Лабораторная работа №2

По дисциплине «Информатика»

Тема: Имитационное моделирование

Выполнил:

студент группы [EMPTY]

[YOUR NAME]

Принял:

[TEACHER NAME]

**Цель работы**: приобрести навыки в определении класса, инкапсулирующего объект из реального мира.

Пример работы с кондиционером

**Структура программы**

* deps.h - содержит макросы, что позволяет программе работать как на Windows так и на Unix-подобных платформах
* deps.cpp - содержит определения Windows-зависимых функций
* aircon.h - содержит объявления функций по управлению
* aircon.cpp - содержит определения функций по управлению

**Ниже представлен код программы**

// deps.h

#ifndef \_\_DEPS\_H

#define \_\_DEPS\_H

# ifdef \_\_WIN32

# include <windows.h>

# include <conio.h>

# define KB\_ENTER 13

# else

# include <unistd.h>

# include <termios.h>

# include <sys/ioctl.h>

# define KB\_ENTER 10

int putch (int c);

int getch (void);

int kbhit (void);

# endif

#define KB\_ESCAPE 27

#endif

// deps.cpp

#include <cstdio>

#include "deps.h"

#ifndef \_\_WIN32

int putch (int c)

{

return putchar (c);

}

int kbhit(void) {

static const int STDIN = 0;

static bool initialized = false;

if (! initialized) {

// Use termios to turn off line buffering

termios term;

tcgetattr(STDIN, &term);

term.c\_lflag &= ~ICANON;

tcsetattr(STDIN, TCSANOW, &term);

setbuf(stdin, NULL);

initialized = true;

}

int bytesWaiting;

ioctl(STDIN, FIONREAD, &bytesWaiting);

return bytesWaiting;

}

int getch (void)

{

int ch;

struct termios oldtc;

struct termios newtc;

tcgetattr(STDIN\_FILENO, &oldtc);

newtc = oldtc;

newtc.c\_lflag &= ~(ICANON | ECHO);

tcsetattr(STDIN\_FILENO, TCSANOW, &newtc);

ch=getchar();

tcsetattr(STDIN\_FILENO, TCSANOW, &oldtc);

return ch;

}

#endif

// aircon.h

#ifndef \_\_AIRCON\_H

#define \_\_AIRCON\_H

enum state {

on,

off

}; // определение типа для элемента, поддерживающего состояние включён/выключен

enum mode {

m\_hot,

m\_cold

}; // определение режима работы кондиционера

class Aircon {

state curr\_button; // поле для описания состояния кнопки

mode curr\_mode; // поле для описания режима работы

state curr\_power; // поле для описания состояния питания

int time; // время, оставшееся до конца работы кондиционера

void turn\_power(void); // метод, изменяющий положение двери

void change\_mode (void); // метод, изменяющий режим работы

void push\_button(void); // метод, обрабатывающий работы с питанием микроволновой печи

void deactivate(void); // метод, обрабатывающий окончание работы кондиционера

void activate(void);

public:

Aircon(); // конструктор, задёт начальные параметры

void Run(void); // метод, обрабатывающий взаимодействие пользователя с кондиционером

};

#endif

// aitcon.cpp

#include <iostream>

#include <cstdio>

#include "aircon.h"

#include "deps.h"

using namespace std;

#if defined (\_\_WIN32)

# define SLEEP(sec) Sleep ((sec) \* 100) // потому что Windows Sleep примимает милисекунды

#else

# define SLEEP(sec) sleep ((sec)) // a Unix-like секунды

#endif

#define TM\_PERIOD 60 /\* промежуток работы кондиционера 60 сек \*/

#define TM\_OFF -1 /\* счётчик сброшен \*/

/\* Кондиционер имеет 3 кнопки:

1) кнопка режима (холодный/горячий воздух);

2) кнопка питания;

3) кнопка работы.

\*/

void menu (void) //выводит на экран меню

{

cout << "b - кнопка работы\n"; // запускает кондиционер на 1 минуту повторное нажатие добавляет еще минуту, если включен

cout << "p - вкл/выкл кондиционер\n"; // питание

cout << "m - холодный/горячий воздух\n";

cout << "ESC - выход\n\n";

}

Aircon::Aircon()

{

// начальное состояние:

curr\_button = off; // кнопка не нажата

curr\_mode = m\_cold; // кондиционер в холодном режиме

curr\_power = off; // питание выключено

time = TM\_OFF; // кондиционер не работет

}

void Aircon::turn\_power(void)

{

if (curr\_power == off) {

curr\_power = on;

cout << "Кондиционер включён\n";

}

else {

/\* если есть питание, то сбрасываем все и выкл кондиционер. \*/

curr\_power = off;

curr\_button = off;

time = TM\_OFF;

cout << "Кондиционер выключен\n";

}

cout << "\n";

}

void Aircon::change\_mode(void)

{

if (curr\_power == on) {

cout << "Выключи питание чтобы поменять режим!\n\n";

return ;

}

if (curr\_mode == m\_cold) {

curr\_mode = m\_hot;

cout << "Режим изменён на горячий\n";

}

else if (curr\_mode == m\_hot) {

curr\_mode = m\_cold;

cout << "Режим изменён на холодный\n";

}

cout << "\n";

}

void Aircon::push\_button(void)

{

if (curr\_power == off) {

cout << "Включи питание перед началом работы!\n\n";

return ;

}

if (curr\_button == off) {

curr\_button = on;

time = TM\_PERIOD;

cout << "Кондиционер работает\n";

}

else {

time += TM\_PERIOD;

cout << "Вам добавлена минута\n";

}

}

void Aircon::deactivate(void)

{

curr\_power = off; // питание выключено

curr\_button = off; // лампа выключена

time = -1; // сбрасываем счётчик времени

cout << "Кондиционер выключен\n\n";

}

void Aircon::Run(void)

{

int key;

menu();

do {

while (!kbhit() && time > 0) //в этот цикл заходим, если еда готовится; условие выхода из цикла: нажата клавиша или истекло время приготовления еды

//функция kbhit() возвращает истину, если пользователь нажал клавишу

{

time--; // вычитаем секунду из времени, оставшегося до конца приготовления еды

SLEEP(1); // задержка в 100 мс

cout << "Осталось " << time << " секунд до выключения\n";

}

if (time==0) { // если кондиционер закончила работу, вызываем метод deactivate()

deactivate();

}

key = getch();

switch (key) {

case 'b':

push\_button();

break;

case 'p':

turn\_power();

break;

case 'm':

change\_mode();

break;

}

fflush(stdin);

} while(key != KB\_ESCAPE);

}

// main.cpp

#include <clocale>

#include "aircon.h"

#include "deps.h"

int main()

{

Aircon Myaircon;

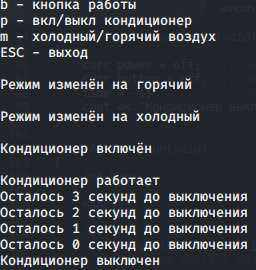
setlocale(0, "RUS");

Myaircon.Run();

return 0;

}

Экранные формы с примерами работы программы



Pис. 1

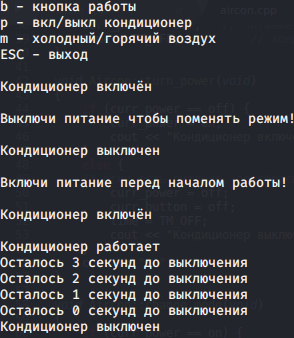


Рис. 2

Рис. 1 отображает меню управления и выполнение правильной последовательности команд для работы виртуального кондиционера. Перед тем как включить устройство можно изменить режим на горячий или холодный, после включить питание и запустить сам кондиционер. После 5 сек. кондиционер выключится сам по себе. Таймер намеренно уменьшен, что правильно отобразить логи.

На Рис. 2 отображен механизм обработки ошибок и выведение предупредительного сообщения пользователю. Первая ошибка сообщает что нужно выключить питание перед тем как поменять режим. Вторая говорит о том что пользователь забыл включить питание.

Заключение

В данной лабораторной работе я научился инкапсулировать объекты из реального мира на примере воздушного кондиционера, использую язык программирования С++ и его классы.