода PrintTree класса Base

№	Предикат	Действия	№ перехода	Комментарий
1	Цикл со счетчиком i		2	
			8	
2	i равно 0		3	
			5	
3	Указатель на головной объект не 0	Переход на новую строку	4	
			4	
4		Вывод имени объекта и вывод пробела	5	
5		Вывод имени очередного дочернего объекта	6	
6	i+1 меньще размера вектора	Вывод пробела	7	
			7	
7	i меньше размера вектора объекта	Увеличение i на 1	1	
			8	
8	Цикл со счетчиком i		9	
			∅	
9	Дочерний объект данного объекта имеет свои дочерние объекта	Вызов метода PrintTree у данного дочернего объекта	10	
			10	
10	i меньше размера вектора объекта	Увеличение i на 1	8	
			∅	

Класс объекта: App

Модификатор доступа: public

Метод: BuildTree

Функционал: Построение дерева объектов с помощью потока ввода в консоли

Параметры: Отсутсвуют

Возвращаемое значение: Отсутсвует

Алгоритм метода представлен в таблице 9.

Таблица 9. Алгоритм метода BuildTree класса App

№	Предикат	Действия	№ перехода	Комментарий
1		Объявление строковых переменных a и b	2	
2		Считывание значения a с клавиатуры	3	
3		Вызов у данного объекта метода SetName с параметром a	4	
4	Бесконечный цикл	Считывание с клавиатуры значений переменных a и b	5	
			5	
5	Имя класс предка не равно имени класса наследника		6	a!=b
			10	
6	Имя базового класса совпадает с именем вводимого класса предка	Создание объекта класса ObjectTree с параметрами this(указатель на главный объект) и b	4	
			7	
7		Инициализация указателя на объект	8	

		класса Base		
8	Имя объекта по указателю не равняется вводимому имени	Присваивание указателю ссылки на последний дочерний элемент объекта	8	
			9	
9		Создание объекта класса ObjectTree с параметрами temp,b	4	
10		Завершение цикла	∅	

Класс объекта: App

Модификатор доступа: public

Метод: exec

Функционал: Главный метод вывода построенного дерева в консоль

Параметры: Отсутсвуют

Возвращаемое значение: Целочисленное значение - код возврата

Алгоритм метода представлен в таблице 10.

Таблица 10. Алгоритм метода exec класса App

№	Предикат	Действия	№ перехода	Комментарий
1		Вывод имени корневого объекта и переход на новую строчку	2	
2		Вызов метода PrintTree	3	
3		Возврат 0	∅	

Конструктор класса: App

Модификатор доступа: public

Функционал: Делегирующий конструктор

Параметры: Указатель на объект класса Base - родительский объект, строка - название текущего объекта

Алгоритм конструктора представлен в таблице 11.

Таблица 11. Алгоритм конструктора класса App

№	Предикат	Действия	№ перехода	Комментарий
1		Передача параметров head, name в конструктор Base	∅	

Блок-схема алгоритма

Представим описание алгоритмов в графическом виде на рисунках ниже.

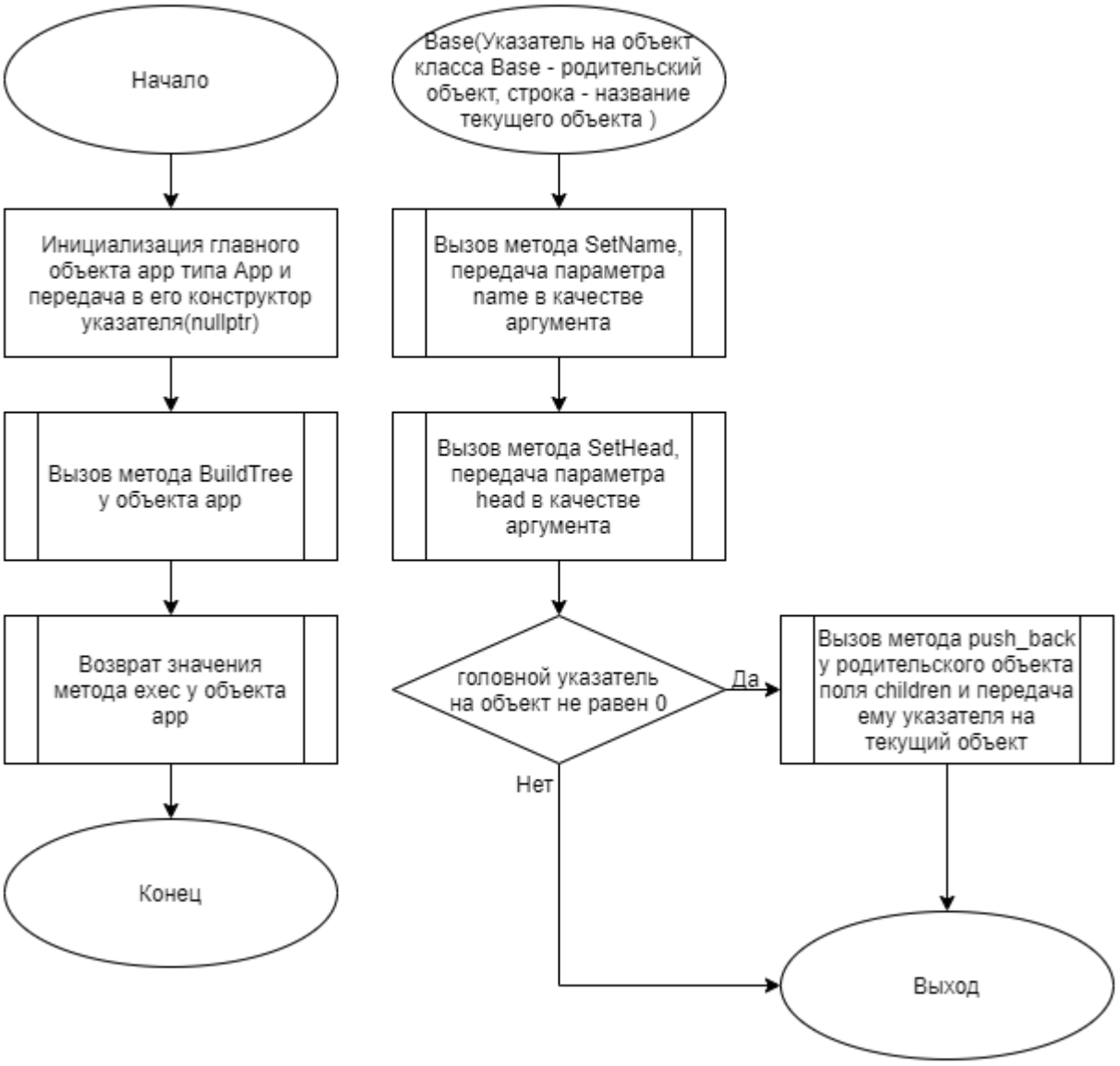

Рис. 1. Блок-схема алгоритма.

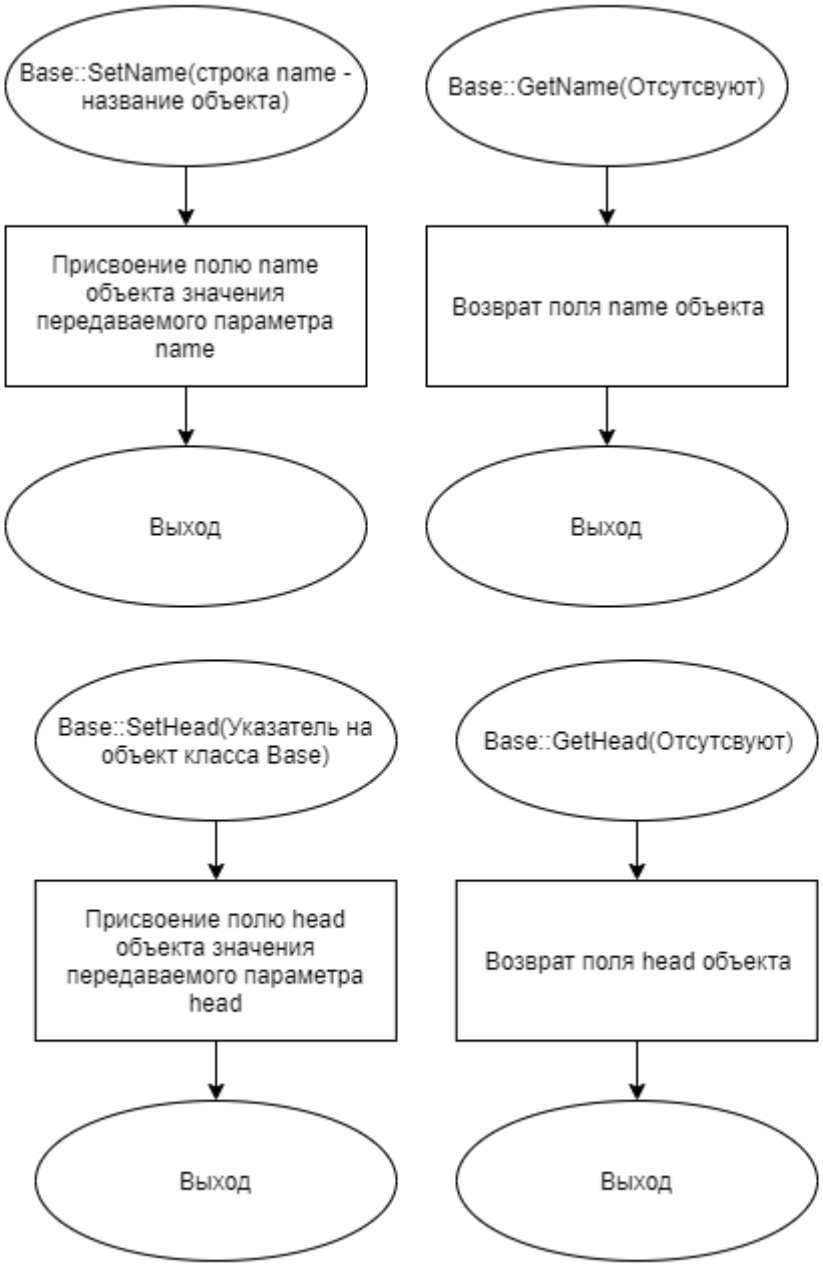

Рис. 2. Блок-схема алгоритма.

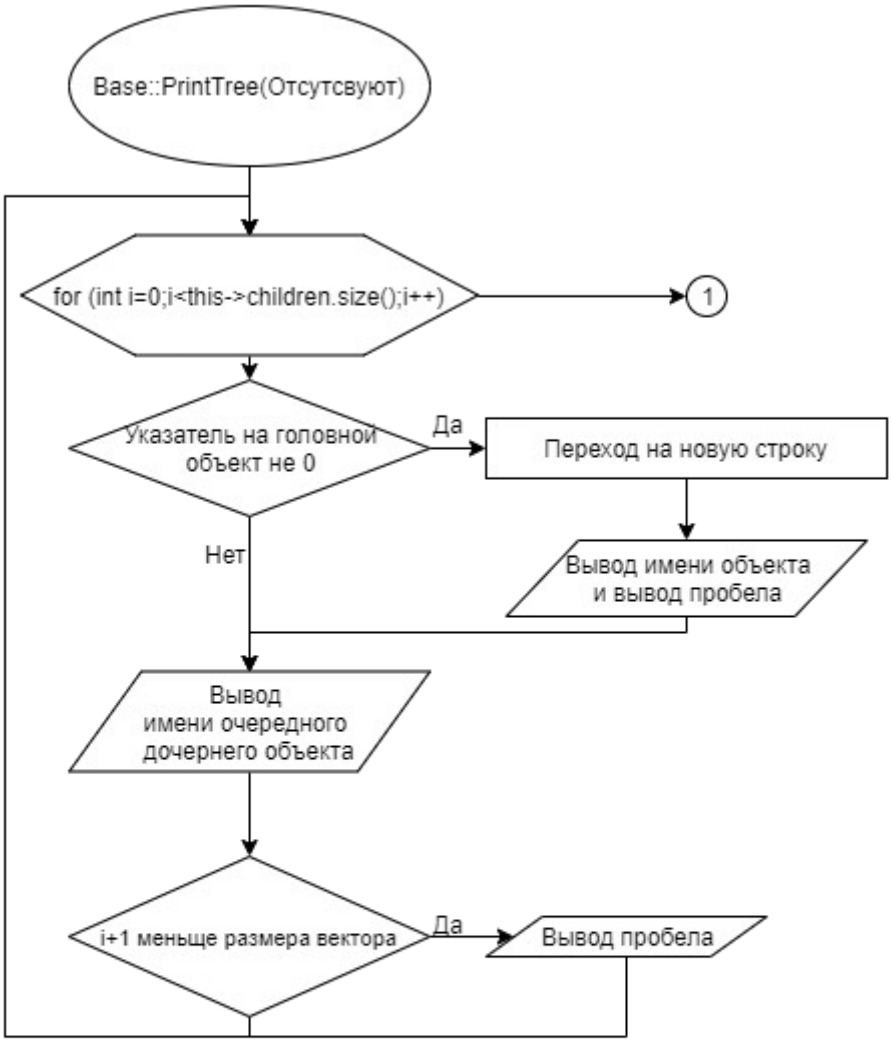

Рис. 3. Блок-схема алгоритма.

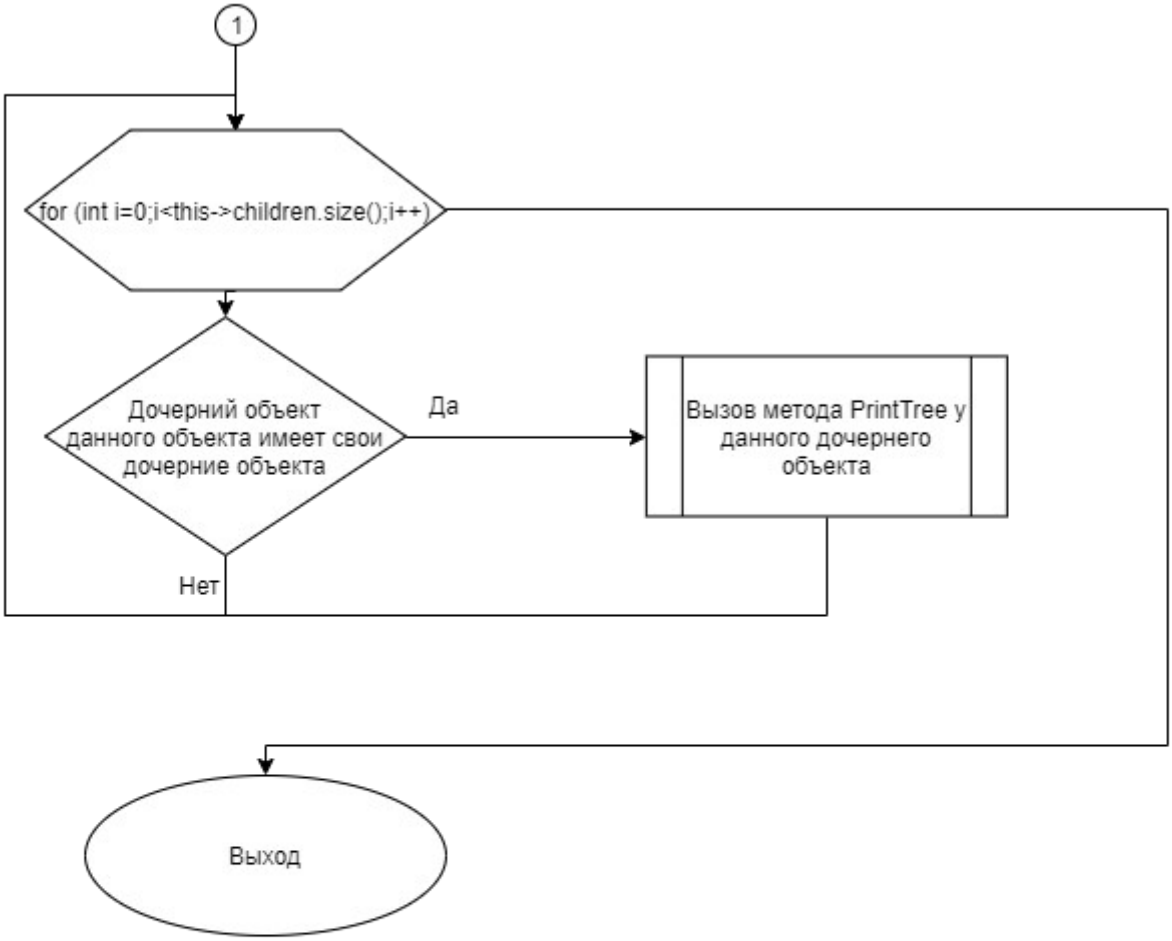

Рис. 4. Блок-схема алгоритма.

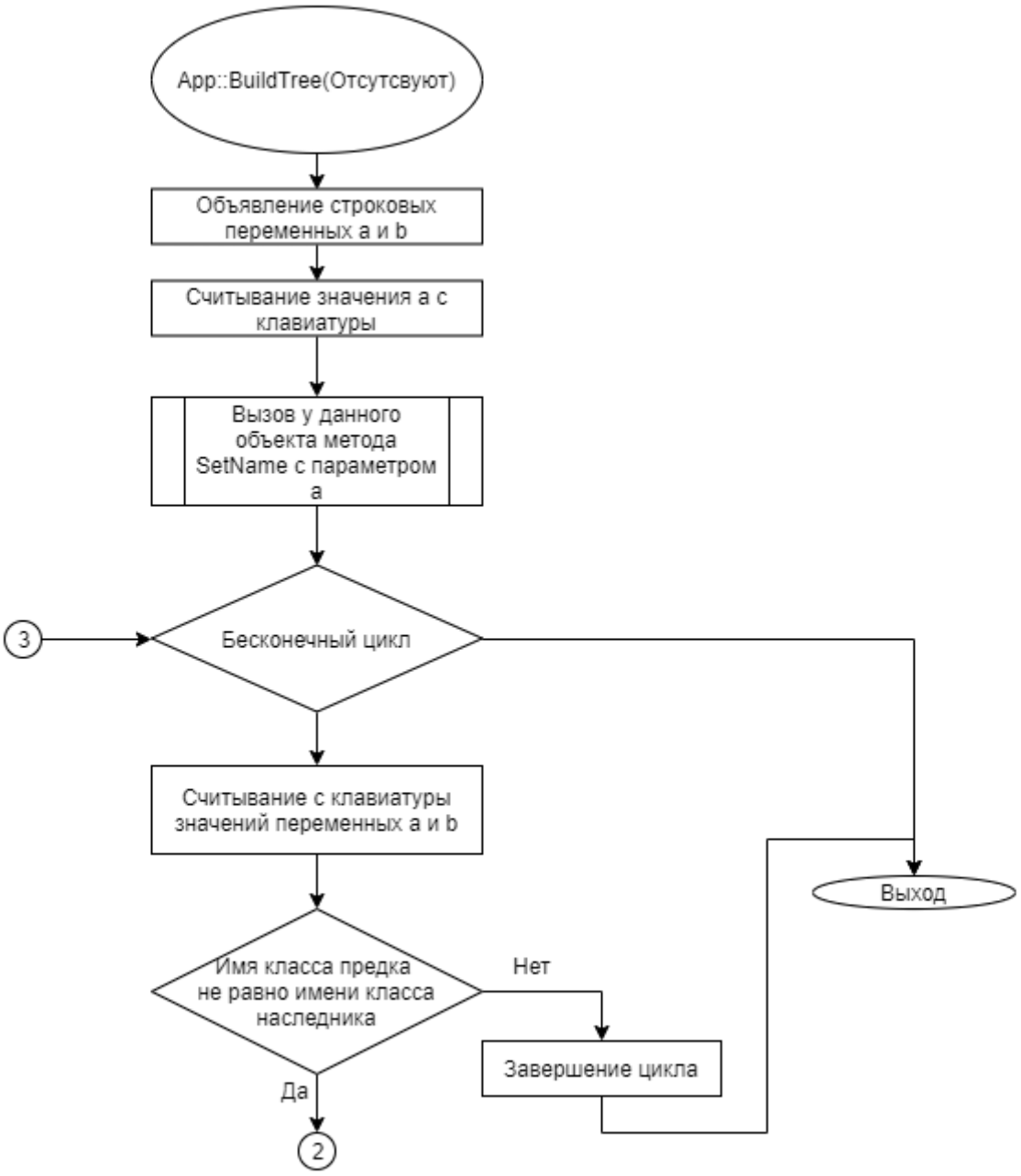

Рис. 5. Блок-схема алгоритма.

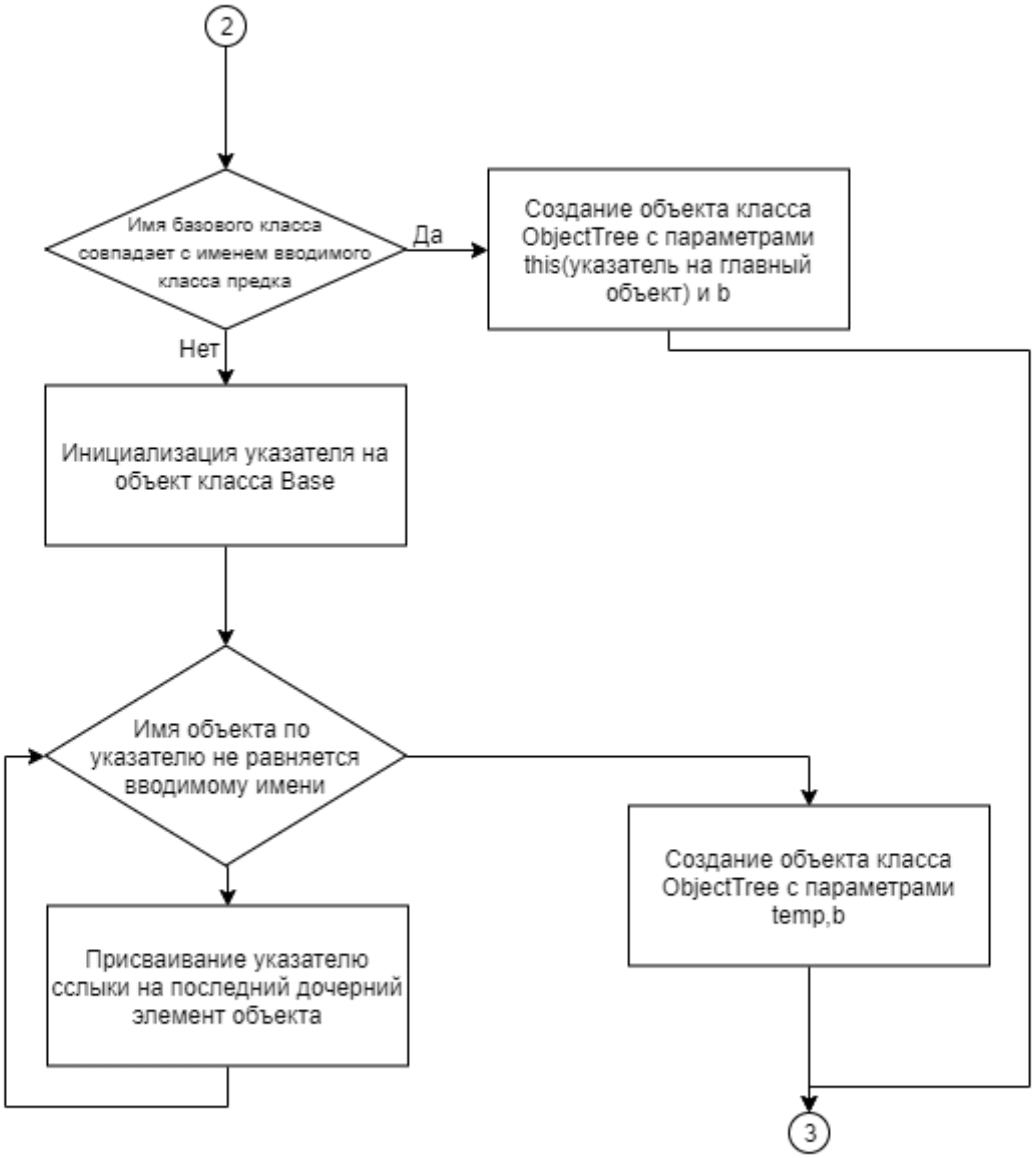

Рис. 6. Блок-схема алгоритма.

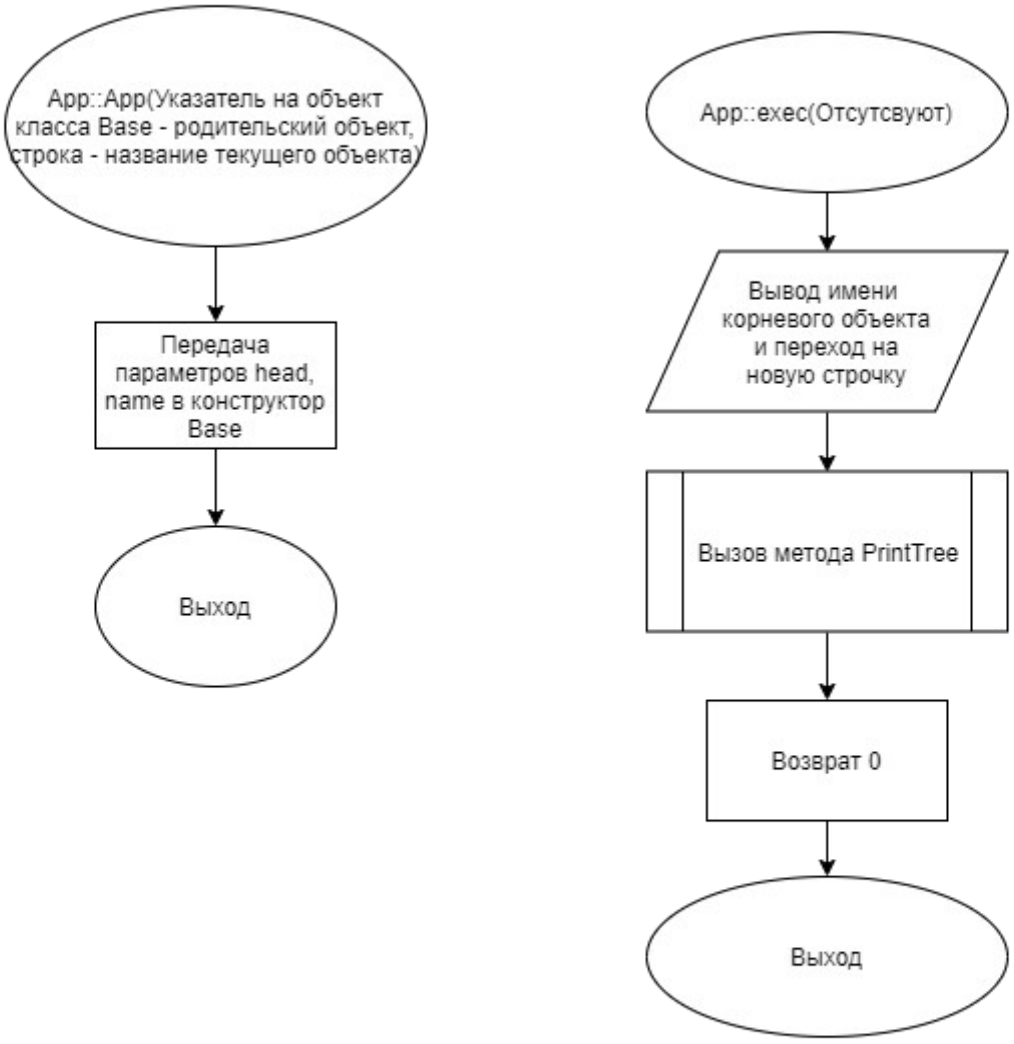

Рис. 7. Блок-схема алгоритма.

Код программы

Программная реализация алгоритмов для решения задачи представлена ниже.

Файл App.cpp

```
#include "App.h"
#include "ObjectTree.h"
#include <string>
#include <iostream>
using namespace std;
int App::exec(){
        cout<<this->GetName()<<endl;
        PrintTree();
        return 0;
}
void App::BuildTree(){
        string a,b;
        cin>>a;
        this->SetName(a);
        while(true){
                cin>>a>>b;
                if (a==b){
                        break;
                }
                else if(this->GetName()==a){
                        new ObjectTree(this,b);
                }
                else {
                        Base* temp=(this->children[children.size()-1]);
                        while (temp->GetName()!=a){
                                temp=temp->children[temp->children.size()-1];
                        }
                        new ObjectTree(temp,b);
                }
        }
}
```

Файл App.h

```
#ifndef APP_H
#define APP_H
#include "Base.h"
class App : public Base{
public:
        App(Base* head, string name=""):Base(head, name)
        {};
        void BuildTree();
        int exec();
```

```
};
#endif
```

Файл Base.cpp

```
#include "Base.h"
#include <iostream>
using namespace std;
Base::Base(Base* head, string name){
        SetName(name);
        SetHead(head);
        if (head!=nullptr){
                head->children.push_back(this);
        }
}
void Base::SetName(string name){
        this->name=name;
}
string Base::GetName(){
        return name;
}
void Base::SetHead(Base* head){
        this->head=head;
}
Base* Base::GetHead(){
        return head;
}
void Base::PrintTree(){
        for(int i=0;i<this->children.size();i++){
                if (i==0){
                        if (this->GetHead()!=nullptr){
                                cout<<endl;
                        }
                        cout<<this->GetName()<<"  ";
                }
                cout<<children[i]->GetName();
                if (i+1<children.size()){
                        cout<<"  ";
                }
        }
        for(int i=0;i<this->children.size();i++){
                if (children[i]->children.size()>0){
                        children[i]->PrintTree();
                }
        }
}
```

Файл Base.h

```
#ifndef BASE_H
#define BASE_H
#include <iostream>
#include <string>
#include <vector>
using namespace std;
class Base{
protected:
        string name;
        Base* head;
public:
        Base(Base* head, string name="");
        void SetName(string name); //метод определения имени объекта
        string GetName(); //метод получения имени объекта
        void PrintTree(); //метод вывода наименований объектов в дереве
иерархии
        void SetHead(Base* head); //метод переопределения головного объекта
для текущего в дереве иерархии
        Base* GetHead(); //метод получения указателя на головной объект
текущего объекта
        vector<Base*>children; //массив указателей на объекты, подчиненные к
текущему объекту в дереве иерархии
};
#endif
```

Файл main.cpp

```
#include "App.h"
int main(){
        App app(nullptr);
        app.BuildTree();
        return app.exec();
}
```

Файл ObjectTree.h

```
#ifndef OBJECTTREE_H
#define OBJECTTREE_H
#include "Base.h"
class ObjectTree:public Base{
public:
        ObjectTree(Base* head, string name=""):Base(head, name)
        {};
};
#endif
```

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
root root 1 3 3	root root 1	root root 1
root root obj1 root obj2 obj2 obj3 obj2 obj4 obj5 obj5	root root obj1 obj2 obj2 obj3 obj4	root root obj1 obj2 obj2 obj3 obj4

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).