МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

« МИРЭА Российский технологический университет»

РТУ МИРЭА

Институт Информационных технологий

Кафедра Вычислительной техники

КУРСОВАЯ РАБОТА

по дисциплине

« Объектно-ориентированное программирование»

Наименование задачи:

« КЛ_3_2 Определение указателя на объект по его координате »

С тудент группы	ИКБО-13-21	Дамарад Д.В.
Руководитель практики	Ассистент	Асадова Ю.С.
Работа представлена	«___»_____2021 г.	_____
		(подпись студента)
Оценка		_____
		(подпись руководителя)

Москва 2021

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Постановка задачи

Иметь возможность доступа из текущего объекта к любому объекту системы, «мечта» разработчика программы.

В составе базового класса реализовать метод получения указателя на любой объект в составе дерева иерархии объектов согласно пути (координаты). В качестве параметра методу передать путь (координату) объекта. Координата задается в следующем виде:

/ - корневой объект;
//«имя объекта» - поиск объекта по уникальному имени от корневого (для однозначности уникальность требуется в рамках дерева);
. - текущий объект;
«имя объекта 1»[/«имя объекта 2»] . . . - относительная координата от текущего объекта, «имя объекта 1» подчиненный текущего;
/«имя объекта 1»[/«имя объекта 2»] . . . - абсолютная координата от корневого объекта.

Примеры координат:

```
/
//ob_3
.
ob_2/ob_3
ob_2
/ob_1/ob_2/ob_3
```

Если координата пустая строка или объект не найден, то вернуть нулевой указатель.

Система содержит объекты пяти классов, не считая корневого. Номера классов: 2,3,4,5,6.

Состав и иерархия объектов строиться посредством ввода исходных данных.
Ввод организован как в версии № 2 курсовой работы.
Единственное различие, в строке ввода первым указано не наименование головного объекта, а абсолютный путь к нему.
При построении дерева уникальность наименования относительно множества непосредственно подчиненных объектов для любого головного объекта соблюдены.
Добавить проверку допустимости исходной сборки. Собрать дерево невозможно, если по заданной координате головной объект не найден (например, ошибка в наименовании или еще не расположен на дереве объектов).

Система отрабатывает следующие команды:
SET «координата» – устанавливает текущий объект;
FIND «координата»– находит объект относительно текущего;
END – завершает функционирование системы (выполнение программы).
Изначально, корневой объект для системы является текущим.
При вводе данных в названии команд ошибок нет. Условия уникальности имен объектов для однозначной отработки соответствующих команд соблюдены.

Описание входных данных

Состав и иерархия объектов строится посредством ввода исходных данных. Ввод организован как в версии № 2 курсовой работы. Единственное различие, в строке ввода первым указано не наименование головного объекта, а абсолютный путь к нему.

После ввода состава дерева иерархии построчно вводятся команды:

SET «координата» - установить текущий объект;

FIND «координата» - найти объект относительно текущего;

END – завершить функционирование системы (выполнение программы).

Команды SET и FIND вводятся произвольное число раз. Команда END присутствует обязательно.

Пример ввода иерархии дерева объектов.

```
root
/                object_1                3
/                object_2                2
/object_2             object_4              3
/object_2             object_5              4
/                object_3                3
/object_2             object_3              6
/object_1             object_7              5
/object_2/object_4          object_7            3
endtree
FIND                 object_2/object_4
SET                     /object_2
```

```
FIND                    //object_5
FIND                    /object_15
FIND                    .
FIND                    object_4/object_7
END
```

Описание выходных данных

Первая строка:

`Object tree`

Со второй строки вывести иерархию построенного дерева как в курсовой работе версия №2.

При ошибке определения головного объекта, прекратить сборку, вывести иерархию уже построенного фрагмента дерева, со следующей строки сообщение:

`The head object «координата головного объекта» is not found`

и прекратить работу программы.

Если дерево построено, то далее построчно:

для команд SET если объект найден, то вывести:

`Object is set: «имя объекта»`

в противном случае:

`Object is not found: «имя текущего объекта»«искомая координата объекта»`

для команд FIND вывести:

«искомая координата объекта» Object name: «наименование объекта»

Если объект не найден, то: «искомая координата объекта» Object is not found

Пример вывода иерархии дерева объектов.

Object tree

root

object_1

object_7

object_2

object_4

object_7

object_5

object_3

object_3

object_2/object_4 Object name: object_4 Object is set: object_2 //object_5 Object name: object_5 /object_15 Object is not found . Object name: object_2 object_4/object_7 Object name: object_7

Метод решения

Для решения поставленной задачи используются:

- Объекты стандартных потоков ввода и вывода cin и cout соответсвенно для считывания с клавиатуры и вывода на экран.
- Объект app класс App.
- Библиотека контейнеров vector для создания вектора, который будет хранить указатели на дочерние объекты.
- Библиотека string для создания переменных строкового типа.
- Класс Base, который является базовым классом для классов App, Class2, Class3, Class4, Class5, Class6.
- Классы Class2, Class3, Class4, Class5, Class6, которые являются основными классами для подчиненных объектов.

Изменения, внесенные в архитектуру программы предыдущей программы КЛ_3_1:

- Изменены:
    - Конструктор класса Base.
    - Метода FindPtr класса Base.
    - Метод BuildTree класса App.
    - Метод exec класса App.
- Добавлены:
    - Поле wrong класса Base.
    - Поле cur_obj класса Base.
    - Метод WayPtr класса Base.
    - Метод SET класса Base.
    - Метод FIND класса Base.

- Оставлены без изменений:
    - Метод GetName класса Base.
    - Метод SetName класса Base.
    - Метод PrintTree класса Base.
    - Классы подчиненных объектов Class2, Class3, Class4, Class5, Class6.

Класс Base:

- Поля:
    - Поле, хранящее имя объекта:
        - Наименование - name;
        - Тип - string;
        - Модификатор доступа - protecred.
    - Поле, хранящее указатель на объект:
        - Наименование - head;
        - Тип - указатель на объект класса Base.
        - Модификатор доступа - protected.
    - Поле, хранящее указатель на текущий объект:
        - Наименование - cur_obj;
        - Тип - указатель на объект класса Base.
        - Модификатор доступа - protected.
    - Поле неправильной координаты:
        - Наименование - wrong;
        - Тип - string;
        - Модификатор доступа - protecred.
    - Поле, хранящее контейнер указателей на дочерние объекты:
        - Наименование - children.
        - Тип - контейнер указателей на объекты типа Base.

        - Модификатор доступа - public.
- Методы:
    - Метод Base:
        - Функционал - параметризированный контсруктор.
    - Метод SetName:
        - Функционал - сеттер поля name.
    - Метод GetName:
        - Функционал - геттер поля name.
    - Метод PrintTree:
        - Функционал - вывод дерева иерархии.
    - Метод FindPtr:
        - Функционал - поиск указателя на объект по имени.
    - Метод WayPtr:
        - Функционал -поиск указателя на объект по координате.
    - Метод FIND:
        - Функционал - поиск объекта по координате.
    - Метод SET:
        - Функционал - установка текущего объекта по координате .

Класс App:

- Поля:
    - Отсутствуют.
- Методы:
    - Метод App:
        - Функционал - делегирующий контсруктор.
    - Метод BuildTree:
        - Функционал - построение дерева иерархии.

        - Метод exec:
            - Функционал - метод запуска приложения.

Класс Class2:

- Поля:
    - Отсутствуют.
- Методы:
    - Метод Class2:
        - Функционал - делегирующий контсруктор.

Класс Class3:

- Поля:
    - Отсутствуют.
- Методы:
    - Метод Class3:
        - Функционал - делегирующий контсруктор.

Класс Class4:

- Поля:
    - Отсутствуют.
- Методы:
    - Метод Class4:
        - Функционал - делегирующий контсруктор.

Класс Class5:

- Поля:
    - Отсутствуют.

- Методы:
    - Метод Class5:
        - Функционал - делегирующий контсруктор.

Класс Class6:

- Поля:
    - Отсутствуют.
- Методы:
    - Метод Class6:
        - Функционал - делегирующий контсруктор.

№	Название класса	Класс-наследник	Модификатор доступа при наследовании	Описание	Номер	Комментарий
1	Base	App	public	Базовый класс	2	
		Class2	public		3	
		Class3	public		4	
		Class4	public		5	
		Class5	public		6	
		Class6	public		7	
2	App			Класс объектов, подчиненных классу Base		
3	Class2			Класс объектов, подчиненных классу Base		
4	Class3			Класс		

				объектов, подчиненных классу Base		
5	Class4			Класс объектов, подчиненных классу Base		
6	Class5			Класс объектов, подчиненных классу Base		
7	Class6			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 объекта app типа App и передача в его конструктор указатель nullptr	2	
2		Вызов у объекта app метода BuildTree	3	
3		Возврат результата выполнения метода exec у объекта app	∅	

Конструктор класса: Base

Модификатор доступа: public

Функционал: Параметризированный конструктор

Параметры: Указатель на объект типа Base, строка - name

Алгоритм конструктора представлен в таблице 3.

Таблица 3. Алгоритм конструктора класса Base

№	Предикат	Действия	№ перехода	Комментарий
1		Присвоение полю head параметр head	2	
2		Присвоение полю name параметр name	3	
3		Присвоение полю wrong значения ""	4	
4	head не нулевой указатель	Запись указателя на данный объект в вектор дочерних объектов children по указателю head	∅	
		Присвоение полю cur_obj указателя this	∅	

Класс объекта: Base

Модификатор доступа: public

Метод: FindPtr

Функционал: Поиск указателя на объект по имени

Параметры: Строка - имя, указатель на объект класса Base, булева переменная f

Возвращаемое значение: Указатель на объект класса Base

Алгоритм метода представлен в таблице 4.

Таблица 4. Алгоритм метода FindPtr класса Base

№	Предикат	Действия	№ перехода	Комментарий
1	Параметр f - ложь		2	
			9	
2	Параметр ptr не нулевой указатель		3	
		Возврат нулевого	∅	

		указателя		
3		Объявление указателя на объект класса Base	4	
4	Цикл со счетчиком i	Присвоение указателю temp указателя из вектора children по индексу i	5	
		Возврат нулевого указателя	∅	
5	Имя по указателю temp совпадает с нужным именем	Возврат указателя temp	∅	
			6	
6	Объект по указателю temp имеет свой вектор дочерних объектов		7	
			8	
7	Результат выполнения метода FindPtr с передаваемыми значениями name, temp не нулевой указатель	Возврат итога выполнения метода FindPtr с передаваемыми значениями name, temp	∅	
			8	
8	i меньше размера вектора children	Увеличение i на 1	4	
			4	
9	Цикл со счетчиком i		10	
			∅	
10	Указатель из вектора указателей на дочерние объекты имеет такое же имя, что и параметр name	Увеличение i на 1	∅	
			11	

11	i меньше размера вектора дочерних объектов	Увеличение i на 1	9	
			9	

Класс объекта: Base

Модификатор доступа: public

Метод: WayPtr

Функционал: Поиск указателя на объект по координате

Параметры: Строка переменная directory, булева переменная f

Возвращаемое значение: Указатель на объект типа Base

Алгоритм метода представлен в таблице 5.

Таблица 5. Алгоритм метода WayPtr класса Base

№	Предикат	Действия	№ перехода	Комментарий
1		Инициализация строковой переменной text пустой строкой	2	
2		Инициализация указателя temp на объект класса Base и присвоение ему нулевого указателя	3	
3	Переданная координата имеет в начале строки "//"	Удаление // в начале строки, присвоение укзателю temp итога вызова метода FindPtr с параметрами directory, this, false. Возврат temp	∅	
			4	
4		Инициализация указателя temp2 на объект класса Base с присвоением ему	5	

		поля cur_obj		
5	Цикл со счетчиком i		6	
			∅	
6	Символ из строки directory равен '/' или это последний символ в строке		7	
		Прибавление к переменной text символа directory[i]	12	
7	Символ из строки не '/'	Прибавление к переменной text символа directory[i]	8	
			8	
8		Присвоение переменной temp2 результата вызова FindPtr с параметрами text, temp2, true	9	
9	temp2 равен нулевому указателю	Возврат temp2	∅	
			10	
10	Больше в строке нет символов	Возврат temp2	∅	
			11	
11		Обнуление строки text	12	
12	i меньше размера вектора children	Увеличение i на 1	5	
			5	

Класс объекта: Base

Модификатор доступа: public

Метод: SET

Функционал: Установка текущего объекта по координате

Параметры: Строка - name

Возвращаемое значение: Отсутсвует

Алгоритм метода представлен в таблице 6.

Таблица 6. Алгоритм метода SET класса Base

№	Предикат	Действия	№ перехода	Комментарий
1	Результат вызова метода WayPtr с параметром name не равен нулевому указателю	Присвоение полю cur_obj результата вызова метода WayPtr с параметром name	2	
		Вывод на экран "Object is not found: ", вывод на экран параметра name	∅	
2		Вывод на экран "Object is not set: ", вывод на экран параметра name	∅	

Класс объекта: Base

Модификатор доступа: public

Метод: FIND

Функционал: Поиск объекта по координате

Параметры: Строка - directory

Возвращаемое значение: Отсутсвует

Алгоритм метода представлен в таблице 7.

Таблица 7. Алгоритм метода FIND класса Base

№	Предикат	Действия	№ перехода	Комментарий
1		Вывод на экран строки вводимой координаты	2	
2	Строка directory не равна "."		3	
		Вывод на экран "Object name: ", вывод на экран параметра cur_obj	∅	
3		Инициализация указаетля temp на объект класса Base с присвоением итога выполнения метода WayPtr с параметрами direcotry	4	
4	temp не нулевой указатель	Вывод на экран "Object name: ", вывод на экран итога выполнения метода GetName по указателю temp	∅	
		Вывод на экран "Object is not found"	∅	

Класс объекта: App

Модификатор доступа: public

Метод: BuildTree

Функционал: Построение дерева иерархии

Параметры: Отсуствуют

Возвращаемое значение: Отсуствует

Алгоритм метода представлен в таблице 8.

Таблица 8. Алгоритм метода BuildTree класса App

№	Предикат	Действия	№ перехода	Комментарий
1		Объявление строковых a,b	2	
2		Объявление целочисленной переменной c	3	
3		Считывание с клавиатуры значения переменной a	4	
4		Присваивание полю name головного объекта значения переменной a	5	
5	Бесконечный цикл	Считывание с клавиатуры значения переменной a	6	
			∅	
6	Значения a равняется строке endtree		∅	
			7	
7	Значения a равняется "/"	Присвоение переменной a поля name головного объекта	8	
			8	
8		Считывание с клавиатуры значенния переменныз b,c	9	
9	Значение переменной a равняется полю name головного объекта	Создание в зависимости от значения с объекта класса Class 2/3/4/5/6 с параметрами this, b	5	
			10	
10		Инициализация указателя на объект типа Base с присвоением результат выолнения метода WayPtr с параметрами a, true	11	

11	temp не равняется нулевому указателю	Создание в зависимости от значения с объекта класса Class 2/3/4/5/6 с параметрами temp, b	5	
		Присвоение полю wrong значения переменной a	∅	

Класс объекта: App

Модификатор доступа: public

Метод: exec

Функционал: Метод запуска приложения

Параметры: Отсуствуют

Возвращаемое значение: Целочисленной значение - код возврата

Алгоритм метода представлен в таблице 9.

Таблица 9. Алгоритм метода exec класса App

№	Предикат	Действия	№ перехода	Комментарий
1		Вывод на экран "Object tree", переход на новую строку	2	
2		Вывод на экран результата вызова метода GetName, переход на новую строку	3	
3		Вызов метода ShowTree с параметром ""	4	
4	Поле wrong не равно пустой строке	Вывод на экран информации о том, что произошла ошибка при построении дерева из-за неверной координаты. Возврат 0	∅	

		Объявление строковых переменных function, object	5	
5	Бесконечный цикл	Считывание с клавиатуры значения переменной function	6	
			9	
6	Значение function равно "END"	Выход из цикла	∅	
			7	
7	Значение function равно "SET"	Вызов метода SET с параметром object	5	
			8	
8	Значение function равно "FIND"	Вызов метода FIND с параметром object	5	
			5	
9		Возврат 0	∅	

Блок-схема алгоритма

Представим описание алгоритмов в графическом виде на рисунках ниже.

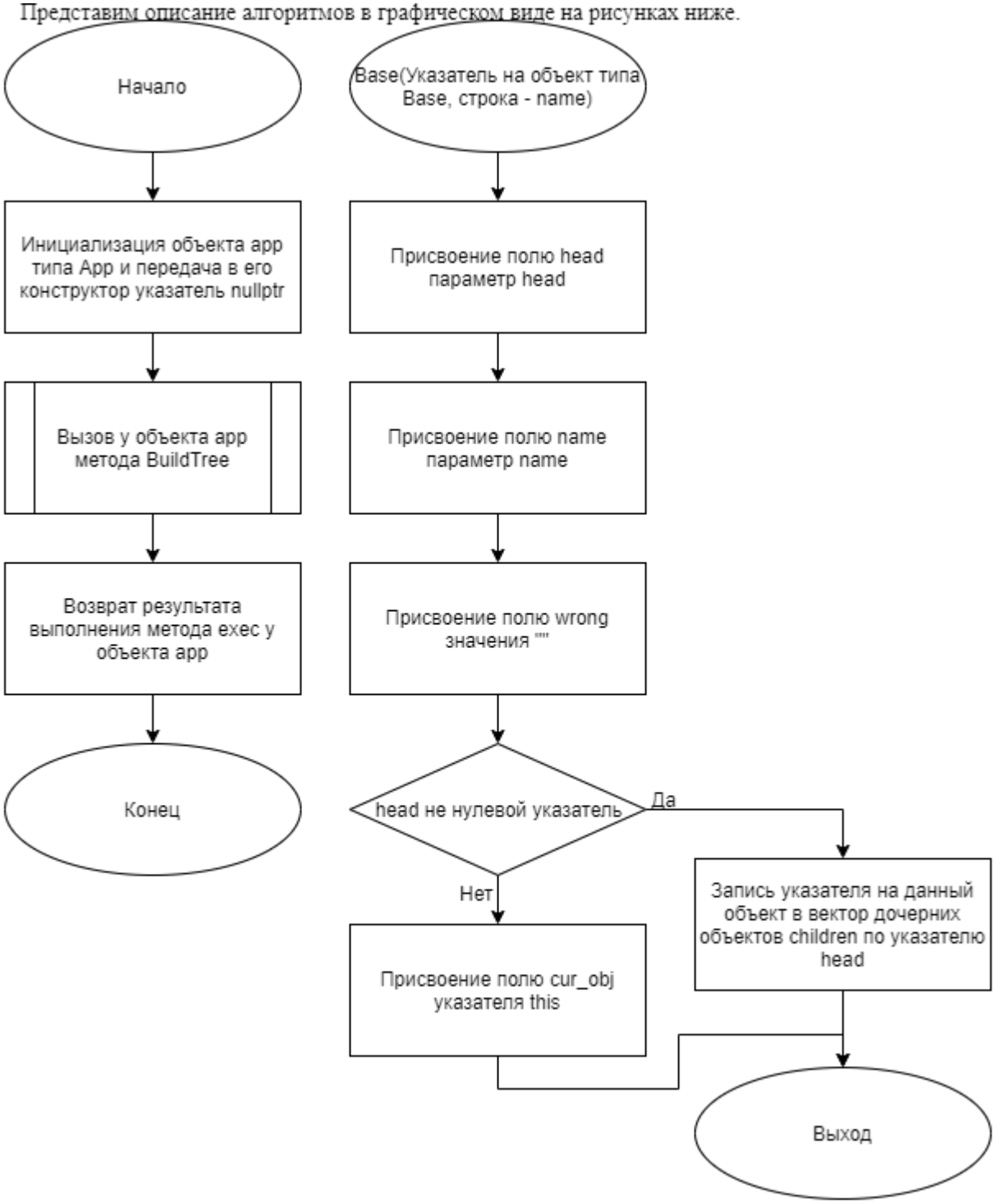

Рис. 1. Блок-схема алгоритма.

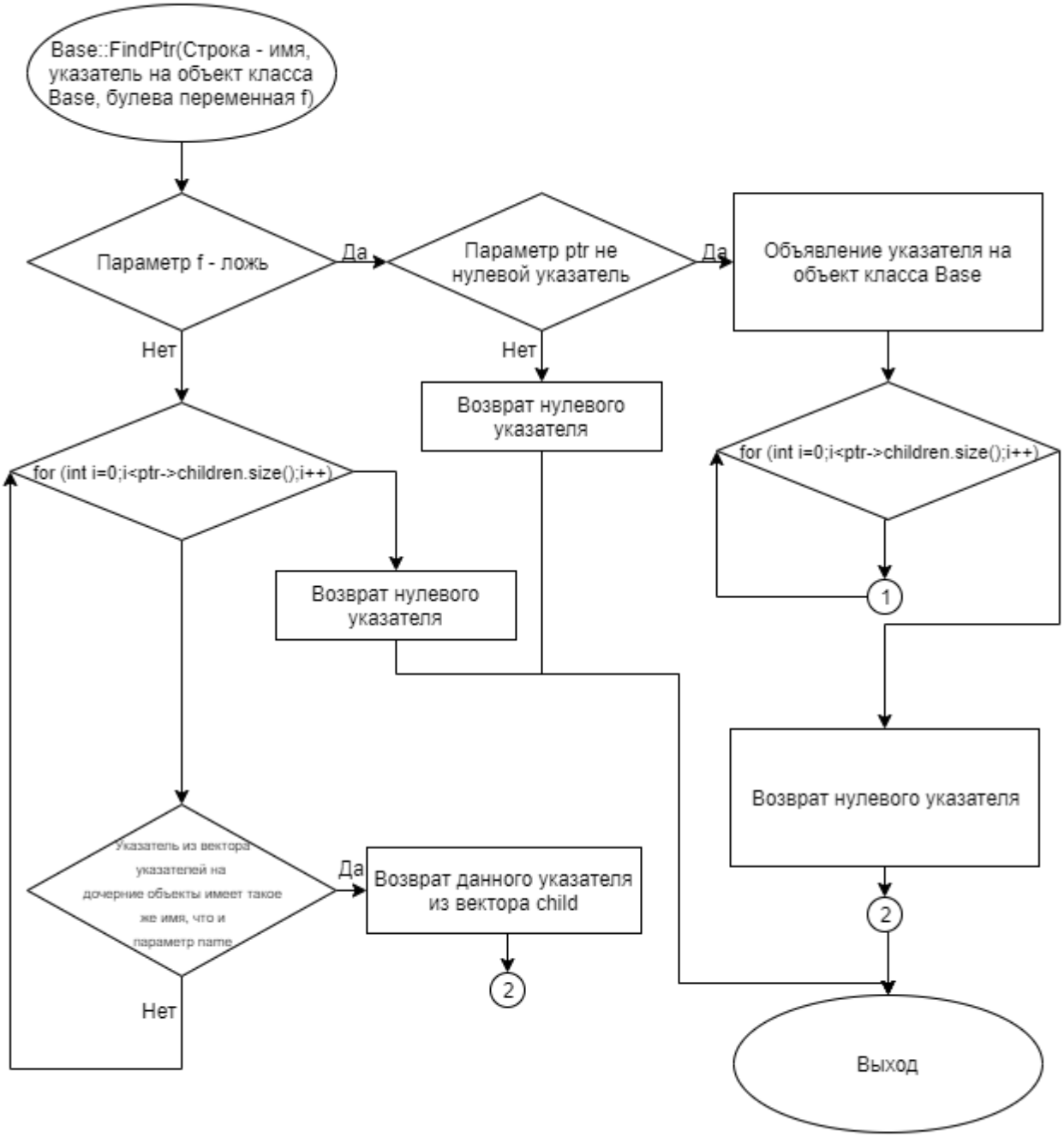

Рис. 2. Блок-схема алгоритма.

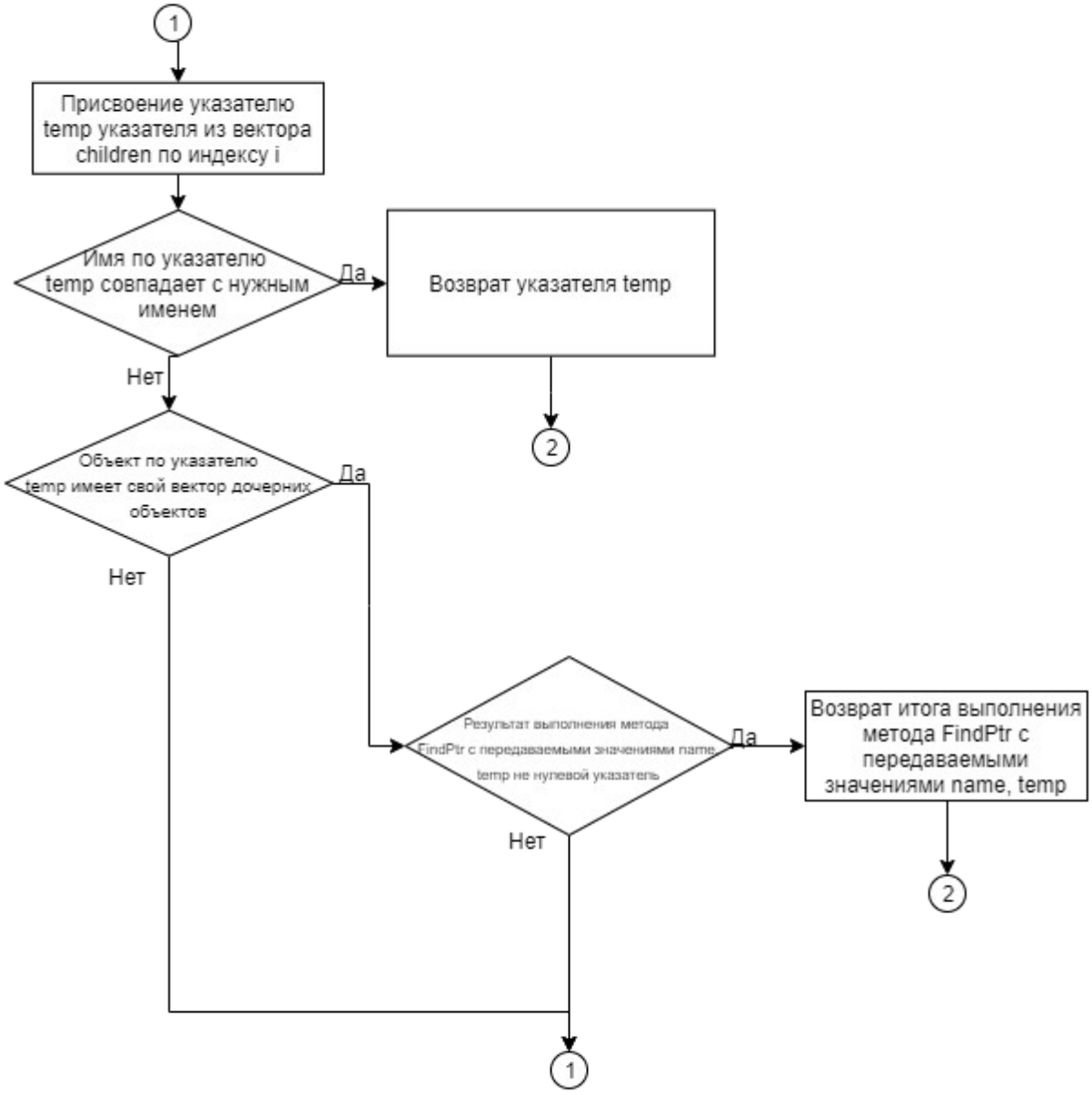

Рис. 3. Блок-схема алгоритма.

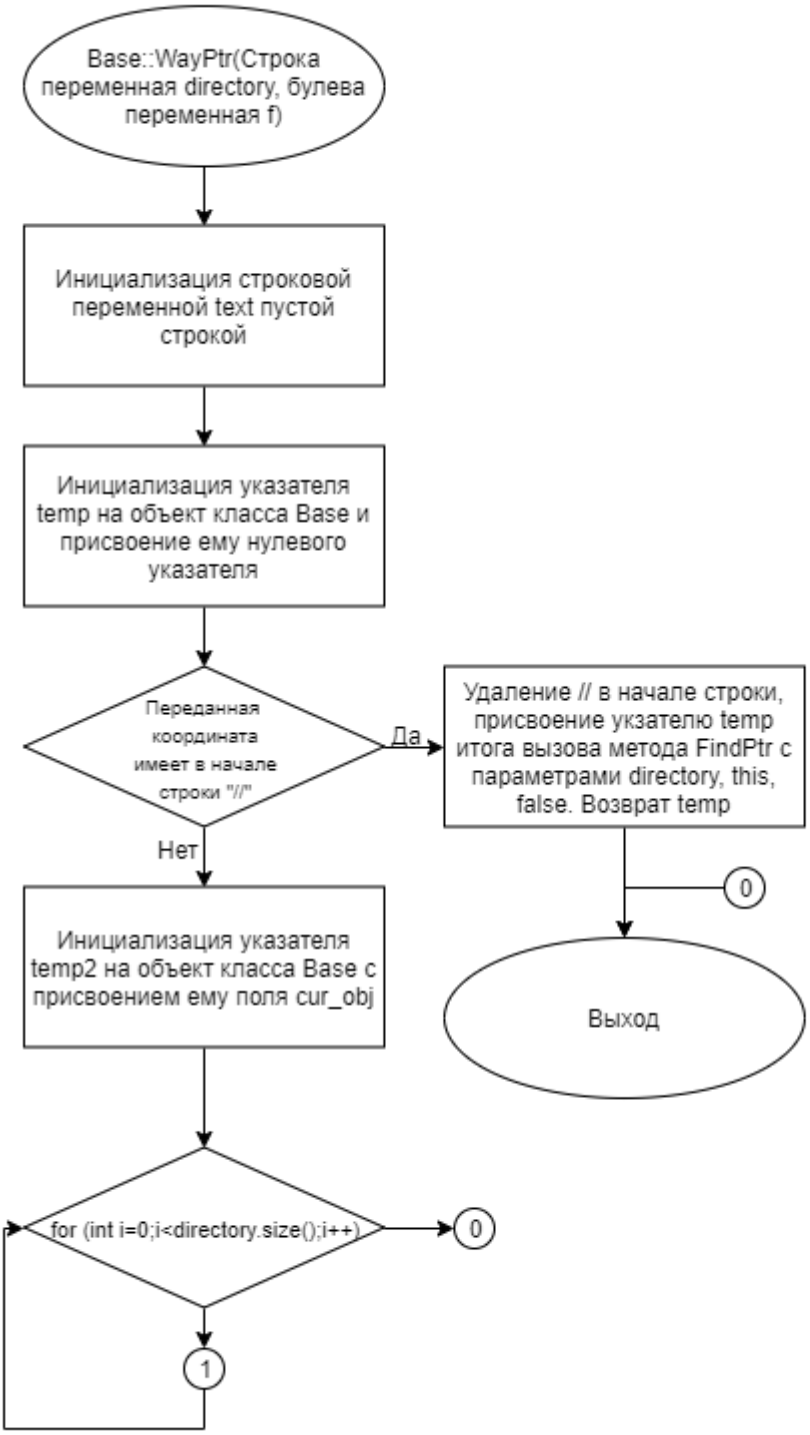

Рис. 4. Блок-схема алгоритма.

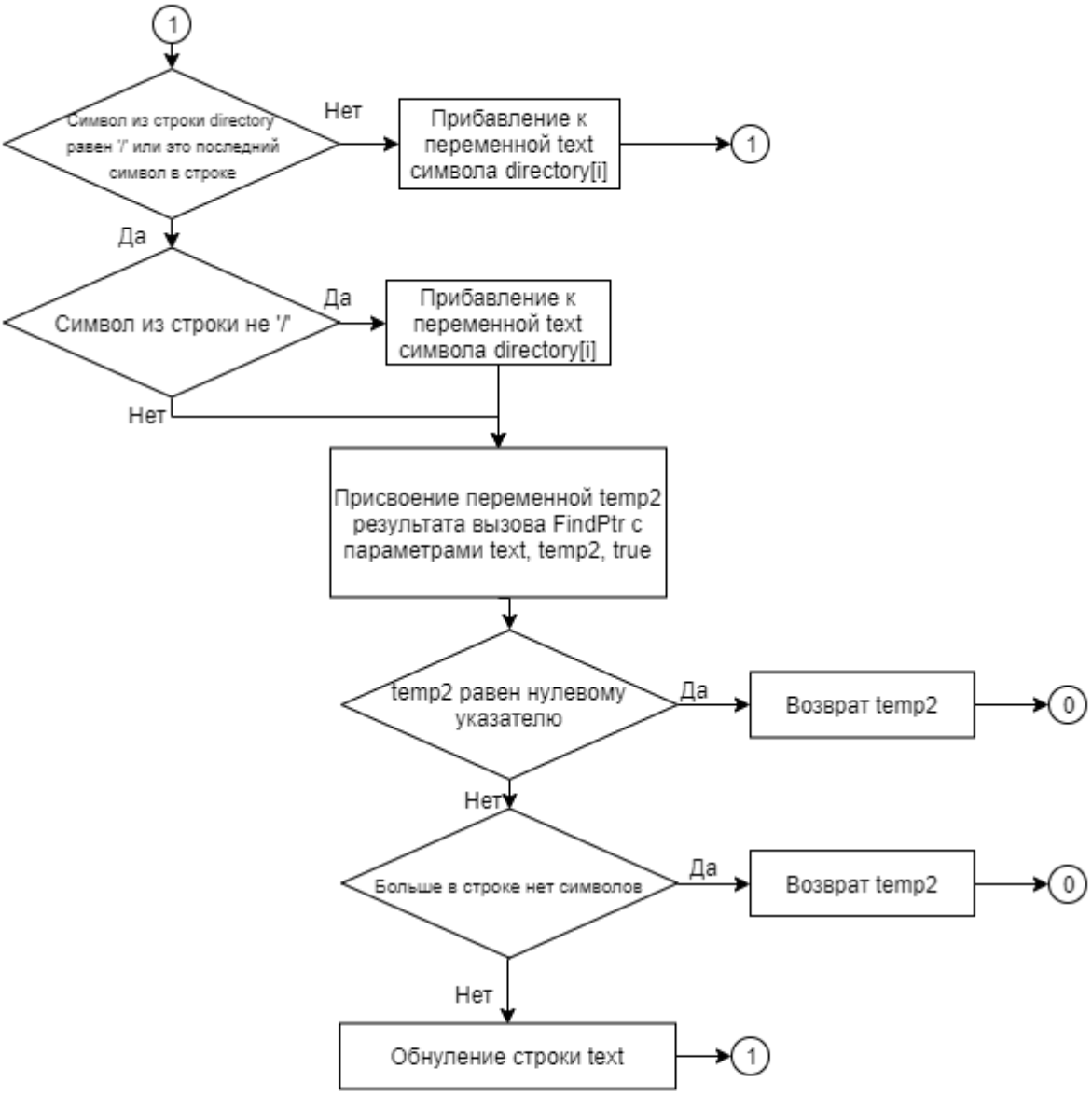

Рис. 5. Блок-схема алгоритма.

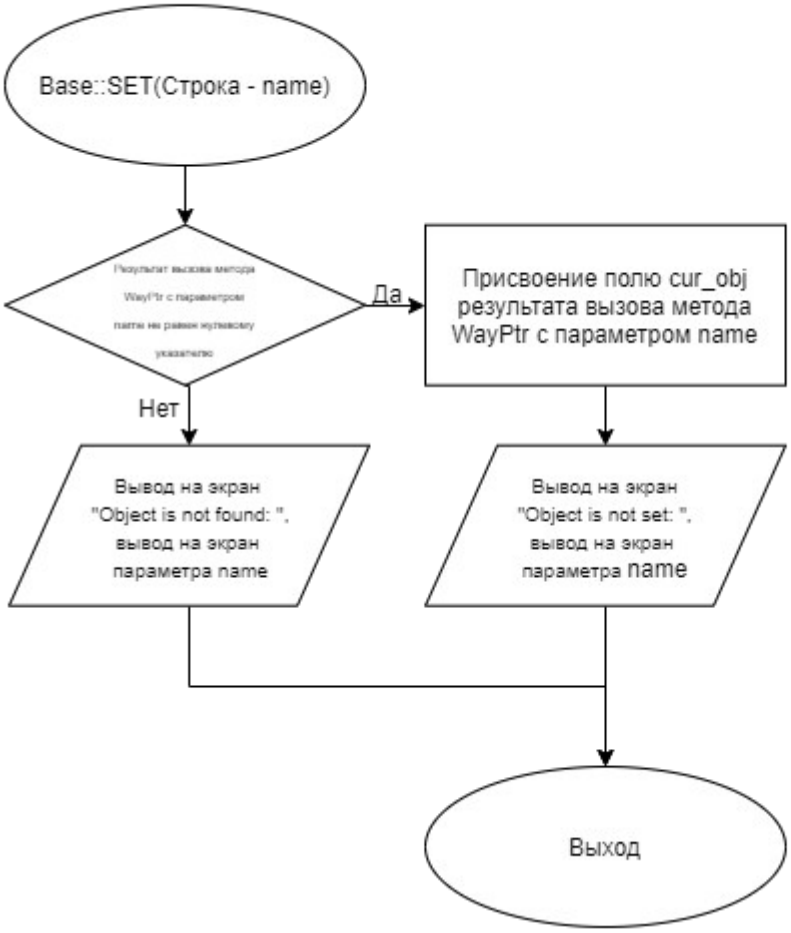

Рис. 6. Блок-схема алгоритма.

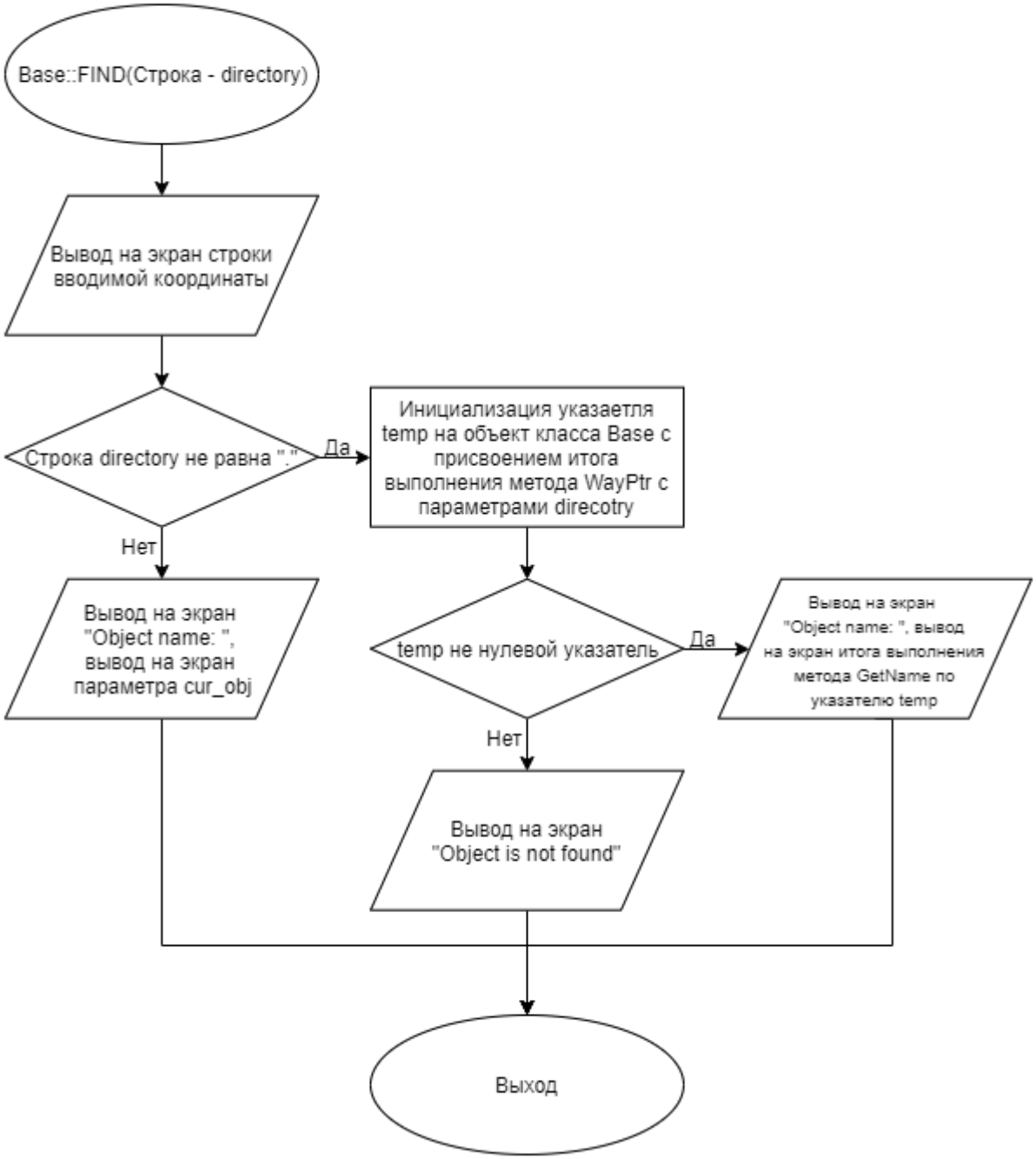

Рис. 7. Блок-схема алгоритма.

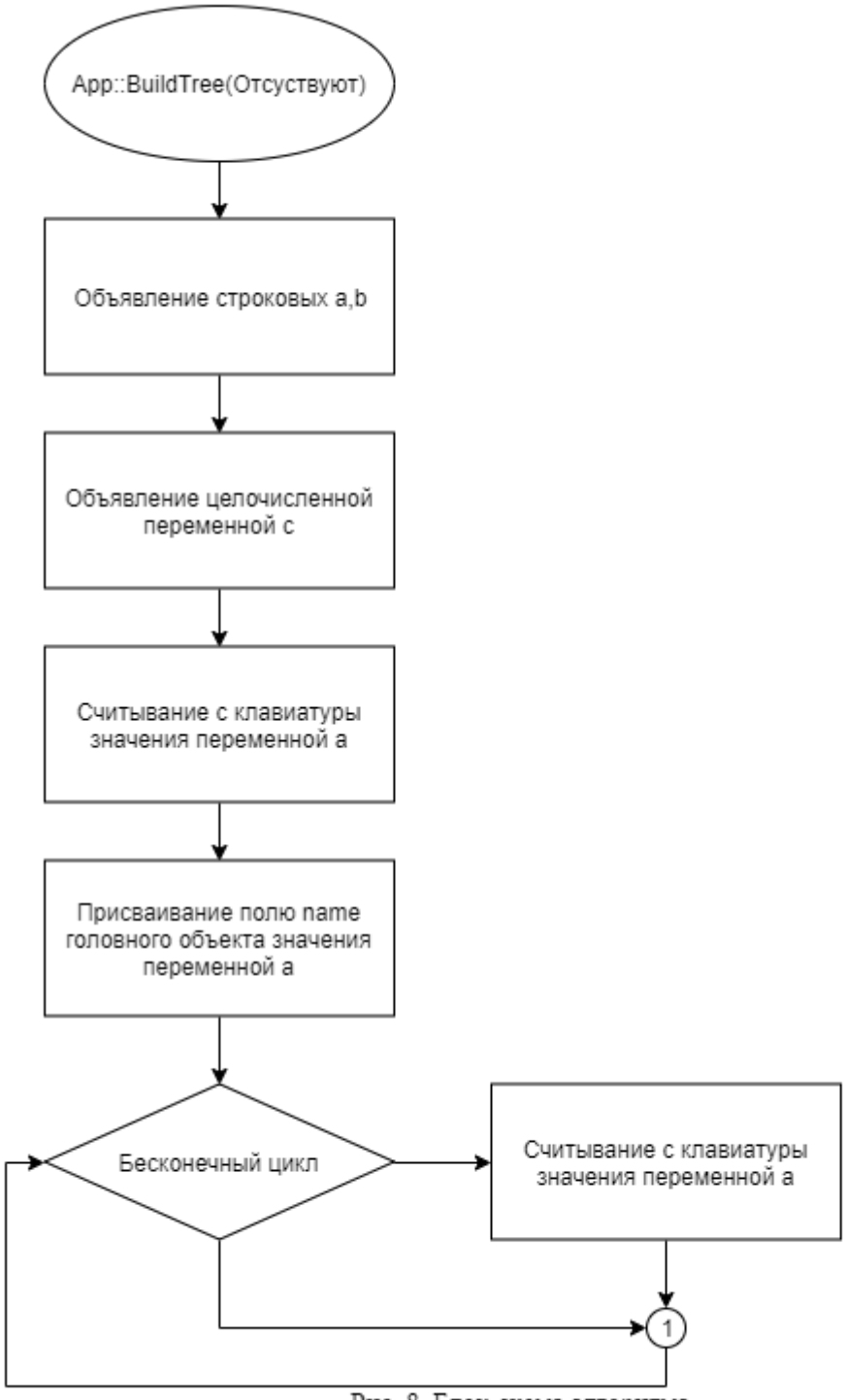

Рис. 8. Блок-схема алгоритма.

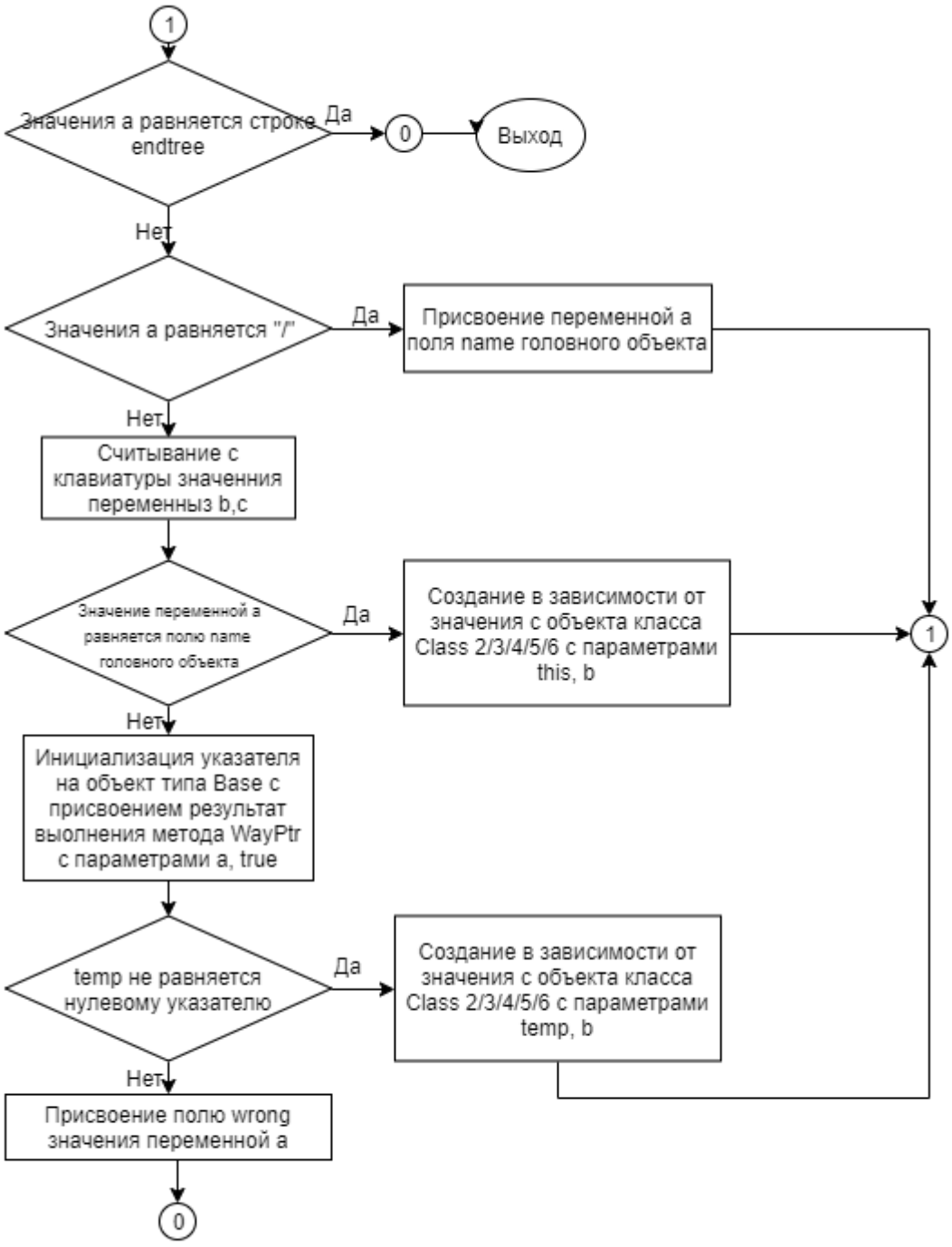

Рис. 9. Блок-схема алгоритма.

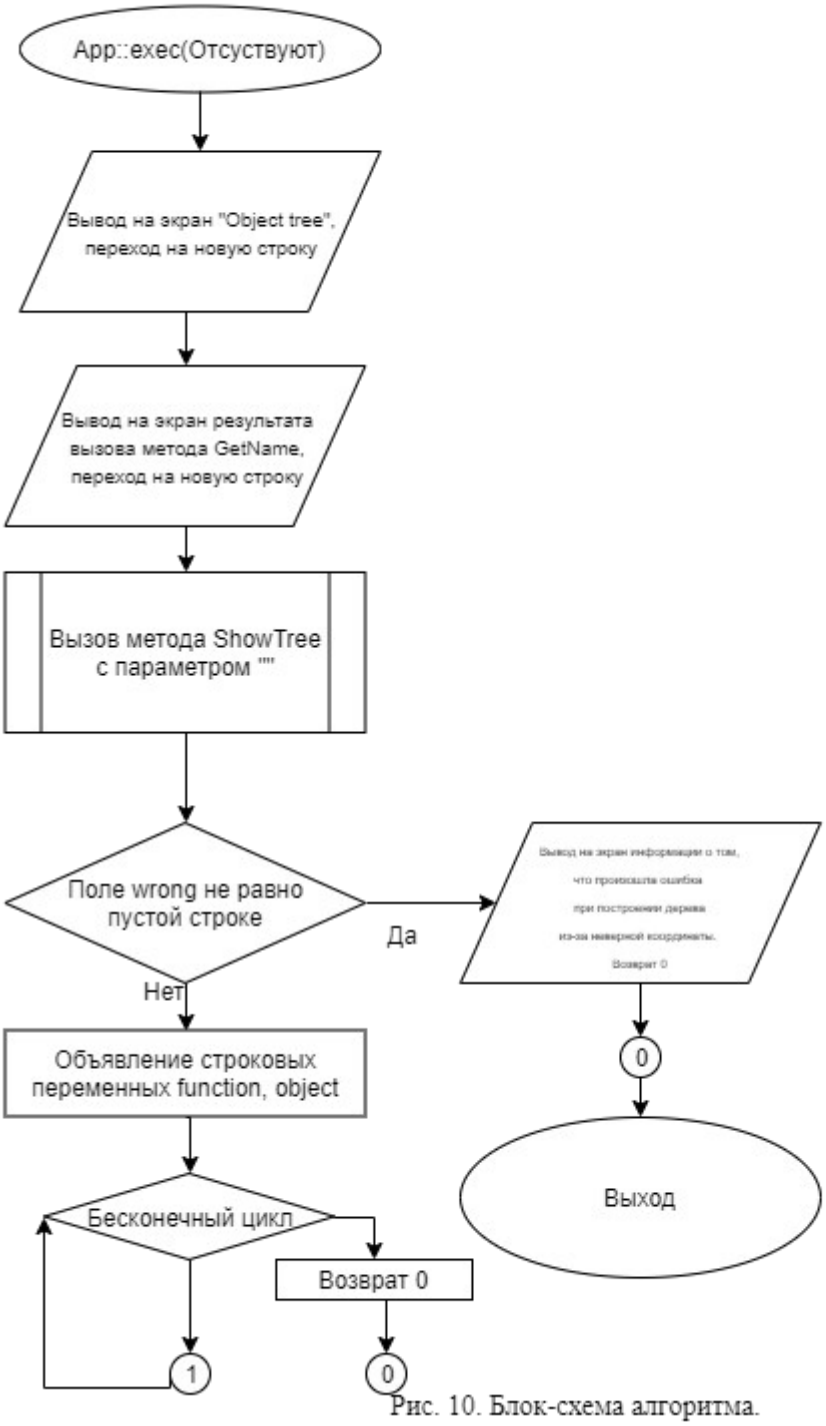

Рис. 10. Блок-схема алгоритма.

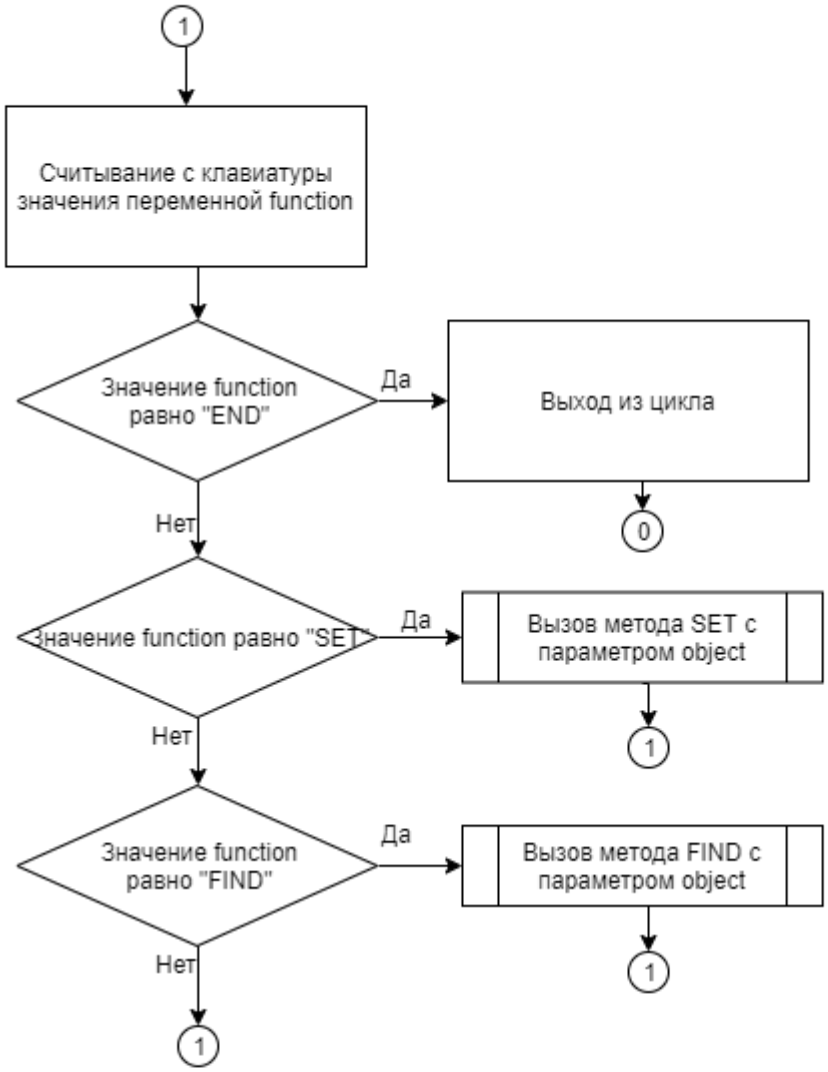

Рис. 11. Блок-схема алгоритма.

Код программы

Программная реализация алгоритмов для решения задачи представлена ниже.

Файл App.cpp

```
#include "App.h"
#include "Class2.h"
#include "Class3.h"
#include "Class4.h"
#include "Class5.h"
#include "Class6.h"
#include <string>
#include <iostream>
using namespace std;
int App::exec(){
        cout<<"Object tree"<<endl;
        cout<<this->GetName();
        PrintTree("");
        if (wrong!=""){
                cout<<endl<<"The head object "<<wrong<<" is not found";
                return 0;
        }
        else{
                string function,object;
                while (true){
                        cin>>function;
                        if (function=="END"){
                                break;
                        }
                        cin>>object;
                        if(function=="SET"){
                                this->SET(object);
                        }
                        if(function=="FIND"){
                                this->FIND(object);
                        }
                }
        }
        return 0;
}
void App::BuildTree(){
        string a,b;
        int cl;
        cin>>a;
        this->SetName(a);
        while(true){
                cin>>a;
                if (a=="endtree"){
                        break;
                }
                if (a=="/"){
                        a=this->name;
                }
```

```
                cin>>b>>cl;
                if (this->GetName()==a){
                        if (cl==2) new Class2(this, b);
                        if (cl==3) new Class3(this, b);
                        if (cl==4) new Class4(this, b);
                        if (cl==5) new Class5(this, b);
                        if (cl==6) new Class6(this, b);
                }
                else{
                        Base* temp=WayPtr(a, true);
                        if (temp!=nullptr){
                                if (cl==2) new Class2(temp, b);
                                if (cl==3) new Class3(temp, b);
                                if (cl==4) new Class4(temp, b);
                                if (cl==5) new Class5(temp, b);
                                if (cl==6) new Class6(temp, b);
                        }
                        else{
                                wrong=a;
                                break;
                        }
                }
        }
}
```

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){
        this->name=name;
        this->head=head;
        wrong = "";
        if (head!=nullptr){
                head->children.push_back(this);
```

```
        }
        else{
                cur_obj=this;
        }
}
void Base::SetName(string name){
        this->name=name;
}
string Base::GetName(){
        return name;
}
void Base::PrintTree(string space){
        for(int i=0;i<this->children.size();i++){
                cout<<endl;
                if (children[i]->children.size()>0){
                        cout<<space+"    "<<children[i]->GetName();
                        ((Base*)children[i])->PrintTree(space+"    ");
                }
                else{
                        cout<<space+"    "<<children[i]->GetName();
                }
        }
}
Base* Base::FindPtr(string name, Base* ptr, bool f){
        if(f==false){
                if (ptr!=nullptr){
                        Base* temp;
                        for(int i=0;i<ptr->children.size();i++){
                                temp=ptr->children[i];
                                if(temp->GetName()==name){
                                        return temp;
                                }
                                else if(temp->children.size()>0){
                                        if (FindPtr(name,temp)!=nullptr){
                                                return FindPtr(name,temp);
                                                break;
                                        }
                                }
                        }
                        return nullptr;
                }
                return ptr;
        }
        else{
                for(int i=0;i<ptr->children.size();i++){
                        if (ptr->children[i]->GetName()==name){
                                return ptr->children[i];
                        }
                }
                return nullptr;
        }
}

Base* Base::WayPtr(string directory, bool f){
        string text="";
        Base* temp=nullptr;
        if(directory=="/"){
                return this;
        }
        if (directory[0]=='/' and directory[1]=='/'){
```

```
                directory.erase(directory.begin()+0);
                directory.erase(directory.begin()+0);
                temp=FindPtr(directory,this,false);
                return temp;
        }
        else{
                Base* temp2=cur_obj;
                for(int i=0;i<directory.size();i++){
                        if (directory[i]=='/' or i==directory.size()-1){
                                if (text!=""){
                                        if (directory[i]!='/'){
                                                text+=directory[i];
                                        }
                                        temp2=FindPtr(text,temp2,true);
                                        if(temp2==nullptr){
                                                return temp2;
                                        }
                                        if (i==directory.size()-1){
                                                return temp2;
                                        }
                                        text="";
                                }
                        }
                        else{
                                text+=directory[i];
                        }
                }
                return nullptr;
        }
}
void Base::SET(string name) {
        if (WayPtr(name)!=nullptr){
                this->cur_obj=WayPtr(name);
                cout<<endl<<"Object is set: "<<this->cur_obj->GetName();
        }
        else{
                cout<<endl<<"Object is not found: "<<name;
        }
}
void Base::FIND(string directory){
        cout<<endl<<directory<<"     ";
        if (directory!="."){
                Base* temp=WayPtr(directory);
                if (temp!=nullptr){
                        cout<<"Object name: "<<temp->GetName();
                }
                else{
                        cout<<"Object is not found";
                }
        }
        else{
                cout<<"Object name: "<<this->cur_obj->GetName();
        }
}
```

Файл Base.h

```
#ifndef BASE_H
#define BASE_H
#include <iostream>
#include <string>
#include <vector>
using namespace std;
class Base{
protected:
        string name;
        string wrong;
        Base* head;
        Base* cur_obj;
public:
        Base(Base* head, string name="");
        void SetName(string name); //метод определения имени объекта
        string GetName(); //метод получения имени объекта
        void PrintTree(string space); //метод вывода наименований объектов в
дереве иерархии
        vector<Base*>children; //массив указателей на объекты, подчиненные к
текущему объекту в дереве иерархии
        Base* FindPtr(string name, Base* ptr=nullptr, bool f=false); // метод
поиска объекта на дереве иерархии по имени
        Base* WayPtr(string directory, bool f=false);
        void SET(string name);
        void FIND(string directory);
};
#endif
```

Файл Class2.h

```
#ifndef _CLASS2_H
#define _CLASS2_H
#include "Base.h"
class Class2:public Base {
public:
        Class2(Base* ptr, string name=""):Base(ptr, name){};
};
#endif
```

Файл Class3.h

```
#ifndef _CLASS3_H
#define _CLASS3_H
#include "Base.h"
class Class3:public Base {
public:
```

```
        Class3(Base* ptr, string name=""):Base(ptr, name){};
};
#endif
```

Файл Class4.h

```
#ifndef _CLASS4_H
#define _CLASS4_H
#include "Base.h"
class Class4:public Base {
public:
        Class4(Base* ptr, string name=""):Base(ptr, name){};
};
#endif
```

Файл Class5.h

```
#ifndef _CLASS5_H
#define _CLASS5_H
#include "Base.h"
class Class5:public Base {
public:
        Class5(Base* ptr, string name=""):Base(ptr, name){};
};
#endif
```

Файл Class6.h

```
#ifndef _CLASS6_H
#define _CLASS6_H
#include "Base.h"
class Class6:public Base {
public:
        Class6(Base* ptr, string name=""):Base(ptr, name){};
};
#endif
```

Файл main.cpp

```
#include "App.h"
```

```
int main(){
        App app(nullptr);
        app.BuildTree();
        return app.exec();
}
```

Тестирование

Результат тестирования программы представлен в следующей таблице.

Входные данные	Ожидаемые выходные данные	Фактические выходные данные
root / object_1 3 / object_2 2 /object_2 object_4 3 /object_2 object_5 4 / object_3 3 /object_2 object_3 6 /object_1 object_7 5 /object_2/object_4 object_7 3 endtree FIND object_2/object_4 SET /object_2 FIND //object_5 FIND /object_15 FIND . FIND object_4/object_7 END	Object tree root object_1 object_7 object_2 object_4 object_7 object_5 object_3 object_3 object_2/object_4 Object name: object_4 Object is set: object_2 //object_5 Object name: object_5 /object_15 Object is not found . Object name: object_2 object_4/object_7 Object name: object_7	Object tree root object_1 object_7 object_2 object_4 object_7 object_5 object_3 object_3 object_2/object_4 Object name: object_4 Object is set: object_2 //object_5 Object name: object_5 /object_15 Object is not found . Object name: object_2 object_4/object_7 Object name: object_7

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).