МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА
Факультет прикладної математики та інформатики

Кафедра програмування

Лабораторна робота № 7

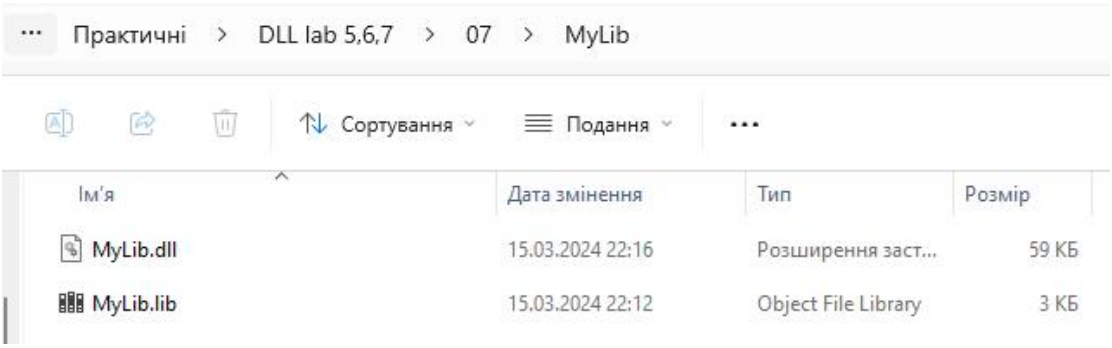
Використання DLL мовами програмування

Виконала
студентка групи ПМО-41
Кравець Ольга

2024

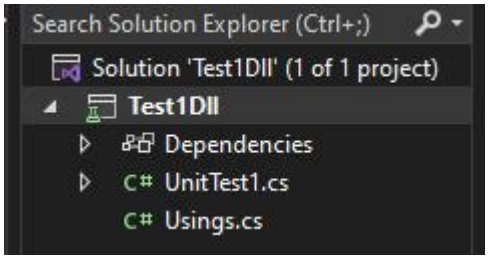
Хід роботи

Взяла за основу бібліотечні файли *.dll і *.lib, створені в лабораторній роботі 5. Ці файли скопіювала в свій окремий каталог, до якого буде доступ з різних проєктів.

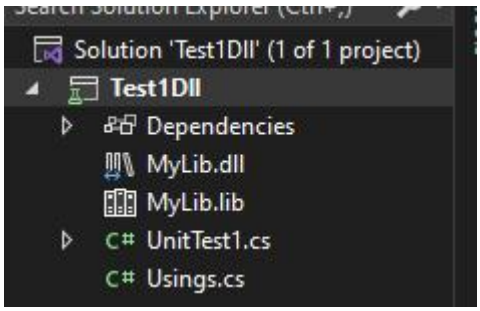


Функції DLL в програмах C# (неявне зв’язування з бібліотеками DLL).

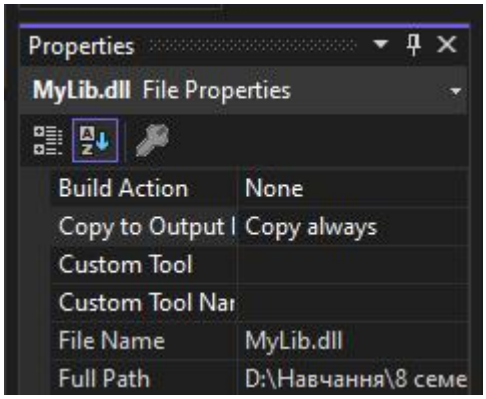
Створила новий NUnit Test Project в Visual Studio для тестування методів з dll файлу.



Для того, щоб dll була знайдена я просто додада цей файл до проєкту в Visual Studio.



У властивостях файла вказала копіювати його в папку з результатом білду програми.



Оголосила кожну функцію окремо для неявного зв’язування.

```
[DllImport("MyLib.dll")]
0 references
private static extern int FindMinimum(int[] s, int arrSize);

[DllImport("MyLib.dll")]
0 references
private static extern int CalculateFactorial(int n);

[DllImport("MyLib.dll")]
0 references
private static extern bool ContainsSub(string toCheck, int checkedSize, string toFind, int foundSize, bool caseSensitiv

[DllImport("MyLib.dll")]
0 references
private static extern double AverageValue(int[] numArr, int arrSize);
```

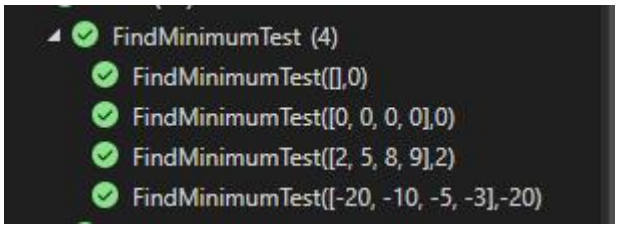
Написала юніт тести до кожного з методів і виконала тестування на різних даних.

Тестування функції FindMinimum.

```
//Тестування функції FindMinimum
[TestCase(new[] { 2, 5, 8, 9 }, 2)]
[TestCase(new[] { -20, -10, -5, -3 }, -20)]
[TestCase(new[] { 0, 0, 0, 0 }, 0)]
[TestCase(new int[] { }, 0)]
0 references
public void FindMinimumTest(int[] inputArray, int expectedMinimum)
{
    int min = FindMinimum(inputArray, inputArray.Length);

    Assert.That(min, Is.EqualTo(expectedMinimum));
}
```

Усі тести пройшли успішно.

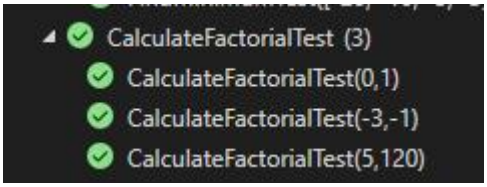


Тестування функції CalculateFactorial.

```
// Тестування функції CalculateFactorial
[TestCase(5, 120)]
[TestCase(0, 1)]
[TestCase(-3, -1)]
0 references
public void CalculateFactorialTest(int n, int shouldBeResult)
{
    int factorialResult = CalculateFactorial(n);

    Assert.That(factorialResult, Is.EqualTo(shouldBeResult));
}
```

Усі тести пройшли успішно.

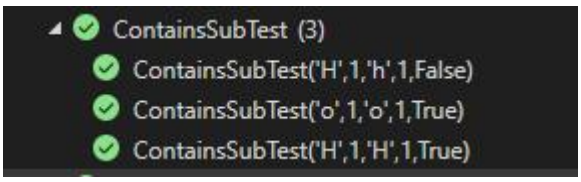


Тестування функції ContainsSub.

```
// Тестування функції ContainsSub
[TestCase('H', 1, 'H', 1, true, ExpectedResult = true)]
[TestCase('o', 1, 'o', 1, true, ExpectedResult = true)]
[TestCase('H', 1, 'h', 1, false, ExpectedResult = true)]
0 references
public bool ContainsSubTest(char toCheck, int checkedSize, char toFind, int foundSize, bool caseSensitive)
{
    bool containsResult = ContainsSub(toCheck, checkedSize, toFind, foundSize, caseSensitive);

    return containsResult;
}
```

Усі тести пройшли успішно.

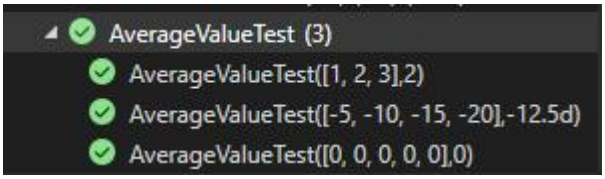


Тестування функції AverageValue.

```
// Тестування функції AverageValue
[TestCase(new int[] { 0, 0, 0, 0, 0 }, 0)]
[TestCase(new int[] { 1, 2, 3 }, 2)]
[TestCase(new int[] { -5, -10, -15, -20 }, -12.5)]
0 references
public void AverageValueTest(int[] numArr, double expectedAverage)
{
    double averageResult = AverageValue(numArr, numArr.Length * sizeof(int));

    Assert.That(averageResult, Is.EqualTo(expectedAverage).Within(0.001));
}
```

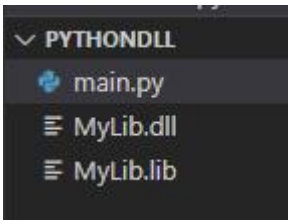
Усі тести пройшли успішно.



Функції DLL в програмах інших алгоритмічних мов.

Створила новий Python проєкт.

Помістила dll в одну папку з .py файлом та задала шлях відносно нього. Після чого завантажила бібліотеку.




```python
import os
import ctypes

dirName = os.path.dirname(__file__)
dllPath = os.path.join(dirName, 'MyLib.dll')

hDLL = ctypes.WinDLL(dllPath)
```

Створила вказівники на функції, задала їм типи повернення та для двох з них задала типи вхідних параметрів.

```python
# Оголошення функцій з бібліотеки DLL
FindMinimum = hDLL.FindMinimum
FindMinimum.restype = ctypes.c_int
FindMinimum.argtypes = [ctypes.POINTER(ctypes.c_int), ctypes.c_int]

CalculateFactorial = hDLL.CalculateFactorial
CalculateFactorial.restype = ctypes.c_int
CalculateFactorial.argtypes = [ctypes.c_int]

ContainsSub = hDLL.ContainsSub
ContainsSub.restype = ctypes.c_bool
ContainsSub.argtypes = [ctypes.c_char_p, ctypes.c_int, ctypes.c_char, ctypes.c_int, ctypes.c_bool]

AverageValue = hDLL.AverageValue
AverageValue.restype = ctypes.c_double
AverageValue.argtypes = [ctypes.POINTER(ctypes.c_int), ctypes.c_int]
```

Прописала обгортки функцій, які забезпечують зручний і безпечний спосіб використання функцій з бібліотеки DLL у середовищі Python. Вони здійснюють необхідні конвертації типів даних та обробку параметрів перед викликом функцій з DLL.

Функція find_minimum(values) приймає список цілих чисел і викликає функцію FindMinimum з бібліотеки DLL. Спочатку створюється масив типу ctypes.c_int, який містить передані значення, а потім цей масив разом з його довжиною передається у функцію FindMinimum. Результат, повернутий з DLL, є найменшим значенням у списку.

```
27  def find_minimum(values):
28      length = len(values)
29      arr = (ctypes.c_int * length)(*values)
30      return FindMinimum(arr, length)
```

Функція calculate_factorial(n) приймає одне ціле число n і викликає функцію CalculateFactorial з бібліотеки DLL. Параметр n передається безпосередньо у функцію CalculateFactorial, а результат, повернутий з DLL, є факторіалом числа n.

```
31  def calculate_factorial(n):
32      return CalculateFactorial(n)
```

Функція contains_sub(to_check, to_find, case_sensitive=True) приймає два рядки to_check та to_find і параметр case_sensitive, який за замовчуванням встановлено як True. Вона викликає функцію ContainsSub з бібліотеки DLL. Рядки перетворюються у байтові об'єкти, якщо case_sensitive встановлено як False і передаються у функцію ContainsSub, а результат, отриманий з DLL, показує, чи містить рядок to_check підрядок to_find.

```
35  def contains_sub(to_check, to_find, case_sensitive=True):
36      to_check_bytes = bytes(to_check, 'utf-8')
37      to_find_bytes = bytes(to_find, 'utf-8')
38      result = ContainsSub(to_check_bytes, len(to_check_bytes), to_find_bytes, len(to_find_bytes), case_sensitive)
39      return result
```

Функція average_value(num_arr) приймає список цілих чисел num_arr і викликає функцію AverageValue з бібліотеки DLL. Спочатку створюється масив типу ctypes.c_int, який містить передані значення з num_arr, а потім цей масив разом з його довжиною передається у функцію AverageValue. Результат, отриманий з DLL, є середнім значенням у списку цілих чисел.

```
41    def average_value(num_arr):
42        arr = (ctypes.c_int * len(num_arr))(*num_arr)
43        return AverageValue(arr, len(num_arr))
44
```

Написала тести до усіх функцій.

Тестування FindMinimum (викликаю через обгортку).

```
minValue = find_minimum([2, 5, 8, 9])
if minValue != 2:
    print(f'Error! Test FindMinimum function failed!')
else:
    print('Test FindMinimum function passed successfully!')
```

Тестування CalculateFactorial (викликаю через обгортку).

```
number = 5
factorial = calculate_factorial(number)
if factorial != 120:
    print(f'Error! Test CalculateFactorial function failed!')
else:
    print('Test CalculateFactorial function passed successfully!')
```

Тестування ContainsSub (викликаю через обгортку).

```
to_check = "Hello, World!"
to_find = "world"
case_sensitive = False
result = contains_sub(to_check, to_find, case_sensitive)
if result:
    print('Test ContainsSub function passed successfully!')
else:
    print(f'Error! Test ContainsSub function failed!')
```

Тестування AverageValue (викликаю через обгортку).

```
num_arr = [1, 2, 3, 4, 5]
expected_average = 1
result = average_value(num_arr)
if result == expected_average:
    print('Test AverageValue function passed successfully!')
else:
    print(f'Error! Test AverageValue function failed!')
```

Усі тести пройшли успішно.

```
PS D:\Навчання\8 семестр\Операційні системи та системне
thon312/python.exe "d:/Навчання/8 семестр/Операційні сис
Test FindMinimum function passed successfully!
Test CalculateFactorial function passed successfully!
Test ContainsSub function passed successfully!
Test AverageValue function passed successfully!
```

Оцінка ефективного використання DLL мовами інтерпретованого типу, такими як Python, Visual Basic та Java, може бути корисною для забезпечення оптимальної швидкодії та інтеграції з існуючими розробками.

Нижче навела кілька рекомендацій щодо ефективного використання DLL в цих мовах:

1. Використання бібліотеки ctypes у Python:
   - ✓ Коли використовується Python, модуль ctypes може бути використаний для виклику функцій з DLL. Це робиться через використання функції ctypes.cdll.LoadLibrary().
   - ✓ Для підвищення продуктивності можна використовувати векторизацію чи асинхронний код, де це можливо, для оптимізації роботи з великими обсягами даних.
2. Використання COM (Component Object Model) у Visual Basic:
   - ✓ У Visual Basic можна використовувати технологію COM для роботи з бібліотеками DLL.

- ✓ Використання декларації функцій і змінних зі специфікатором Declare дозволяє Visual Basic взаємодіяти з функціями DLL.

3. Використання Java Native Interface (JNI) у Java:
   - ✓ У Java можна використовувати JNI для зв'язування з бібліотеками DLL. Це зазвичай використовується для забезпечення інтеграції з низькорівневими операціями або функціями, що написані на мові C/C++.
   - ✓ Важливо враховувати правильне управління пам'яттю при використанні JNI, оскільки це може призвести до витоку пам'яті або інших проблем з продуктивністю.
4. Оптимізація взаємодії з DLL:
   - ✓ Уникайте зайвої взаємодії з DLL, зокрема зайвих викликів функцій, які можуть призвести до зниження продуктивності.
   - ✓ Враховуйте можливості кешування результатів функцій DLL, щоб уникнути зайвих обчислень.
5. Перевірка на витоки пам'яті та інші проблеми:
   - ✓ При використанні великих обсягів даних або багатократних викликах функцій з DLL важливо перевіряти програму на витоки пам'яті та інші проблеми з продуктивністю.

Врахування цих рекомендацій допоможе забезпечити ефективне використання функцій з DLL у мовах програмування інтерпретованого типу.