МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ИНЖЕНЕРНО-ТЕХНОЛОГИЧЕСКАЯ АКАДЕМИЯ

Институт компьютерных технологий и информационной безопасности

Кафедра математического обеспечения и применения ЭВМ

**Лабораторная работа № 2**

по курсу «Объектно-ориентированное программирование» **«Наследование в С++»**

**10 ВАРИАНТ**

Выполнили:

студенты гр. КТбо2-1

Неприн М.А.

Проверил:

Тарасов С. А.

**Таганрог 2020**

1. **Вариант задания №10**

Описать класс Element (элемент логической схемы) с двумя входами и одним выходом и полем, хранящим название элемента. Определить функцию, которая преобразует входные двоичные значения в выходное. На его основе реализовать классы AND и OR - двоичные вентили, которые реализуют логическое умножение и сложение соответственно. В дополнительном классе Sсheme создать массив элементов (до 10) и обеспечить подачу двоичных сигналов на их входы с выводом выходных значений. Входные сигналы хранятся в файле

1. **Спецификация классов**

class Element

{

public:

Element();

bool getOutput();

void setInput(bool input);

virtual bool produceOutput() = 0;

protected:

bool input[2];

bool output;

bool isFirst;

};

class And : public Element

{

bool produceOutput() override;

};

class Or : public Element

{

bool produceOutput() override;

};

class Scheme

{

public:

Scheme();

~Scheme();

bool doStep();

bool getOutput();

void genNewInput();

private:

const int n;

Element\* logics[10];

bool output;

short int path[10][10];

};

class UI

{

public:

UI();

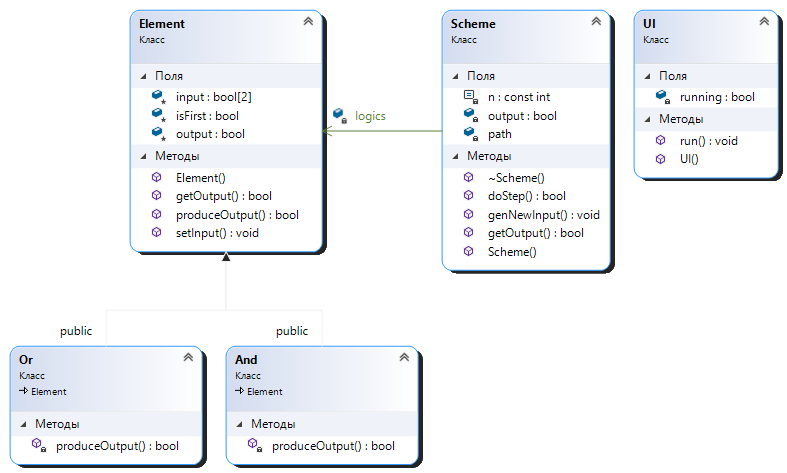
void run();

private:

bool running;

};

1. **Диаграммы классов**



1. **Листинг программы**

**main.cpp**

/\*

// ЮФУ ИКТИБ МОП ЭВМ

// Неприн Михаил Андреевич, КТбо2-1

// Лабораторная работ №2, Вариант №10

// 15.10.2020

\*/

#include "ui.h"

#include <random>

#include <ctime>

int main()

{

srand(time(0));

UI ui;

ui.run();

return 0;

}

**Element.h**

#pragma once

class Element

{

public:

Element();

bool getOutput();

void setInput(bool input);

virtual bool produceOutput() = 0;

protected:

bool input[2];

bool output;

bool isFirst;

};

**Element.cpp**

#include "Element.h"

Element::Element() : input{ 0,0 }, output(0), isFirst(0)

{

}

void Element::setInput(bool input)

{

if (isFirst) {

this->input[0] = input;

}

else {

this->input[1] = input;

}

isFirst = !isFirst;

return;

}

bool Element::getOutput()

{

return output;

}

**Or.h**

#pragma once

#include "Element.h"

class Or : public Element

{

bool produceOutput() override;

};

**Or.cpp**

#include "Or.h"

bool Or::produceOutput()

{

return output = (input[0] || input[1]);

}

**And.h**

#pragma once

#include "Element.h"

class And : public Element

{

bool produceOutput() override;

};

**And.cpp**

#pragma once

#include "Element.h"

class And : public Element

{

bool produceOutput() override;

};

**Scheme.h**

class Scheme

{

public:

Scheme();

~Scheme();

bool doStep();

bool getOutput();

void genNewInput();

private:

const int n;

Element\* logics[10];

bool output;

short int path[10][10];

};

**Scheme.cpp**

#include "Scheme.h"

#include <random>

#include "Or.h"

#include "And.h"

Scheme::Scheme() : n{ 10 }, output{}, path{

{0,0,1,1,0,0,0,0,0,0},

{0,0,1,0,1,0,0,0,0,0},

{0,0,0,1,1,1,1,0,0,0},

{0,0,0,0,0,1,0,0,0,0},

{0,0,0,0,0,0,1,0,0,0},

{0,0,0,0,0,0,0,1,1,0},

{0,0,0,0,0,0,0,1,1,0},

{0,0,0,0,0,0,0,0,0,1},

{0,0,0,0,0,0,0,0,0,1},

{0,0,0,0,0,0,0,0,0,0}

}

{

for (size\_t i = 0; i < n; ++i)

{

bool r = rand() % 2;

if (r) {

logics[i] = new Or;

}

else {

logics[i] = new And;

}

}

}

Scheme::~Scheme()

{

delete[] logics;

}

bool Scheme::doStep()

{

for (int i = 0; i < n; ++i) {

logics[i]->produceOutput();

}

for (int i = 0; i < n; ++i) {

for (int j = 0; j < n; j++) {

if (path[i][j]) {

logics[j]->setInput(logics[i]->getOutput());

}

}

}

return output = logics[9]->getOutput();

}

bool Scheme::getOutput()

{

return output;

}

void Scheme::genNewInput()

{

logics[0]->setInput(rand() % 2);

logics[0]->setInput(rand() % 2);

logics[1]->setInput(rand() % 2);

logics[1]->setInput(rand() % 2);

}

**ui.h**

#ifndef UI\_H

#define UI\_H

class UI

{

public:

UI();

void run();

private:

bool running;

};

#endif // UI\_H

**ui.cpp**

#include "ui.h"

#include "Scheme.h"

#include <iostream>

#include <Windows.h>

using std::cout; using std::endl;

UI::UI(): running {false}

{

}

void UI::run()

{

running = true;

Scheme scheme;

bool result = 0;

while (running)

{

result = scheme.doStep();

cout << result << endl;

scheme.genNewInput();

Sleep(100);

}

}