Теоретичні питання Шевлюк Вікторія:

12. Чи має можливість клас-нащадок змінювати область видимості членів класу-предка?

Клас-нащадок не має можливості змінювати область видимості членів класу-предка.

26. Копіювання одновимірних масивів

В мові C# для копіювання одновимірних масивів може використовуватися метод Clone().

Втім, цей метод здійснює неглибоку копію масиву. Також існує статичний метод Copy() класу Array. На вхід цей метод приймає 3 аргументи: перший – масив, з якого потрібно скопіювати дані, другий – масив у який потрібно скопіювати дані, третій – кількість елементів для копіювання.

Також можна здійснити копіювання шляхом проходу масиву в циклі, присвюючи кожен його елемент іншому масиву.

Наприклад у нас є два масиви: int[] first = {1, 2, 3, 4, 5, 6 } та масив int[] second = new int[5]. Тоді для копіювання елементів масиву first у масив second використаємо цикл.

for(int i = 0; i < second.Length; i++) {

second[i] = first[i];

}

40. Чи можна змінній типу інтерфейс надавати значення об’єкта типу класу, в якому  
реалізовано цей інтерфейс?

Так, змінній типу інтерфейс можна надавати значення об’єкта типу класу, в якому реалізовано цей інтерфейс.

Circle circle = new Circle();

IRadius r = (IRadius) circle;

Console.WriteLine (r.GetNumberOfPoints());

**Практичне завдання:**

using System;

using System.Collections.Generic;

using System.Text;

namespace ConsoleApp2

{

class Graph

{

private int verNum;

private int[,] adjMatr;

private System.Random rand = new System.Random();

public Graph(int vN)

{

bool rand = vN == 1;

if (rand) vN = this.rand.Next(2, 21);

this.verNum = vN;

this.adjMatr = new int[vN, vN];

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

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

{if (i != j)

this.adjMatr[i, j] = -1;

else

this.adjMatr[i, j] = 0;

}}

if (rand)

{for (int i = vN \* vN / 2; i > 0; i--)

{this.addEdge(this.rand.Next(vN), this.rand.Next(vN), this.rand.Next(10), false);

}}}

public void addEdge(int ver1, int ver2, int weight, bool printI)

{if (ver1 >= this.verNum || ver2 >= this.verNum)

{if (printI) System.Console.WriteLine("Invalid input: vertex doesn't exist");

return;}

if (weight < 0)

{if (printI) System.Console.WriteLine("Invalid input: negative weight");

return;

}

if (ver1 == ver2)

{

if (printI) System.Console.WriteLine("Invalid input: entered the same vertex twice");

}

else if (this.adjMatr[ver1, ver2] != -1)

{

if (printI) System.Console.WriteLine("The edge between vertices " + ver1 + " & " + ver2 + " already exists");

}

else

{this.adjMatr[ver1, ver2] = weight;

this.adjMatr[ver2, ver1] = weight;

if (printI) System.Console.WriteLine("The edge between vertices " + ver1 + " & " + ver2 + " successfully created");

}}

public void deleteEdge(int ver1, int ver2)

{

this.adjMatr[ver1, ver2] = -1;

this.adjMatr[ver2, ver1] = -1;

System.Console.WriteLine("The edge is deleted!");

public void showVN()

{

System.Console.WriteLine("Vertices number: " + this.verNum);

}

public void printMatr()

{

System.Console.WriteLine();

for (int i = 0; i < this.verNum; i++)

{for (int j = 0; j < this.verNum; j++)

{System.Console.Write(this.adjMatr[i, j] + "\t");

}

System.Console.WriteLine();

}}

public void showEdge(int ver1, int ver2)

{System.Console.WriteLine("Weight of this vertix is " + this.adjMatr[ver1, ver2]);}

public void showAllEdg()

{bool isshown = false;

for (int i = 0; i < this.verNum; i++)

{

for (int j = i + 1; j < this.verNum; j++)

{

if (this.adjMatr[i, j] != -1)

{

showEdge(i, j);

isshown = true;

}}}

if (!isshown)

System.Console.WriteLine("There is no such edge!");}

public void printShortestPath(int ver1, int ver2)

{

int curVert = -1, minDistance, step = 1;

int[] dists = new int[this.verNum];

int[] paths = new int[this.verNum];

bool[] isl = new bool[this.verNum]

for (int i = 0; i < this.verNum; i++)

{

dists[i] = System.Int32.MaxValue;

paths[i] = -1;

isl[i] = false;}

dists[ver1] = 0”

while (!isl[ver2])

{minDistance = System.Int32.MaxValue;

for (int i = 0; i < this.verNum; i++)

{

if (!isl[i] && minDistance >= dists[i])

{minDistance = dists[i];

curVert = i;  
}}

isl[curVert] = true;

for (int i = 0; i < this.verNum; i++)

{

if (

!isl[i] &&

this.adjMatr[curVert, i] != -1 &&

dists[curVert] != System.Int32.MaxValue &&

dists[i] > dists[curVert] + this.adjMatr[curVert, i]

) dists[i] = dists[curVert] + this.adjMatr[curVert, i];}}

paths[0] = ver2;

curVert = ver2;

while (isl[ver1])

{

isl[curVert] = false;

for (int i = 0; i < this.verNum; i++)

{

if (isl[i] &&

this.adjMatr[curVert, i] != -1 &&

dists[i] == dists[curVert] - this.adjMatr[curVert, i])

{

paths[step] = i;

step++;

curVert = i;

break;}}}

System.Console.WriteLine("The shortest path is " + ver1 + " & " + ver2 + " is: " + dists[ver2] );

for (int i = 1; i < this.verNum; i++)

{

if (paths[i] == -1) break;

System.Console.Write("<-" + paths[i]);

}}}}

namespace ConsoleApp2

{

class Program

{public static int inpMin(string txt, int min)

{int outp = -1;

string inp;

System.Console.WriteLine("\n" + txt);

inp = System.Console.ReadLine();

return outp;

}

static void Main(string[] args)

{Graph myGraph = new Graph(inpMin("Enter your vertices:", 1));

int ans = 0;

ans = inpMin("Enter:\n► 1 show vertex number,\n► 2 show matrix,\n► 3 show edge,\n► 4 show all edges," +

"\n► 5 to add edge,\n► 6 remove edge,\n► 7 find the shortest path,\n► 8 exit.", 1);

switch (ans)

{

case (1):

myGraph.showVN();

break;

case (2):

myGraph.showVN();

break;

case (3):

myGraph.showEdge(inpMin("input the first vertex index", 0), inpMin("input the second vertex index", 0));

break;

case (4):

myGraph.showAllEdg();

break;

case (5):

myGraph.addEdge(inpMin("input the first vertex index", 0), inpMin("input the second vertex index", 0), inpMin("Enter the weight", 0));

break;

case (6):

myGraph.deleteEdge(inpMin("input the first vertex index", 0), inpMin("input the second vertex index", 0));

break;

case (7):

myGraph.printShortestPath(inpMin("input the first vertex index", 0), inpMin("input the second vertex index", 0));

break;

case (8):

break;

}}}}}