

1. Цель и задачи

Цель: познакомиться с паттернами программирования.

Задачи:

1. Изучить несколько паттернов программирования.
2. Научиться применять паттерны для решения классических ситуаций.
3. Задание

Общее ограничение на лабораторную работу:

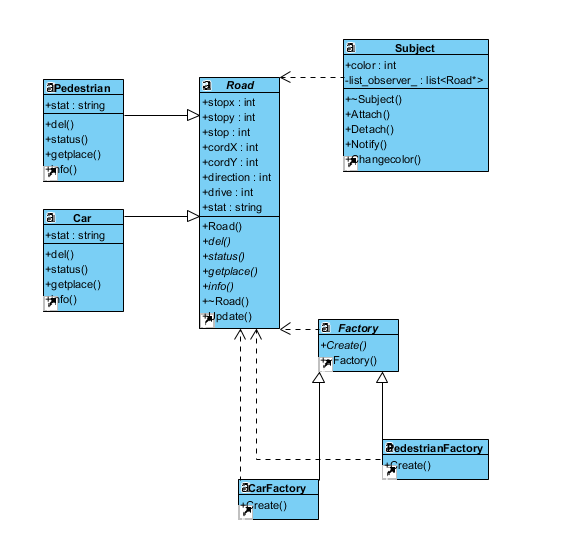
• использовать разбитие на файлы основной программы (в main.cpp только функция main.cpp);

• документировать весь код используя аннотации Doxygen.

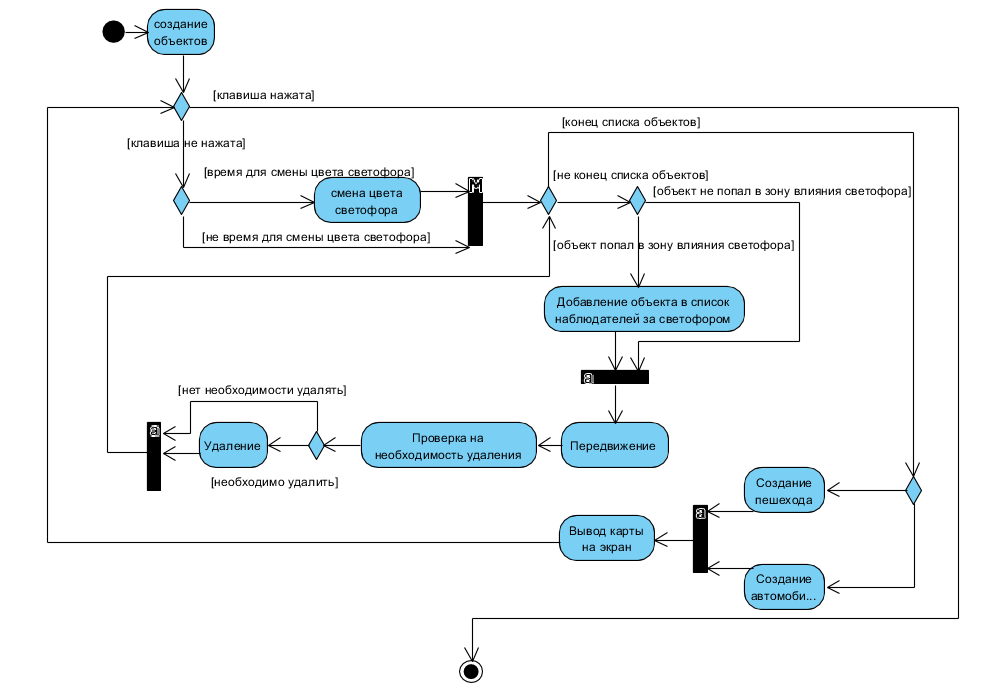
Требования к лабораторной (Вариант 2):

* перекрёсток должен быть нарисован и не двигаться по мере выполнения программы (т. е. для наблюдателя перекрёсток статичен);
* машины появляются только на нужных полосах и двигаются от одного края дороги, до другого и удаляются из программы;
* пешеходы появляются на границах пешеходного перехода и двигаются до границ пешеходного перехода и удаляются из программы;
* базовое условное поле с которым можно работать: 80 столбцов, 40 строк;
* раз в 1 секунду должно происходить движение всех доступных объектов;
* скорость машины: 2 деления, скорость пешехода: 1 деление;
* машины и пешеходы и время их появления должны быть заданы в момент запуска программы (кодом).

1. ДИАГРАММЫ



*Рисунок 1 – диаграмма классов*

**

*Рисунок 2 – диаграмма деятельности функции main*

1. Код программы

Вариант 4. Основной выполняемый файл main.cpp:

/\*\*

\* @file main.cpp

\* @author Oreshnikov Egor

\* @brief Содержит меню программы

\*/

#include <iostream>

#include <windows.h>

#include <stdio.h>

#include <conio.h>

#include "fac.h"

#include "head.h"

#include <stdlib.h>

#include <time.h>

#define n 10

using namespace std;

int main()

{

Subject\* svetofor = new Subject;

PedestrianFactory pedestrian;

CarFactory car;

int av[n]={0};

Factory \*ptr1 = &pedestrian;

Factory \*ptr2 = &car;

Road \*ObjectRoad[n]={0};

int c,i,num,d;

i=0; c=0; num=0;

while(!\_kbhit())

{

if (c==4){

svetofor->Changecolor();

c=0;

}

for (int i2=0; i2<n; i2++)

{

if (av[i2]==1)

{

if ((ObjectRoad[i2]->cordX==ObjectRoad[i2]->stopx)||(ObjectRoad[i2]->cordY==ObjectRoad[i2]->stopy))

{svetofor->Attach(ObjectRoad[i2]); ObjectRoad[i2]->drive=svetofor->color;

}

bye(ObjectRoad[i2]);

movel(ObjectRoad[i2]);

d=ObjectRoad[i2]->del();

if (d==-1) {delete ObjectRoad[i2]; av[i2]=0;}

}

}

srand(time(NULL));

int typ=0+rand()%2;

if (c==3){

if (num>=n)

{ num=0;

delete ObjectRoad[num];

av[num]=0;}

if (typ==0){

ObjectRoad[num] = foo(ptr1); av[num]=1;

}

if (typ==1){

ObjectRoad[num] = foo(ptr2); av[num]=1;

}

ObjectRoad[num]->getplace();

num++;}

mapaful(av,ObjectRoad,svetofor);

c=c+1;

Sleep(1000);

system("cls");

i++;

}

return 0;}

Файл с alg.cpp:

/\*\*

\* @file alg.cpp

\* @author Oreshnikov Egor

\* @brief Содержит выполняемый алгоритм

\*/

#include <iostream>

#include <conio.h>

#include <stdlib.h>

#include <time.h>

#include "obs.h"

#define n 10

using namespace std;

/\*\*

\* Функция вывода на экран

\* @param av существующие объекты

\* @param ObjectRoad[n] объекты

\* @param svetofor отслеживаемый объект

\*/

void mapaful( int av[n], Road\* ObjectRoad[n], Subject\* svetofor ) {

int x,y,m,p;

for (int j=1; j<40; j++){

y=j;

for (int i=1; i<80; i++)

{ x=i; m=0; p=0;

if ((x==15)&&(y==10))

{

if (svetofor->color==1) {cout<<"GOR";}

else {cout<<"VER";}

i=i+3; x=i;

}

for (int num=0; num<n; num++)

{

if (av[num]==1){

if ((ObjectRoad[num]->cordX==x)&&(ObjectRoad[num]->cordY==y)&&(ObjectRoad[num]->status()=="car"))

{m=1;}

else if ((ObjectRoad[num]->cordX==x)&&(ObjectRoad[num]->cordY==y)&&(ObjectRoad[num]->status()=="pedestrian"))

{p=1;}}}

if (m==1) {cout<<"#";} else if (p==1){cout<<"$";}

else{

if (((x==35)||(x==45))&&((y<=14)||(y>=26)))

{cout<<"|";}

else if (((y==15)||(y==25))&&((x<=35)||(x>=46)))

{cout<<"\_";}

else cout<<" "; }

}

cout<<endl;}

}

/\*\*

\* Функция передвижения объектов

\* @param r объект

\*/

void movel(Road\* r) {

if(((r->direction==1)&&(r->drive==-1))||((r->direction==1)&&(r->stop==1)))r->cordY=r->cordY+1;

else if(((r->direction==2)&&(r->drive==-1))||((r->direction==2)&&(r->stop==1)))r->cordY=r->cordY-1;

else if(((r->direction==3)&&(r->drive==1))||((r->direction==3)&&(r->stop==1)))r->cordX=r->cordX+1;

else if(((r->direction==4)&&(r->drive==1))||((r->direction==4)&&(r->stop==1)))r->cordX=r->cordX-1;

}

/\*\*

\* Функция проверки необходимости слежения за светофором

\* @param r объект

\*/

void bye(Road\* r)

{

if (((r->cordX==r->stopx+3)&&(r->direction==3))||((r->cordY==r->stopy+3)&&(r->direction==1))||((r->cordY==r->stopy-3)&&(r->direction==2))||((r->cordX==r->stopx-3)&&(r->direction==4)))

r->stop=1;

}

Файл с head.h:

/\*\*

\* @file obs.h

\* @author Oreshnikov Egor

\* @brief Содержит описание

\*/

#ifndef head

#define head

#include <iostream>

#include <string>

#define n 10

class Subject;

class Road;

/\*\*

\* Функция вывода на экран

\* @param av существующие объекты

\* @param ObjectRoad[n] объекты

\* @param svetofor отслеживаемый объект

\*/

void mapaful( int av[n], Road\* ObjectRoad[n], Subject\* svetofor );

#endif

Файл с obs.h:

/\*\*

\* @file obs.h

\* @author Oreshnikov Egor

\* @brief Содержит классы

\*/

#ifndef obs

#define obs

#include <iostream>

#include <list>

#include <string>

class Subject;

class Road{

friend void movel(Road \*r);

friend void bye(Road\* r);

public:

it stopx; int stopy; int stop;

int cordX;

int cordY;

int direction;

int drive;

Road(){stop=0;}

virtual int del()=0;

std::string stat;

virtual std::string status()=0;

virtual void getplace()=0;

virtual void info() = 0;

virtual ~Road() {}

void Update(const int &color\_from\_subject) {

drive = color\_from\_subject;}

};

class Subject {

public:

int color=1;

virtual ~Subject() {

}

/\*\*

\* Методы управления подпиской.

\*/

void Attach(Road\* observer) {

list\_observer\_.push\_back(observer);

}

void Detach(Road\* observer) {

list\_observer\_.remove(observer);

}

void Notify() {

std::list<Road \*>::iterator iterator = list\_observer\_.begin();

while (iterator != list\_observer\_.end()) {

(\*iterator)->Update(color);

++iterator;}

}

void Changecolor() {

this->color = color\*(-1);

Notify();

}

private:

std::list<Road \*> list\_observer\_;};

#endif

Файл с ped.h:

/\*\*

\* @file ped.h

\* @author Oreshnikov Egor

\* @brief Содержит дочерний класс-пешеход

\*/

#pragma once

#include <string>

#include <conio.h>

#include <stdlib.h>

#include <time.h>

#include <iostream>

#include "obs.h"

using namespace std;

class Pedestrian:public Road{

public:

string stat="pedestrian";

int del() override

{

if ((cordX==47)||(cordX==33)||(cordY==13)||(cordY==27))

return (-1);

else return (0);

}

string status() override

{

return stat;

}

virtual void getplace() override{

int cor=1+rand()%8;

if (cor==1) {cordX=34; cordY=14; direction=1;drive=-1; }

if (cor==2) {cordX=34; cordY=14; direction=3;drive=1;}

if (cor==3) {cordX=46; cordY=14; direction=1;drive=-1;}

if (cor==4) {cordX=46; cordY=14; direction=4;drive=1;}

if (cor==5) {cordX=34; cordY=26; direction=2;drive=-1;}

if (cor==6) {cordX=34; cordY=26; direction=3;drive=1;}

if (cor==7) {cordX=46; cordY=26; direction=4;drive=1;}

if (cor==8) {cordX=46; cordY=26; direction=2;drive=-1;}

stopx=cordX; stopy=cordY;

}

void info(){

cout << "Pedestrian appeared\n"<< " x="<<cordX<<" y="<<cordY<<" dir="<<direction<<endl; //<-- Информация на экран для наглядности

}};

Файл с car.h:

/\*\*

\* @file car.h

\* @author Oreshnikov Egor

\* @brief Содержит дочерний класс-машину

\*/

#ifndef binarySearch

#define binarySearch

#include <string>

#include <conio.h>

#include <stdlib.h>

#include <time.h>

#include <iostream>

#include "obs.h"

class Car:public Road{

public:

string stat="car";

int del() override

{

if ((cordX==0)||(cordX==80)||(cordY==0)||(cordY==40))

return (-1);

}

string status() override

{

return stat;

}

virtual void getplace() override{

int cor=1+rand()%4;

if (cor==1) {cordX=1; cordY=18; direction=3; drive=1;stopx=32;}

if (cor==2) {cordX=79; cordY=22; direction=4; drive=1;int stopx=48;}

if (cor==3) {cordX=38; cordY=1; direction=1;drive=-1;int stopy=12;}

if (cor==4) {cordX=42; cordY=39; direction=2;drive=-1;stopy=28;}

}

void info(){

cout<<stat << " appeared\n"<< " x="<<cordX<<" y="<<cordY<<" dir="<<direction<<endl;//<-- Информация на экран для наглядности

}

};

#endif

Файл с fac.h:

/\*\*

\* @file obs.h

\* @author Oreshnikov Egor

\* @brief Содержит фабрику

\*/

#pragma once

#include <string>

#include <conio.h>

#include <stdlib.h>

#include <time.h>

#include <iostream>

#include "ped.h"

#include "car.h"

class Factory{

public:

virtual Road\* Create() = 0;

virtual ~Factory(){}

};

class PedestrianFactory:public Factory{

public:

Road\* Create(){

return new Pedestrian;

}

};

class CarFactory:public Factory{

public:

Road\* Create(){

return new Car;}

};

/\*\*

\* Функция создания

\* @param value объект

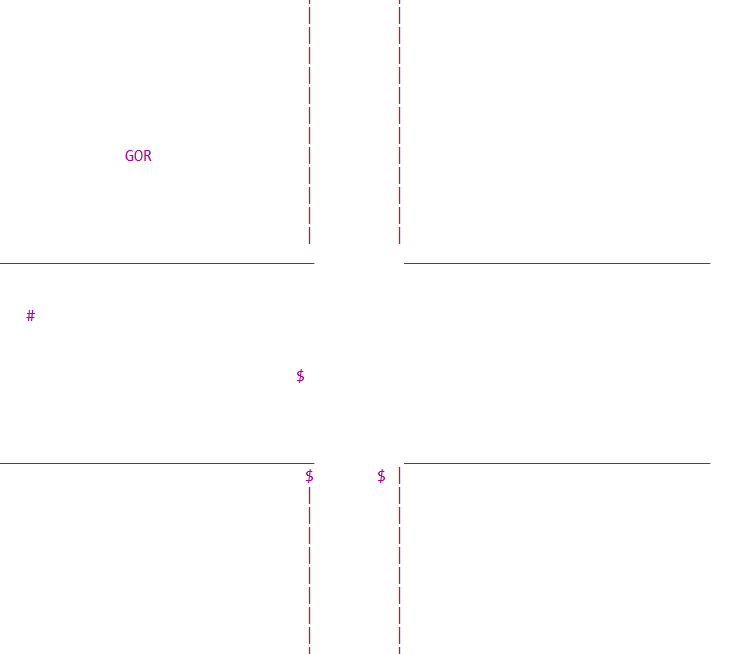
\*/

Road\* foo(Factory \*value){

return value->Create();

}

В качестве результата работы программы представлен скриншот выходного файла на рисунке 4.



*Рисунок 4 – выполнение программы*

# ВЫВОД

В процессе выполнения лабораторной работы были получены навыки работы с паттернами.