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	ИНБО-15-20	Ло В.Х.
Руководитель практики	Ассистент	Рогонова О.Н.
Работа представлена	«___»_____ 2021 г.	_____
		(подпись студента)
Оценка		_____
		(подпись руководителя)

Москва 2021

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. 3

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: 12

- €€€€€€ 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

Использую потоки ввод/вывода(cin/cout)

1. Объект класса: cl_base
   Описание класса: cl_base
   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.

2. Объект класса: cl_application наследуется от cl_base
   Описание класса: cl_application
   Методы:
   bild_tree_objects() - построение дерева объекта
   exec_app() - запуск системы
3. Объект класса: cl_2 наследуется от cl_base
   Описание класса: cl_2
   Свойства: string name, cl_base* parent
   Методы:
   cl_2(string name, cl_base* parent):cl_base(name, parent) - параметризированный конструктор cl_2 наследуется от cl_base

Описание алгоритма

Класс объекта: cl_base

Модификатор доступа: public

Метод: cl_base

Функционал: параметризированный конструктор с параметрами

Параметры: cl_base* parent, string name

Возвращаемое значение: нет

№	Предикат	Действия	№ перехода	Комментарий
1		set_object_name(name);	2	
2		this->parent=parent;	3	
3	this->parent!=0	this->parent->children.push_back(this);	∅	
			∅	

Класс объекта: cl_base

Модификатор доступа: public

Метод: set_object_name()

Функционал: опеределения имени объекта

Параметры: string name

Возвращаемое значение: нет

№	Предикат	Действия	№ перехода	Комментарий
1		опеределения имени объекта	∅	

Класс объекта: cl_base

Модификатор доступа: public

Метод: get_object_name()

Функционал: получения имени объекта

Параметры: нет

Возвращаемое значение: нет

№	Предикат	Действия	№ перехода	Комментарий
1		получения имени объекта	∅	

Класс объекта: cl_base

Модификатор доступа: public

Метод: show_tree()

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

Параметры: нет

Возвращаемое значение: нет

№	Предикат	Действия	№ перехода	Комментарий
1		вывода наименований объектов в дереве иерархии слева направо и сверху вниз	∅	

Класс объекта: cl_application

Модификатор доступа: public

Метод: exec_app()

Функционал: запуск системы

Параметры: нет

Возвращаемое значение: код, int возврата

№	Предикат	Действия	№ перехода	Комментарий
1		this->show_tree(); return 0;	∅	

Класс объекта: cl_application

Модификатор доступа: public

Метод: bild_tree_objects()

Функционал: построение дерева объекта

Параметры: нет

Возвращаемое значение: нет

№	Предикат	Действия	№ перехода	Комментарий
1		cl_base* first; cl_base* second; ввод имени объекта	2	

2		string nameParent, nameChild;
		first=this;
		second=first;
			3	

3		ввод nameParent, nameChild	4	
4	nameParent==nameChild	break;	∅	
			5	
5	nameParent==second->get_object_name()	first=new cl_2(second,nameChild);	3	
			6	
6	nameParent==first->get_object_name()	second=first; first=new cl_2(second,nameChild);	∅	
			∅	

Класс объекта: cl_2

Модификатор доступа: public

Метод: cl_2

Функционал: параметризированный конструктор cl_2 наследуется от cl_base

Параметры: string name, cl_base* parent

Возвращаемое значение: нет

№	Предикат	Действия	№ перехода	Комментарий
1			∅	

Класс объекта: cl_base

Модификатор доступа: public

Метод: set_parent()

Функционал: переопределение головного объекта для текущего в дереве иерархии

Параметры: cl_base* parent

Возвращаемое значение: нет

№	Предикат	Действия	№ перехода	Комментарий
1	parent		2	
			∅	
2	this->parent	int i=0	3	
		this->parent=parent; this->parent->children.push_back(this);	∅	
3	i<this->parent->children.size()		4	
			∅	
4	this->parent->children[i]==this	this->parent->children.erase(children.begin()+i); i++	3	
		i++	3	

Класс объекта: cl_base

Модификатор доступа: public

Метод: get_parent()

Функционал: получения указателя на головной объект текущего объекта.

Параметры: нет

Возвращаемое значение: parent

№	Предикат	Действия	№ перехода	Комментарий
1		получения указателя на головной объект текущего объекта	∅	

Функция: main

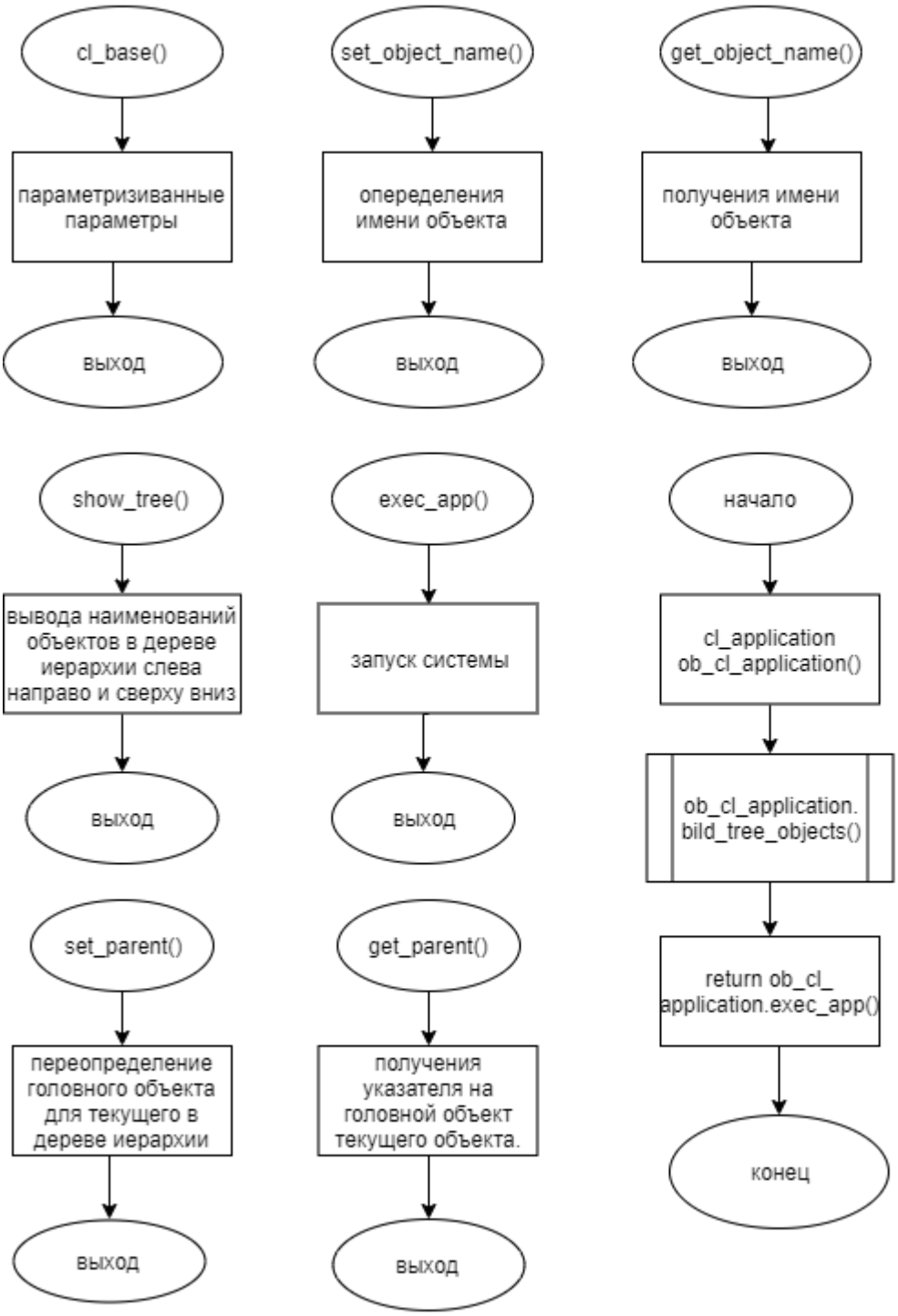
Функционал: основная программа

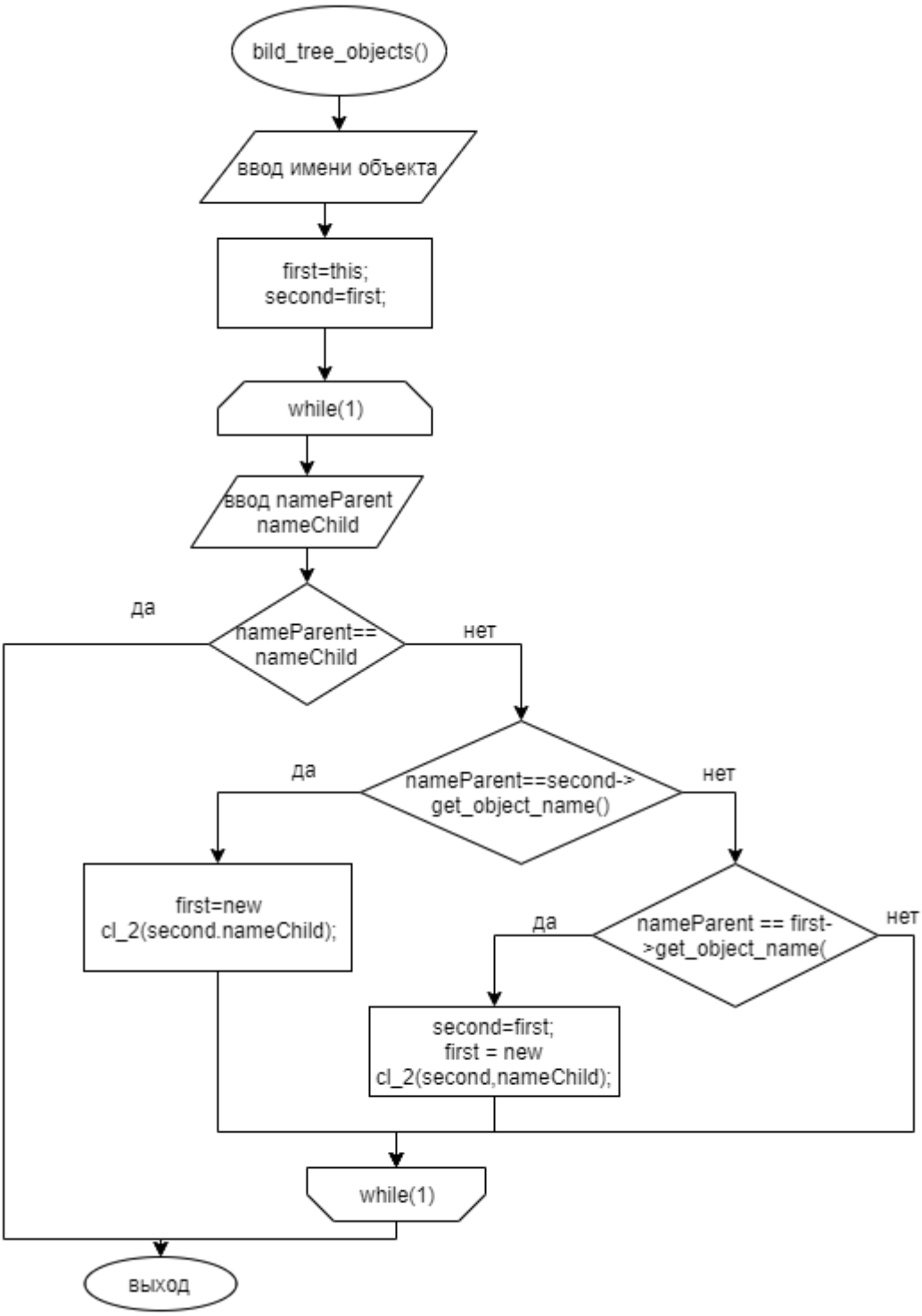
Параметры: нет

Возвращаемое значение: ob_cl_application.exec_app()

№	Предикат	Действия	№ перехода	Комментарий
1		возвращает функцию запуск системы	∅	

Блок-схема алгоритма


cl_base()
параметризиванные параметры
выход
set_object_name()
опеределения имени объекта
выход
get_object_name()
получения имени объекта
выход
show_tree()
вывода наименований объектов в дереве иерархии слева направо и сверху вниз
выход
exec_app()
запуск системы
выход
начало
cl_application ob_cl_application()
ob_cl_application. bild_tree_objects()
return ob_cl_ application.exec_app()
конец
set_parent()
переопределение головного объекта для текущего в дереве иерархии
выход
get_parent()
получения указателя на головной объект текущего объекта.
выход



Код программы

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(cl_base* parent,string name) : cl_base(parent,name){}
};
#endif
```

Файл cl_application.cpp

```
#include<iostream>
#include "cl_application.h"
#include "cl_2.h"
using namespace std;
void cl_application::bild_tree_objects()
{
        cl_base* first;
        cl_base* second;
        cin>>name;
        string nameParent,nameChild;
        first = this;
        second= first;
        while(1)
        {
                cin>>nameParent>>nameChild;
                if( nameParent == nameChild)
                {
                        break;
                }
                if(nameParent==second->get_object_name())
                        first= new cl_2(second, nameChild);
                else if(nameParent == first->get_object_name())
                {
                        second=first;
                        first = new cl_2(second,nameChild);
                }
        }
}
int cl_application::exec_app()
{
        this->show_tree();
        return 0;
}
```

Файл cl_application.h

```
#ifndef CL_APPLICATION_H
#define CL_APPLICATION_H
#include "cl_base.h"
class cl_application : public cl_base
{
public:
                void bild_tree_objects();
                int exec_app();
};
#endif
```

Файл cl_base.cpp

```
#include<iostream>
#include "cl_base.h"
cl_base::cl_base(cl_base* parent,string name)
{
        set_object_name(name);
        this->parent = parent;
        if(this->parent != 0)
        {
                this->parent->children.push_back(this);
}
}
cl_base::~cl_base()
{
        for(int i=0;i<this->children.size();i++)
        {
                delete this->children[i];
        }
}
void cl_base::set_object_name (string name)
{
        this->name = name;
}
string cl_base::get_object_name()
{
        return this->name;
}
void cl_base::show_tree()
{
        vector<cl_base*>::iterator it;
        if(parent==0)
                cout<<name<<endl;
        cout<<name;
        for(it = children.begin();it != children.end();it++)
```

```
        {
                cout<<"  "<<(*(it))->name;
        }
        for(it = children.begin();it != children.end();it++)
        {
                if((*it)->children.size()!=0)
                {
                        cout<<endl;
                        (*(it))->show_tree();
                        break;
                }
        }
}
void cl_base::set_parent(cl_base* parent)
{
        if(parent)
        {
                if(this->parent)
                {
                        for(int i = 0 ; i<this->parent->children.size();i++)
                        {
                                if(this->parent->children[i] == this)
                                {
                                        this->parent-
>children.erase(children.begin()+i);
                                }
                        }
                }
                this->parent = parent;
                this->parent->children.push_back(this);
        }
}
cl_base* cl_base:: get_parent()
{
        return parent;
}
```

Файл cl_base.h

```
#ifndef CL_BASE_H
#define CL_BASE_H
#include<iostream>
#include<vector>
#include<string>
using namespace std;
class cl_base
{
protected:
        vector <cl_base*> children;
        cl_base* parent;
        string name;
public:
```

```
        cl_base(cl_base* parent = nullptr,string name = "cl_base");
        ~cl_base();
        void set_object_name(string name);
        string get_object_name();
        void show_tree();
        void set_parent(cl_base* parent);
        cl_base* get_parent();
};
#endif
```

Файл main.cpp

```
#include "cl_application.h"
#include "cl_base.h"
#include "cl_2.h"
int main()
{
        cl_application ob_cl_application;
        ob_cl_application.bild_tree_objects();
        return ob_cl_application.exec_app();
}
```

Тестирование

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