**Міністерство освіти та науки України**

**Державний вищий навчальний заклад “Київський національний економічний університет ім. Вадима Гетьмана”**

**Кафедра інформаційних систем в економіці**

**Дисципліна “Адміністрування та програмування баз даних в штучному інтелекті”**

**ЗВІТ**

**з лабораторної роботи №1**

**«С# Starter»**

**Підготував:**

студент 5 курсу гр. ІШІ-501

спеціальності «8і09»

Донєв Данило Романович

**Прийняла:**

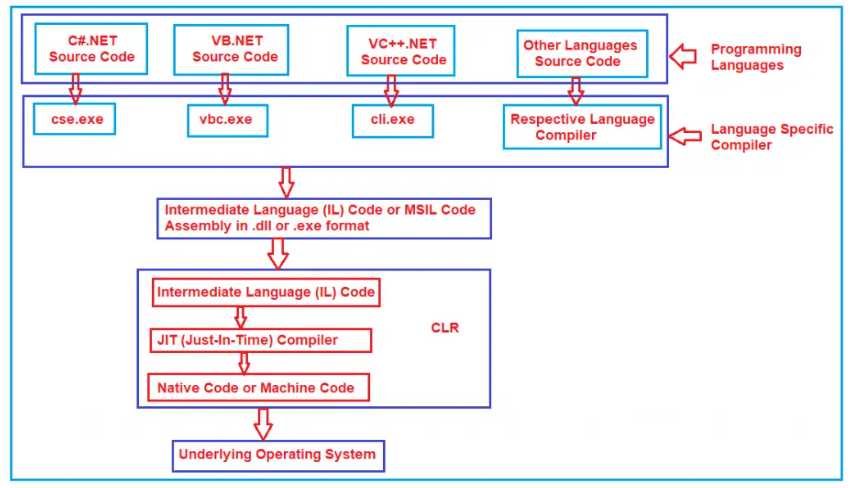
доц. Добролюбова М.В.

***ВІДПОВІДІ НА ПИТАННЯ***

1. **Пояснити схему виконання програми в .NET.**

У .NET виконання програми складається з 2 кроків. Вони такі:

На кроці 1 компілятор відповідної мови компілює вихідний код у проміжну мову (IL), а на другому кроці JIT-компілятор CLR перетворить код проміжної мови (IL) на машинний або двійковий код, який потім може виконуватися основною операційною системою. Описаний вище процес показаний на зображенні нижче.



Оскільки код .NET має формат Intermediate Language (IL), а не двійковий або машинний код, то дод .NET можна переносити на будь-яку платформу, якщо на цільовій платформі є Common Language Runtime (CLR). CLR цільової платформи перетворює код Intermediate Language у рідний код або машинний код, який може зрозуміти базова операційна система.

Код проміжної мови також називають керованим кодом. Це тому, що CLR керує кодом, який виконується всередині нього. Наприклад, у програмі написаній на VB, розробник відповідає за перерозподіл пам’яті, яка споживається об’єктом. Якщо програміст забуде звільнити пам’ять, він може отримати винятки нестачі пам’яті. З іншого боку, програмісту .NET не потрібно турбуватися про перерозподіл пам’яті, яку споживає об’єкт. CLR забезпечує автоматичне керування пам’яттю, також відоме як збір сміття. Оскільки CLR керує та виконує проміжну мову, її (IL) також називають керованим кодом.

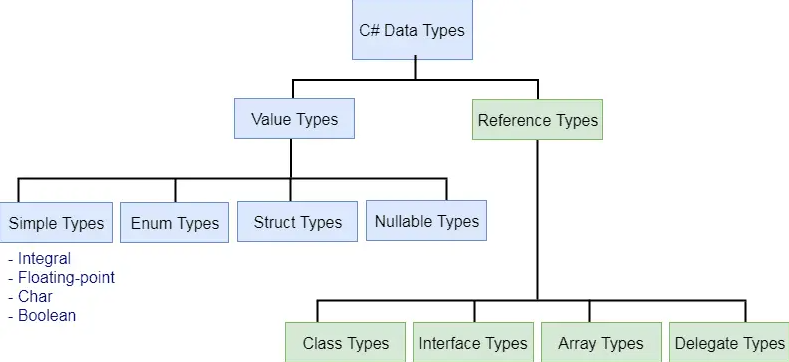
.NET підтримує різні мови програмування, такі як C#, VB, J# і C++. C#, VB і J# можуть генерувати лише керований код (IL), тоді як C++ може генерувати як керований код (IL), так і некерований код (власний код).

Власний код ніде не зберігається постійно після того, як ми закриваємо програму, рідний код викидається. Коли ми знову виконуємо програму, рідний код знову генерується.

**2. Що таке типи даних? Які типи даних є в мові С#? Як вони сполучаються з варіантами збереження інформації в оперативному запам’ятовуючому пристрої? Розкрити поняття «перетворення типів». Яким воно може бути? Які типи даних не можуть бути перетворені неявно ні в які інші типи?**

Тип даних — це класифікація даних, яка повідомляє компілятору чи інтерпретатору, як програміст збирається використовувати дані. Тип даних — це атрибут, пов’язаний із фрагментом даних, який повідомляє комп’ютерній системі, як інтерпретувати його значення. Розуміння типів даних гарантує, що дані збираються в бажаному форматі, а значення кожної властивості відповідає очікуванням.

C# в основному класифікує типи даних на два типи: типи значень і типи посилань. До типів значень належать прості типи (такі як int, float, bool і char), типи enum, struct типи та типи значень Nullable. ці типи даних зберігаються в стеку. До посилальних типів належать типи класів, типи інтерфейсів, типи делегатів і типи масивів. ці типи зберігаються в купі.



Варіанти зберігання інформації в ОЗП:

* 1 байт = 8 біт byte sbyte bool
* 2байти = 16 біт (машинне слово) short ushort char
* 4 байти = 32 біти (подвійне машинне слово) int uint float
* 8 байт = 64 біти (збільшене учетверо машинне слово) long ulong double

**Приведення (перетворення) типів** — в програмуванні це зміна типу сутності одного типу даних в інший, що може відбуватися різними способами, явно чи неявно.

Це використовується з ціллю скористатися деякими особливостями ієрархій типів або подання типу. Одним із прикладів такої ситуації є використання невеликих цілих чисел, які зберігаються в більш компактному форматі змінної і приводяться в тип з більшим діапазоном значень для проведення арифметичних операцій. В об'єктно-орієнтованому програмуванні, дозволяє програмам приводити об'єкти різного типу до їх спільного батьківського типу, для спрощення операцій з ними.

**Неявне перетворення**. Застосовується, коли перетворення з типу А в тип В можливо за будь-якими обставинами, і правила для виконання перетворення є досить простими для того, щоб його можна було довірити компілятору.

**Явне перетворення або приведення типів**. Застосовується тоді, коли перетворення з типу А в тип В можливо тільки за певних обставин або коли правила для перетворення є досить складними для того, щоб було вигідним використання додаткової обробки.

Загальне правило: неявно можна виконувати всі перетворення, які не призведуть до втрати інформації. Тому дані тинів bool, double, decimal **не можуть бути неявно перетворені ні в які типи даних**. Float може бути перетворений в double; int може бути претворений в long, float, double, decimal; long може бути перетворений в float, double, decimal.

**3.Які умовні конструкції є в мові С#?**

if з неповним розгалуженням.

if (умовний\_вираз)

інструкція;

if з повним розгалуженням

if (умовний\_вираз)

{

інструкція;

}

else

{

Інструкція;

}

Тернарний умовний оператор ?

int n = a > b ? a : b;

//аналогічний запис

if(a > b)

{

n = a;

}

else

{

n = b;

}

Умовний оператор switch

switch (r)

{

case 1:

case 2:

v = "Незадовільно";

break;

case 3:

v = "Задовільно";

break;

case 4:

v = "Добре";

break;

case 5:

v = "Відмінно";

break;

default:

v = "Помилка";

break;

}

**4. Які циклічні конструкції є в мові С#?**

У C# існують 4 типи циклів: for , while , do-while , foreach

1.for(ініціалізація; умова; інструкції)

{

// інструкції;

}

2.while (умова) {

//інструкції

}

3.do

{

//інструкції

} while (умова);

4.foreach (int number in numbers)

Оператор foreach виконує оператор або блок операторів для кожного елемента в екземплярі типу, який реалізує інтерфейс System.Collections.IEnumerable або System.Collections.Generic.IEnumerable<T>

**5. Що таке метод? В чому полягає різниця між функціями та процедурами? Що таке сигнатура та семантика метода?**

Методи є підпрограми, які маніпулюють даними, визначеними в класі, а в багатьох випадках вони надають доступ до цих даних та можуть викликатися стільки раз, скільки необхідно. Як правило, інші частини програми взаємодіють з класом за допомогою його методів. Загальна форма визначення методу :

доступ тип,що повертається імʼя (список параметрів)

{

// тіло методу

}

**різниця між функціями та процедурами** - функції повертають значення, процедури не повертають значення

**Сигнатура** методу це ім'я методу плюс параметри (причому порядок параметрів має значення). У сигнатуру методу не входить значення, що повертається, а також викиди, що кидаються ним.

**Семантика** методу – це опис того, що даний метод робить. Семантика методу включає опис того, що є результатом обчислення методу, як і від чого цей результат залежить. Зазвичай, результат виконання залежить від значень аргументів методу. До поняття семантики можна віднести як тіло методу і значення, що повертається, так і out аргументи.

**6. В чому різниця між передачею параметра за значенням і за посиланням? В чому різниця між ref та out параметрами?**

Метод виклику за значенням передає лише значення змінної до коду функції, і якщо є будь-яка зміна у значенні змінної всередині цієї функції, це не впливає на початкове значення цієї змінної. У методі call by reference ми передаємо цю змінну в аргумент, і зміна значення змінної також впливає на початкове значення цієї змінної. Основна відмінність між двома методами полягає в тому, що метод call by value передає значення змінної, а call by reference проходить адресу цієї змінної.

Модифікатор ref призначений для вказання того, що параметр методу повинен передаватись за посиланням а не за значенням. Іншими словами, якщо в метод потрібно примусово передати аргумент за посиланням, то при оголошенні методу перед відповідним формальним параметром потрібно вказати модифікатор ref.

Модифікатор параметру ref використовується в описі формального параметру методу. Модифікатор параметру ref вказується перед типом формального параметру, як показано нижче:

Модифікатор out використовується, якщо потрібно виконання двох умов:

методу не потрібно передавати значення;

метод обов’язково повинен повертати значення з допомогою параметру.

Між модифікаторами ref та out є три взаємопов’язані відмінності:

1. Параметр з модифікатором out використовується тільки для повернення значення з методу. Параметр з модифікатором ref може використовуватись і для повернення і для задавання значення в методі іншим змінним. Тому, перед викликом методу, немає потреби присвоювати якесь значення змінній, що використовується з модифікатором out.

2. В методі, змінна оголошена з параметром out вважається неініціалізованою. Змінна, оголошена з ref вважається ініціалізованою. Тому, не можна використовувати out-змінну у правій частині оператора присвоювання. А ref-змінну можна.

3. Якщо параметр оголошено з модифікатором out, то в тілі методу цьому параметру обов’язково має бути присвоєне якесь значення. Інакше буде помилка компіляції. Якщо параметр оголошено з модифікатором ref, то цьому параметру присвоювати значення в тілі методу не обов’язково.

**7. Як працює рекурсія?**

Метод викликає сам себе безпосередньо (проста рекурсія)

або через інший метод (складна рекурсія).

Якщо метод викликає сам себе, то у пам'яті відбуваються такі процеси: у системному стеку виділяється пам'ять для нових локальних змінних та параметрів; програмний код методу виконується спочатку з новими локальними змінними та параметрами. Ці локальні змінні та параметри набувають нових початкових значень. Сам програмний код методу не копіюється; при поверненні з рекурсивного методу (оператор return) відбувається відновлення старих локальних змінних та параметрів, а також їх значень у точці виклику цього методу.

**8. Які різновиди масивів реалізуються в мові С#?**

С# підтримує наступні типи масивів:

* Одновимірні – це набір елементів фіксованої довжини та наперед заданого типу, доступ до яких здійснюється з використанням одного індексу.
* Ступінчасті - масив масивів, різної розмірності
* Багатовимірні

***ЗАВДАННЯ***

***Загальні завдання***

Покликання на гітхаб з кодом - https://github.com/doniev-danylo/kneu\_cs.git

***Завдання 1***

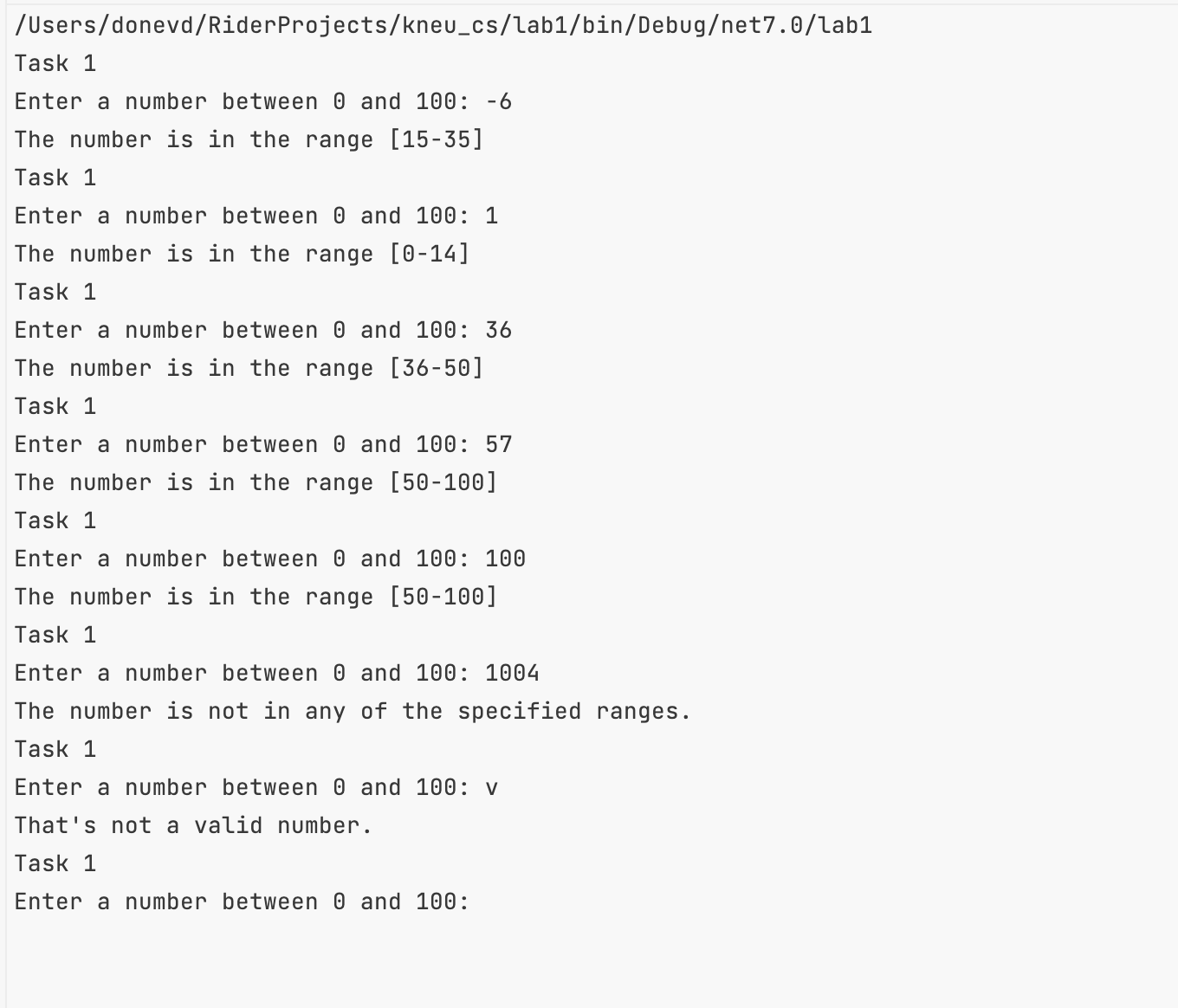
Використовуючи Visual Studio, створіть проект за шаблоном ConsoleApplication. Напишіть програму визначення щодо попадання вказаного користувачем числа від 0 до 100 у числовий діапазон [0-14] [15-35] [36-50] [50-100]. Якщо так, то вкажіть, в який саме. Якщо користувач вказує число, яке не входить в жоден з наявних числових діапазонів, то виводиться відповідне повідомлення.ʼ

КОД ПРОГРАМИ

public class Program  
{  
 private static void Task1()  
 {  
 Console.WriteLine("Task 1");  
 Console.Write("Enter a number between 0 and 100: ");  
 var input = Console.ReadLine();  
  
 if (int.TryParse(input, out var number))  
 {  
 switch (number)  
 {  
 case >= 0 and <= 14:  
 Console.WriteLine("The number is in the range [0-14]");  
 break;  
 case <= 35:  
 Console.WriteLine("The number is in the range [15-35]");  
 break;  
 case <= 50:  
 Console.WriteLine("The number is in the range [36-50]");  
 break;  
 case <= 100:  
 Console.WriteLine("The number is in the range [50-100]");  
 break;  
 default:  
 Console.WriteLine("The number is not in any of the specified ranges.");  
 break;  
 }  
 }  
 else  
 {  
 Console.WriteLine("That's not a valid number.");  
 }  
 }

public static void Main()  
 {  
 while (true)  
 {  
 Task1();} }  
}

РЕЗУЛЬТАТИ



***Завдання 2***

Є N клієнтів, яким компанія виробник повинна поставити товар. Скільки існує можливих маршрутів постачання товару, з врахуванням того, що товар буде постачати одна машина?

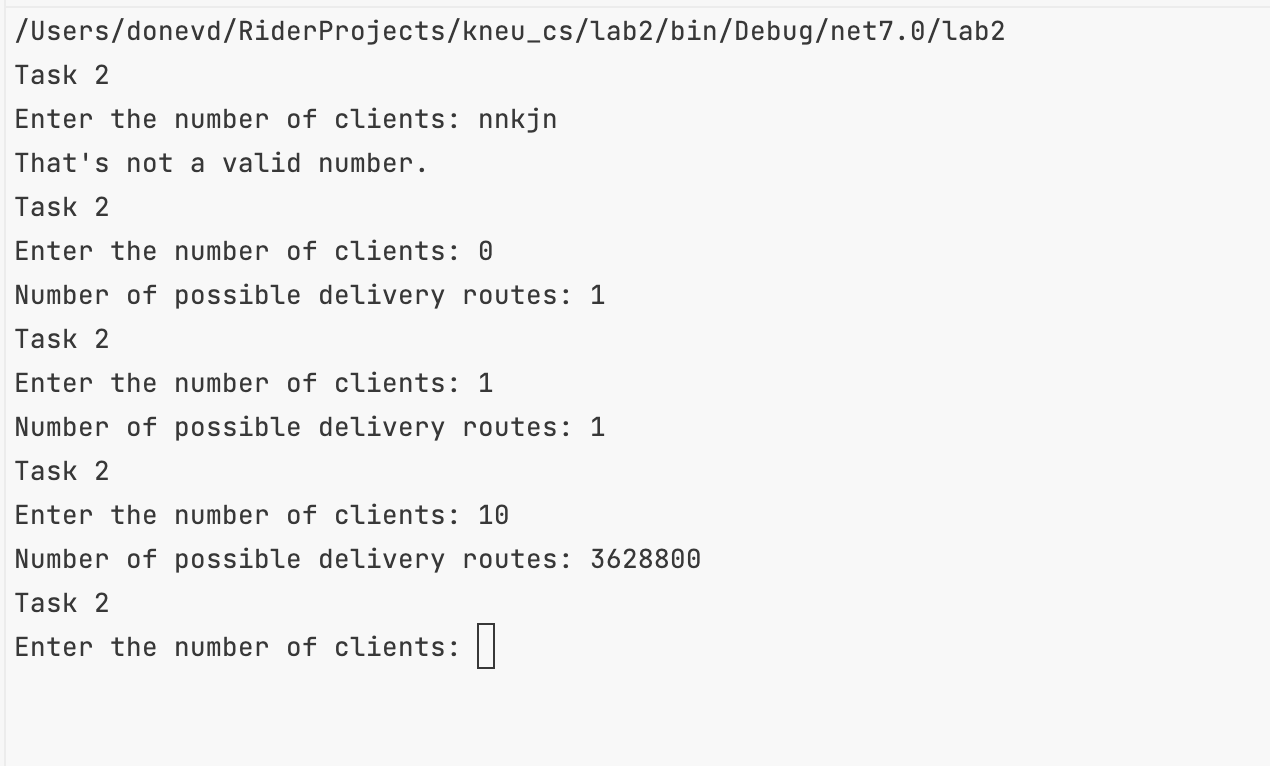
Використовуючи VisualStudio, створіть проект за шаблоном ConsoleApplication.

Напишіть програму, яка буде розраховувати та виводити на екран кількість можливих варіантів доставки товару. Для розв’язку задачі, використайте факторіал N!, який розраховується за допомогою цикла do-while.

КОД ПРОГРАМИ

public class Program  
{  
 private static void Task2()  
 {  
 Console.WriteLine("Task 2");  
 Console.Write("Enter the number of clients: ");  
 var input = Console.ReadLine();  
  
 if (int.TryParse(input, out var n))  
 {  
 long fact = 1;  
 var count = 1;  
 do  
 {  
 fact \*= count;  
 count++;  
 } while (count <= n);  
  
 Console.WriteLine($"Number of possible delivery routes: {fact}");  
 }  
 else  
 {  
 Console.WriteLine("That's not a valid number.");  
 }  
 }  
  
  
 public static void Main()  
 {  
 while (true)  
 {  
 Task2();  
 }  
 }  
}

РЕЗУЛЬТАТИ



***Індивідуальні завдання***

**Варіант 5**

Використовуючи VisualStudio, створити проєкт за шаблоном ConsoleApplication. Написати метод, який приймає два рядкові аргументи та повертає їх конкатенацію. Результат вивести на екран.

КОД ПРОГРАМИ

public class Program  
{  
 private static void ConcatenateStrings()  
 {  
 Console.WriteLine("Concatenate Strings | variant 5");  
 Console.Write("Enter the first string: ");  
 var str1 = Console.ReadLine();  
  
 Console.Write("Enter the second string: ");  
 var str2 = Console.ReadLine();  
  
 Console.WriteLine($"Result: {str1 + str2}");  
 }  
  
  
 public static void Main()  
 {  
 while (true)  
 {  
 ConcatenateStrings();  
 }  
 }  
}

РЕЗУЛЬТАТИ

