Міністерство освіти і науки України

Національний технічний університет України

“Київський політехнічний інститут”

Кафедра АСОІУ

**ЗВІТ**

про виконання комп’ютерного практикуму № 2

з дисципліни

“ООП”

Тема: «Наслідування С++»

|  |  |  |
| --- | --- | --- |
| Прийняв: |  | Виконав: |
| Головченко  Максим  Миколайович |  | студент 2-го курсу  гр. ІП-51 ФІОТ  Субцельний Олександр Володимирович |

Київ – 2016

**ЗМІСТ**

[1 Мета роботи 4](#_Toc463208490)

[2 Постановка задачі (варіант 7, рівень В) 5](#_Toc463208491)

[3 Покроковий алгоритм 6](#_Toc463208492)

[4 Діаграма класів 7](#_Toc463208493)

[5 Код програми 8](#_Toc463208494)

[6 Приклади виконання програми 12](#_Toc463208495)

[7 Висновок 13](#_Toc463208496)

# Мета роботи

Мета роботи - вивчити основні концепції об'єктно-орієнтованого програмування. Вивчити особливості успадкування і множинного успадкування.

# Постановка задачі (варіант 7, рівень В)

Определить базовый класс «Кости домино», в котором хранится некоторое количество костей в виде «a/b». В дочернем классе определить метод для правильного упорядочивания костей, или вывести сообщение о невозможности выполнения такого упорядочивания

# Покроковий алгоритм

* Вирішення правдоподібності задачі

1. ПОЧАТОК
2. Підрахувати кількість кожного ребра з однаковою кількістю точок
3. Якщо рнепарна кількість ребер з орднаковими точками більше 2 :
   1. Неможливо скласти доміно
4. Інакше:

4.1 Взяти за перше доміно з ребром, яке входить у склад непарниої кількості ребер

4.2 Взяти за останнє доміно з ребром, яке входить у склад непарниої кількості ребер

1. КІНЕЦЬ

* Знаходження маршруту доміно

1. ПОЧАТОК
2. У циклі від 1 до кількості доміно:
3. Якщо кількість виклику рекурсії дорівнює кількості доміно, записати у кінцевий маршрут поточний маршрут

3.1Якщо у доміно є ребро з підходящим номером точок:

3.1.1Додати доміно у масив поточного маршруту

3.1.2Задати підходящим ребром другу половину доміно

3.13Рекурсивно викликати функцію знаходження маршруту

3.2Видалити останнє доміно з поточного маршруту

1. КІНЕЦЬ

# Діаграма класів

Діаграма класів наведена на рисунку 4.1:

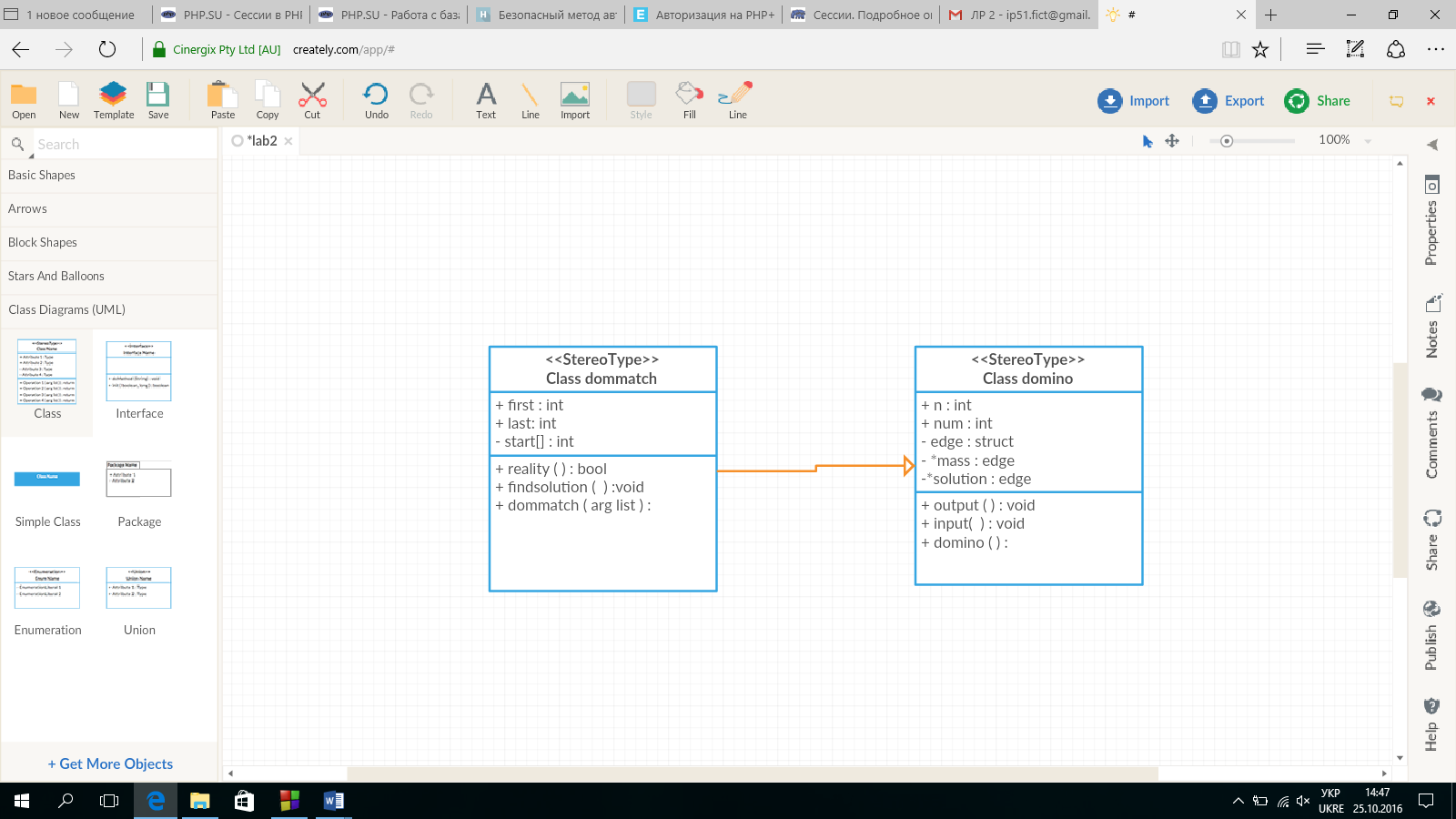


Рисунок 4.1 – Діаграма класів

# Код програми

“main.cpp”

#include <iostream>

#include "domino.h"

#include "dommatch.h"

using namespace std;

int main()

{

cout << "Subtselnyi Oleksandr Volodymyrovych" << endl;

cout << "V : 20" << endl;

cout << "L : B" << endl << endl;

char h;

while (h!='s'){

dommatch\* jbj = new dommatch(); //створення обэкта

jbj->output();//вивести згенеровані конструктором елементи

jbj->input();//моєжливість ввести свої дані

if (jbj->reality()){ //перевірка на реальність

cout<<"maybe possible"<<endl;

jbj->findsolution();//знайти шлях

}

else cout<<"NO!"<<endl;

delete jbj; //видалите обєкт

cout<<"if you want to stop press s";

cin>>h;

}

}

“Domino.h”

#ifndef DOMINO\_H

#define DOMINO\_H

#include <stdlib.h>

#include <stdio.h>

#include <iostream>

#include <ctime>

using namespace std;

class domino

{

public:

int n; //кількість доміно

int num=0; //лічильник

void output(); //вивести

void input(); //вести

domino(); //конструктор

virtual ~domino(); //деструктор

protected:

struct edge //доміношні ребра

{

int a,b;

bool mark;

};

edge \*mass; //масив з доміно

edge \*solution; //маршрут

edge \*solution1;

private:

};

#endif // DOMINO\_H

“Domino.cpp”

#include "domino.h"

domino::domino() //конструтор, рандомне генерування

{

n=10;

srand( time(NULL) );

mass=new edge[n];

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

mass[i].a = rand()%6+1;

mass[i].b = rand()%6+1;

mass[i].mark=true;

}

}

domino::~domino() //деструтор

{

delete [] mass;

delete [] solution;

delete [] solution1;

}

void domino::output() //виведення доміно

{

cout<<"number of domino's ="<<n<<endl;

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

cout<<"DOMINOSHKA "<<i+1<<" EDGE a="<<mass[i].a<<" EDGE b="<<mass[i].b<<endl;

}

}

void domino::input()//можливість ввести свої занчення

{

char c;

cout<<"do you want to leave ctor values?[y/n]"<<endl;

cin>>c;

if (c=='y'||c=='Y')

return;

else {

cout<<"enter number of domino's"<<endl;

cin>>n;

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

{

cout<<"Enter edge a for "<<i+1<<endl;

cin>>mass[i].a;

cout<<"Enter edge b for "<<i+1<<endl;

cin>>mass[i].b;

}

}

}

“Dommatch.h”

#ifndef DOMMATCH\_H

#define DOMMATCH\_H

#include <domino.h>

class dommatch : public domino //агрегація

{

public:

bool reality(); //перевірка на реальність

void findsolution(); //знаходження маршруту

bool findsolution2(int);

dommatch(); //конструктор

virtual ~dommatch(); //деструктор

protected:

private:

int start[6]; //допоміжний масив

int first, //перше доміно

last; //останнє доміно

};

#endif // DOMMATCH\_H

“Dommatch.cpp”

#include "dommatch.h"

dommatch::dommatch() //конструтор

{

n=10;

solution=new edge[n];

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

solution[i].a=0;

solution[i].b=0;

}

/\* solution1=new edge[n];

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

solution1[i].a=0;

solution1[i].b=0;

}\*/

}

dommatch::~dommatch() /деструтор

{

delete [] solution;

//delete [] solution1;

}

bool dommatch::reality() //перевірка на реальність

{

int l=0;

int massiv[6];

int cc=0;

for (int i=0;i<6;i++) //створення допоміжного масиву

massiv[i]=0;

for (int i=0;i<n;i++) //знаходження кількості непарних ребер

{

massiv[mass[i].a]++;

massiv[mass[i].b]++;

}

for (int i=1;i<=6;i++) //перевірка на кількість непарних ребер

{

if (massiv[i]%2!=0){

cc++;

start[l++]=i;

}

}

if (cc>2||cc==1) return false; //повернути false якщо кількість нерарних ребер більше 2

for (int i=0;i<n;i++){ //задати перший елемент

if (mass[i].a==start[1]){

first=mass[i].b;

mass[i].mark=false;

solution[num++]=mass[i];

break;

}

if (mass[i].b==start[1]){

first=mass[i].a;

mass[i].mark=false;

solution[num++]=mass[i];

break;

}

}

for (int i=0;i<n;i++){//задати останні елемент

if ((mass[i].a==start[0])&&(mass[i].a!=first)&&mass[i].mark){

last=mass[i].b;

mass[i].mark=false;

solution[n-1]=mass[i];

break;

}

if ((mass[i].b==start[0])&&(mass[i].b!=first)&&mass[i].mark){

last=mass[i].a;

mass[i].mark=false;

solution[n-1]=mass[i];

break;

}

}

return true;

}

void dommatch::findsolution() //знайти маршрут

{

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

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

{

if (num==n) continue;

if (mass[i].mark&&(mass[i].a==first))//якщо співпадають ребра

{

first=mass[i].b;//якщо співпадають ребра

mass[i].mark=false;

solution[num++]=mass[i]; //додати до маршуруту

}

if (mass[i].mark&&(mass[i].b==first))//якщо співпадають ребра

{

first=mass[i].a;

mass[i].mark=false;

solution[num++]=mass[i];//додати до маршруту

}

}

}

if (num!=n-1){cout<<"NO WAY"; return;}

for (int i=0;i<n;i++) //вивести маршгрут

{

cout<<solution[i].a<<" "<<solution[i].b<<endl;

}

}

# Приклади виконання програми

Приклад виконання програми наведений на рисунку 6.1:

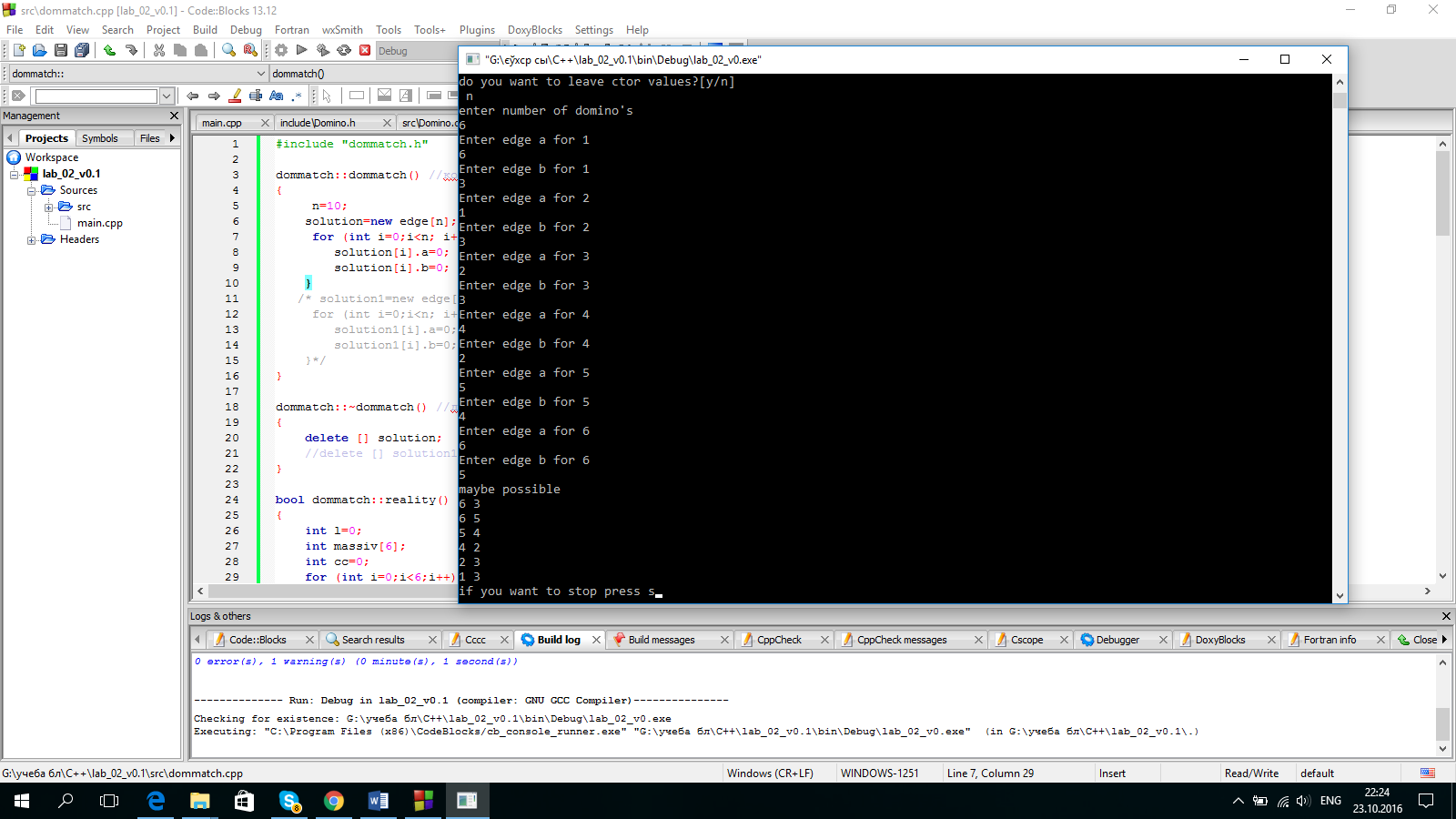


Рисунок 6.1 – Приклад виконання програми

# Висновок

Отже, дана програма створює об’єкт класа dommatch, який наслідує класс domino, який шукає можливий маршрут складання доміно, які задані у класі domino. Оскільки дана програма реалізована за модульним принципом і кожен модуль програми створювався й налагоджувався автономно, то дана програма буде видавати очікуваний результат на всіх наборах вхідних даних.