ОС і СП
Лабораторна робота 11: інтерпретація без рангів

Вивчити модель інтерпретатора формул за операціями без рангів. Доповнити інтерпретатор іншими бінарними і унарними операціями. Реалізувати можливість вибору системи числення.

Матеріали до завдання

Теоретичний матеріал початкового рівня для цієї роботи викладений в лекції "L11 Обчислення формул інтерпретацією.pdf" в розділах "Правила будови синтаксичних визначень" і "Найпростіший приклад інтерпретації".

Для глибшого вивчення питань цього розділу курсу і для виконання лабораторних робіт 11 і 12 треба звертатись за посиланнями:

Вікіпедія. Інтерпретатор мови програмування [Електронний ресурс]
https://uk.wikipedia.org/wiki/Інтерпретатор

Інтерпретатор [Електронний ресурс]
http://kytok.org.ua/?p=732

Вікіпедія. Компілятор [Електронний ресурс]
https://uk.wikipedia.org/wiki/Компілятор

Завдання частина 1

1. На початку виконати різні обчислювальні експерименти з доданим файлом `simple.py`. Зокрема, в останньому розділі файла для виконання

```
"if __name__ == "__main__" :"
```

пробувати виконати різні підстановки в оператор "`formula = ...`", щоб мати достатнє уявлення про методи розбору виразів та їх інтерпретацію.

Пояснення до п.1.

1.1. В одному рядку текстового файлу чи в деякому іншому текстовому рядку записана послідовність десяткових цілих чисел і знаків арифметичних операцій (формула). Наприклад:

65 + 122 – 99 / 6 – 12 * 2 + 1

Необхідно виконати обчислення формули за такими *правилами і обмеженнями*: числа десяткові цілі; допустимі операції є +, -, *, /; операції

рангу не мають і їх виконують зліва направо в порядку запису; результатом кожної операції є ціле або дійсне число; формула записана коректно, зокрема, для запису чисел використані лише десяткові цифри і немає операції ділення на нуль. Така формула дуже нагадує просту послідовність операцій на звичайному калькуляторі.

1.2. Програмний код мовою Python інтерпретатора таких формул є в доданому файлі `simple.py`. Вивчити програмний код. Можна прийняти його за основу або реалізувати іншою алгоритмічною мовою.

Інтерпретатор працює за два перегляди тексту формули. До початку першого перегляду метод `delblank()` викреслює з тексту всі незначущі пропуски. За першим переглядом будують список лексем цілої формули. За другим переглядом обчислюють формулу, якщо не було помилок за першим переглядом. У випадку помилок друкують найпершу помилкову знайдену літеру.

Оператори `print()` метода `calc()` можна розкоментувати, щоб побачити проміжні результати процедури обчислення. Можна також додати для експериментів інші оператори `print()` до тексту методів, і отримати спостереження за ходом виконання інтерпретатора.

1.3. Результати кроків виконання інтерпретації є такі:

задано	`"65 + 122 - 99 / 6 - 12 * 2 + 1"`
викреслити пропуски	`"65+122-99/6-12*2+1"`
поділити на лексеми	`['65', '+', '122', '-', '99', '/', '6', '-', '12', '*', '2', '+', '1']`
додати знак "+" на початок	`['+', '65', '+', '122', '-', '99', '/', '6', '-', '12', '*', '2', '+', '1']`
цикл обчислення пар [знак,операнд]	`['+', '65'], ['+', '122'], ['-', '99'], ['/', '6'], ['-', '12'], ['*', '2'], ['+', '1']`

2. Виконайте декілька експериментів обчислення різних формул, в тому числі формул з помилками.

Завдання частина 2

Основна частина завдання викладена в пп.3,4,5. Щоб виконати ці пункти, треба на початку побудувати розширене синтаксичне означення правил запису формули відповідно до кожного пункта, після цього виконати редагування програмного коду інтерпретатора.

УВАГА! Рекомендуємо спочатку повністю реалізувати п.3 завдання, виконати повне тестування, отримати правильні результати, тоді перейти до п.4 завдання. Так само далі - до п.5.

3. Додайте до інтерпретатора `simple.py` інші бінарні операції, наприклад, "*остача від ділення*", "*менше з останнього обчисленого і наступного*". Бажано визначити чотири інші бінарні операції, ніж реалізовані +, -, *, /.

4. Додайте до інтерпретатора 2-3 унарні операції, наприклад, *корінь квадратний* чи *sin*. Унарну операцію в нашому випадку застосовують до останнього обчисленого результату (постфіксна форма). Наприклад, запис

```
25 + 52 sqrt – 120 sin * 2
```

означає

```
( sin ( sqrt( 25 + 52 ) – 120 ) ) * 2
```

5. Реалізуйте можливість виконання обчислень в інших системах числення: *двійкова*; *шістнадцяткова*. Для цього треба зафіксувати допустимі літери для запису цифр, виконувати переведення з іншої системи числення в десяткову кожного числа, самі обчислення можна виконати, як раніше, в десятковій системі числення, в кінці показати результат у вхідній системі числення.

Подумати, як позначити в самій формулі систему числення. Повний запис має укладатись в один рядок.

Звіт за роботу

В результаті виконання завдання надіслати:

1) побудовані власні розширені синтаксичні означення формул для кожного пп.3,4,5 завдання окремо;

2-3) програмний код файлу python доповненого інтерпретатора `simple.py`, спільний для всіх пп.3,4,5 завдання;

2-3) якщо реалізація іншою мовою, ніж python, тоді надіслати архів проєкту;

4) в будь-якому варіанті реалізації програмний код має бути коментований в тих позиціях, де були внесені зміни, щоб бачити різницю від базового варіанту `simple.py`;

5) тестові приклади формул, на яких виконали перевірку інтерпретатора, разом з отриманими результатами; звертаємо увагу, що перелік тестів має бути достатнім для повної демонстрації реалізації всіх елементів завдання;

6) відповіді до 1 і 5 оформити коротким звітом - ОКРЕМИМ ФАЙЛОМ, не архівованим.