

МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

Кафедра інформаційної безпеки

Лабораторна робота №4

з дисципліни «Системна інженерія» на тему:

«Індивідуальний проект»

|  |  |
| --- | --- |
| Перевірив:  \_\_\_\_\_\_\_\_\_\_\_ | Виконала:  Студентка групи ФБ-91 Легенчук М. О. |

Київ – 2021

Система: Мультиварка

Мова програмування: С++

Програмний код:

**Multicooker.h:**

#pragma once

#include <iostream>

#include <Windows.h>

#include <string>

using namespace std;

class Display

{

protected:

string mode;

bool state;

public:

Display();

bool getState();

void setState();

};

class Button

{

protected:

bool state;

string type;

public:

Button();

void setState();

bool getState();

void setType(string t);

};

class ModeButton

{

protected:

int n\_state;

string state;

string type;

public:

ModeButton();

void setState();

int getState();

};

class ControlPanel

{

protected:

Button start;

Button stop;

ModeButton mode;

Display display;

public:

ControlPanel();

void makeSysToCook();

void selectMode();

bool checkButtons();

void finish();

void sound();

void displayState();

};

class SecuritySystem

{

protected:

bool blockedCover;

public:

SecuritySystem();

void blockCover();

bool checkBlock(bool m);

};

class CondensateSystem

{

protected:

int volume;

bool fullness;

bool state;

public:

CondensateSystem();

void setState();

bool getState();

};

class Power

{

protected:

int power;

bool state;

public:

Power();

void setState();

bool getState();

};

class Cover

{

protected:

string material;

bool state;

public:

Cover();

void setState();

bool getState();

};

class CookingContainer

{

protected:

int volume;

int size;

Cover cover;

Power power;

public:

CookingContainer();

bool chCover(bool m);

};

class Rosette

{

protected:

int power;

string type;

public:

Rosette();

void setState();

int getState();

};

class PowerSupply

{

protected:

Rosette rosette;

bool wire;

public:

PowerSupply();

bool energySupply();

void setPower();

};

class HeatingSystem

{

protected:

int power;

int temperature;

bool state;

public:

HeatingSystem();

void setState();

bool getState();

};

class Multicooker

{

protected:

ControlPanel controlSystem;

HeatingSystem heatingSystem;

CondensateSystem condensateSystem;

CookingContainer cookingContainer;

PowerSupply powerSupply;

Power power;

SecuritySystem securitySystem;

bool state;

public:

Multicooker();

bool isCooking();

void cooking();

void finishCooking();

void makeSysToHeat();

void turnOn();

void modeButton();

bool coverInfo(int m);

bool chPower(bool p);

bool getPower();

};

class Human

{

protected:

string name;

bool products;

public:

Human();

bool checkProducts(bool m);

void pushOnPowerButton(Multicooker\* mt);

void pushOnStartButton(Multicooker\* mt);

void pushOnStopButton(Multicooker\* mt);

void pushOnModeButton(Multicooker\* mt);

void openOrCloseCover(Multicooker\* mt);

};

**Multicooker.cpp:**

#include "Multicooker.h"

Display::Display()

{

mode = "Off";

state = FALSE;

}

bool Display::getState()

{

return state;

}

void Display::setState()

{

if (state == TRUE)

{

state = FALSE;

mode = "Off";

cout << "Дисплей вимкнено\n";

}

else

{

state = TRUE;

mode = "On";

cout << "Дисплей увімкнено\n";

}

}

Button::Button()

{

state = FALSE;

type = "Start";

}

void Button::setState()

{

if (state == TRUE)

state = FALSE;

else

state = TRUE;

}

void Button::setType(string t)

{

type = t;

}

bool Button::getState()

{

return state;

}

ModeButton::ModeButton()

{

n\_state = 1;

state = "Випічка";

type = "Mode";

}

void ModeButton::setState()

{

if (n\_state < 6)

++n\_state;

else

n\_state = 1;

switch (n\_state)

{

case 1:

state = "Випічка";

break;

case 2:

state = "Тушіння";

break;

case 3:

state = "Суп";

break;

case 4:

state = "На пару";

break;

case 5:

state = "Жарка";

break;

case 6:

state = "Йогурт";

break;

}

cout << "Встановлено режим: '" << state << "'\n";

}

int ModeButton::getState()

{

return n\_state;

}

ControlPanel::ControlPanel()

{

start;

stop.setType("Stop");

mode;

display;

}

void ControlPanel::makeSysToCook()

{

if (this->start.getState() == FALSE)

{

this->start.setState();

cout << "Мультиварка починає приготування їжі...\n";

}

else

cout << "Мультиварка вже працює!\n";

}

bool ControlPanel::checkButtons()

{

return this->start.getState();

}

void ControlPanel::finish()

{

if (this->start.getState() == TRUE)

{

this->start.setState();

cout << "Приготування їжі завершене!\n";

}

}

void ControlPanel::sound()

{

cout << "Можна забрати страву!\n";

}

void ControlPanel::selectMode()

{

mode.setState();

}

void ControlPanel::displayState()

{

display.setState();

}

SecuritySystem::SecuritySystem()

{

blockedCover = FALSE;

}

void SecuritySystem::blockCover()

{

if (this->blockedCover == FALSE)

{

blockedCover = TRUE;

cout << "Кришку мультиварки заблоковано\n";

}

else

{

blockedCover = FALSE;

cout << "Кришку мультиварки розблоковано\n";

}

}

bool SecuritySystem::checkBlock(bool m)

{

if (m == FALSE)

return blockedCover;

else

this->blockCover();

}

CondensateSystem::CondensateSystem()

{

volume = 3;

fullness = FALSE;

state = FALSE;

}

bool CondensateSystem::getState()

{

return state;

}

void CondensateSystem::setState()

{

if (state == TRUE)

{

state = FALSE;

cout << "Систему сбору конденсату деактивовано\n";

}

else

{

state = TRUE;

cout << "Систему сбору конденсату активовано\n";

}

}

Power::Power()

{

power = 220;

state = FALSE;

}

void Power::setState()

{

if (state == FALSE)

{

state = TRUE;

cout << "Мультиварку увімкнено\n";

}

else

{

state = FALSE;

cout << "Мультиварку вимкнено\n";

}

}

bool Power::getState()

{

return state;

}

Cover::Cover()

{

material = "iron";

state = FALSE;

}

void Cover::setState()

{

if (state == FALSE)

{

state = TRUE;

cout << "Кришку відкрито\n";

}

else

{

state = FALSE;

cout << "Кришку закрито\n";

}

}

bool Cover::getState()

{

return state;

}

CookingContainer::CookingContainer()

{

volume = 3;

size = 5;

cover;

power;

}

bool CookingContainer::chCover(bool m)

{

if (m == TRUE)

cover.setState();

else

return cover.getState();

}

Rosette::Rosette()

{

power = 220;

type = "standard";

}

void Rosette::setState()

{

cout << "Увімкнути мультиварку у розетку напругою...\n";

cout << "1 - 200 B\n";

cout << "2 - 220 В\n";

cout << "3 - 240 В\n";

cout << "0 - не змінювати розетку\n";

cout << "Команда>>";

int a;

cin >> a;

switch (a)

{

case 1:

{

cout << "Вибрано розетку напругою 200 В\n";

power = 200;

}

break;

case 2:

{

cout << "Вибрано розетку напругою 220 В\n";

power = 220;

}

break;

case 3:

{

cout << "Вибрано розетку напругою 240 В\n";

power = 240;

}

break;

default:

cout << "Розетку не було змінено. На даний момент напруга розетки: " << power << endl;

break;

}

}

int Rosette::getState()

{

return power;

}

PowerSupply::PowerSupply()

{

rosette;

wire = TRUE;

}

bool PowerSupply::energySupply()

{

if (this->rosette.getState() == 200)

return FALSE;

else

return TRUE;

}

void PowerSupply::setPower()

{

this->rosette.setState();

}

HeatingSystem::HeatingSystem()

{

temperature = 90;

power = 1000;

state = FALSE;

}

void HeatingSystem::setState()

{

if (state == TRUE)

{

cout << "Систему нагрівання деактивовано\n";

state = FALSE;

}

else

{

state = TRUE;

cout << "Систему нагрівання активовано\n";

}

}

bool HeatingSystem::getState()

{

return state;

}

Multicooker::Multicooker()

{

controlSystem;

heatingSystem;

condensateSystem;

cookingContainer;

powerSupply;

power;

securitySystem;

state = TRUE;

}

void Multicooker::makeSysToHeat()

{

this->heatingSystem.setState();

}

void Multicooker::turnOn()

{

if (this->powerSupply.energySupply() == TRUE)

{

this->power.setState();

this->controlSystem.displayState();

}

else

cout << "Недостатньо напруги для увімкнення мультиварки!\n";

}

void Multicooker::modeButton()

{

this->controlSystem.selectMode();

}

bool Multicooker::coverInfo(int m)

{

if (m == 0)

this->cookingContainer.chCover(TRUE);

else if (m == 1)

return this->cookingContainer.chCover(FALSE);

else if (m == 2)

this->securitySystem.checkBlock(TRUE);

else

return this->securitySystem.checkBlock(FALSE);

}

bool Multicooker::isCooking()

{

return controlSystem.checkButtons();

}

void Multicooker::cooking()

{

this->securitySystem.blockCover();

Sleep(1000);

this->controlSystem.makeSysToCook();

Sleep(1000);

this->condensateSystem.setState();

Sleep(1000);

this->heatingSystem.setState();

cout << "Страва готується, зачекайте...\n";

Sleep(5000);

}

void Multicooker::finishCooking()

{

this->controlSystem.finish();

Sleep(1000);

this->heatingSystem.setState();

Sleep(1000);

this->condensateSystem.setState();

Sleep(1000);

this->securitySystem.blockCover();

Sleep(1000);

this->controlSystem.sound();

}

bool Multicooker::chPower(bool p)

{

if (p == FALSE)

return this->powerSupply.energySupply();

else

{

this->powerSupply.setPower();

return TRUE;

}

}

bool Multicooker::getPower()

{

return this->power.getState();

}

Human::Human()

{

name = "user";

products = FALSE;

}

void Human::pushOnPowerButton(Multicooker\* mt)

{

mt->turnOn();

}

void Human::pushOnModeButton(Multicooker\* mt)

{

mt->modeButton();

}

void Human::openOrCloseCover(Multicooker\* mt)

{

mt->coverInfo(0);

}

void Human::pushOnStartButton(Multicooker\* mt)

{

mt->cooking();

}

void Human::pushOnStopButton(Multicooker\* mt)

{

mt->finishCooking();

}

bool Human::checkProducts(bool m)

{

if (m == TRUE)

{

if (this->products == FALSE)

{

this->products = TRUE;

cout << "Продукти покладено у мультиварку\n";

}

else

{

this->products = FALSE;

cout << "Страву вийнято\n";

}

}

else

return this->products;

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

Human hn;

Multicooker mt;

cout << "-------СИМУЛЯТОР МУЛЬТИВАРКИ--------\n";

int a = 0;

while (TRUE)

{

cout << "\nДоступні дії:\n";

cout << "1 - Змінити розетку, у яку увімкнено мультиварку\n";

cout << "2 - Увімкнути мультиварку\n";

cout << "3 - Відкрити кришку мультиварки\n";

cout << "4 - Покласти їжу у мультварку\n";

cout << "5 - Натиснути на кнопку 'вибір режиму'\n";

cout << "6 - Закрити кришку мультиварки\n";

cout << "7 - Натиснути кнопку 'старт'\n";

cout << "8 - Натиснути кнопку 'стоп'\n";

cout << "9 - Забрати їжу\n";

cout << "10 - Вимкнути мультиварку\n";

cout << "0 - Завершити виконання програми\n";

cout << "Команда >> ";

cin >> a;

cout << endl;

switch (a)

{

case 1:

{

if (mt.getPower() == FALSE)

mt.chPower(TRUE);

else

cout << "Не можна вимикати мультиварку із розетки, коли вона працює!\n";

}

break;

case 2:

{

if (mt.getPower() == TRUE)

cout << "Мультиварка вже увімкнена...\n";

else

hn.pushOnPowerButton(&mt);

}

break;

case 3:

{

if (mt.coverInfo(1) == FALSE)

{

if (mt.coverInfo(3) == FALSE)

{

mt.coverInfo(0);

}

else

cout << "Неможливо відкрити кришку під час приготування їжі!\n";

}

else

cout << "Кришку мультиварки вже відкрито...\n";

}

break;

case 4:

{

if (hn.checkProducts(FALSE) == TRUE)

cout << "Нічого класти... Продукти вже завантажені у мультиварку\n";

else if (mt.coverInfo(1) == FALSE)

cout << "Спочатку треба відкрити кришку...\n";

else

hn.checkProducts(TRUE);

}

break;

case 5:

{

if (mt.getPower() == FALSE)

cout << "Мультиварку вимкнено, кнопка вибору режиму деактивована...\n";

else if (mt.isCooking() == TRUE)

cout << "Неможливо змінити режим під час процесу приготування їжі...\n";

else

hn.pushOnModeButton(&mt);

}

break;

case 6:

{

if (mt.coverInfo(1) == TRUE)

mt.coverInfo(0);

else

cout << "Кришку мультиварки вже закрито...\n";

}

break;

case 7:

{

if (mt.getPower() == FALSE)

cout << "Неможливо запустити процес приготування їжі, коли мультиварку вимкнено!\n";

else if (hn.checkProducts(FALSE) == FALSE)

cout << "Нічого готувати: мультиварка порожня!\n";

else if (mt.coverInfo(1) == TRUE)

cout << "Неможливо запустити процес приготування їжі, коли кришку відкрито!\n";

else if (mt.isCooking() == TRUE)

cout << "Процес приготування їжі вже запущено!\n";

else

hn.pushOnStartButton(&mt);

}

break;

case 8:

{

if (mt.isCooking() == TRUE)

hn.pushOnStopButton(&mt);

else

cout << "Наразі мультиварка нічого не готує!\n";

}

break;

case 9:

{

if (mt.coverInfo(1) == FALSE)

cout << "Спочатку треба відкрити кришку...\n";

else if (hn.checkProducts(FALSE) == FALSE)

cout << "Мультиварка порожня...\n";

else

hn.checkProducts(TRUE);

}

break;

case 10:

{

if (mt.getPower() == TRUE)

{

if (mt.isCooking() == TRUE)

cout << "Неможливо вимкнути мультиварку, коли запущений процес приготування їжі!\n";

else

hn.pushOnPowerButton(&mt);

}

else

cout << "Мультиварку вже вимкнено\n";

}

break;

case 0:

return 0;

break;

}

}

}