УДК 004.021:51

АВТОМАТИЗАЦІЯ ПЕРЕВЕДЕННЯ ЧИСЕЛ З ДОВІЛЬНОЇ ПОЗИЦІЙНОЇ СИСТЕМИ ЧИСЛЕННЯ В ДЕСЯТКОВУ ЗА ДОПОМОГОЮ СХЕМИ ГОРНЕРА ТА РОЗШИРЕНОГО ЗАПИСУ ЧИСЛА

Оніщенко Т.В.[1]; Буц О.В.[2]
[1,2]Національний університет «Одеська політехніка»
[1,2]Україна, Одеса
[1]tonishenko@gmail.com; [2]8404423@stud.op.edu.ua

Анотація. *В роботі на основі схеми Горнера та розширеного запису числа розроблено алгоритм на мові Python, який демонструє покрокову реалізацію цих методів, забезпечуючи візуалізацію обчислень для освітніх цілей. Досліджено теоретичні основи методів, включаючи необхідний математичний апарат, аналіз складності та обробку дробових частин.*
Ключові слова. *системи числення; схема Горнера; розширений запис числа; Python; автоматизація обчислень.*

Вступна частина. Системи числення є фундаментальною частиною програмної інженерії, оскільки визначають способи представлення та обробки даних у цифрових системах [1]. Розуміння принципів переведення між системами числення дозволяє ефективно вирішувати задачі кодування, криптографії, обробки сигналів та оптимізації алгоритмів. Недостатнє розуміння цих принципів може призводити до помилок у програмуванні, особливо в низькорівневих системах.

Мета дослідження полягає у реалізації алгоритму на мові програмування Python, який демонструє покрокову реалізацію цих методів, забезпечуючи візуалізацію обчислень для вузькоспеціалізованих освітніх цілей.

Основна частина. Розширений запис числа — це запис, який являє собою суму творів цифр числа на підставі системи числення, яка зведена в ступінь відповідного розряду [2]. Розширений запис числа може бути представлений в наступному вигляді:

$$A_N=+(a_{n-1}N^{n-1}+a_{n-2}N^{n-2}+\ldots+a_0+a_{-1}N^{-1}+\ldots a_{-s}N^{-s}),$$

де A — число, N — основа системи числення, a — цифри, що належать алфавіту даної системи числення, n — число цілих розрядів числа, s — число дрібних розрядів числа.

Підстава N вказує, у скільки разів одиниця n + 1-го розряду більше одиниці n-го розряду, а цифра a_n відповідає кількості одиниць n-го розряду, що містяться в числі.

Кількісний еквівалент цифри в числі залежить від займаної нею позиції, і чисельно дорівнює добутку кількісного еквівалента власне цифри на вагу позиції, у якій вона розміщена.

Число, так само як і цифра, є кодом деякої кількості та, отже, має кількісний еквівалент. Кількісний еквівалент числа дорівнює сумі кількісних еквівалентів усіх його цифр з урахуванням ваги їх позицій:

$$A = \sum_{i=s}^{n-1} a_i N^i$$

Схема Горнера — алгоритм обчислення значення многочлена, записаного у вигляді суми одночленів $A_N=+(a_{n-1}N^{n-1}+a_{n-2}N^{n-2}+\ldots+a_0+a_{-1}N^{-1}+\ldots a_{-s}N^{-s})$, при заданому значенні змінної обчислюється за формулами [3]:

Ціла частина: $A_N = (((a_{p-1}N + a_{p-2})N\ldots + a_2)N + a_1)N + a_0$

Дробова частина: $A_N = ((((a_{-1}N + a_{-2})N \ldots + a_{n-1})N + a_s)N^{-s}$,

де N — основа системи числення, з якої проводиться переведення.

Розроблений алгоритм на Python передбачає введення користувачем числа як рядка (для підтримки літер) та основи (від 2 до 36). Використовується словник для мапінгу літер на числові значення (наприклад, 'A' → 10). Функція нормалізації перевіряє валідність символів та перетворює рядок у список цілих чисел.

```python
digits = {ch: str(i) for i, ch in enumerate('0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ')}

def normalize(number: str, base: int):  2 usages
    result = []
    for ch in number:
        if ch == '.':
            result.append('.')
        elif ch in digits:
            val = int(digits[ch])
            if val >= base:
                raise ValueError(f"Символ '{ch}' не допустим в системі з основою {base}")
            result.append(val)
        else:
            raise ValueError(f"Недопустимий символ: {ch}")
    return result
```


```python
digits = {ch: str(i) for i, ch in enumerate('0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ')}

def normalize(number: str, base: int):  2 usages
    result = []
    for ch in number:
        if ch == '.':
            result.append('.')
        elif ch in digits:
            val = int(digits[ch])
            if val >= base:
                raise ValueError(f"Символ '{ch}' не допустим в системі з основою {base}")
            result.append(val)
        else:
            raise ValueError(f"Недопустимий символ: {ch}")
    return result
```


Рисунок 1. Нормалізація числа для розрахунків

Для розгорнутого запису обчислюється позиція крапки (для дробових чисел), степені основи та сума добутків з покроковим виводом.

```python
def expansion_method(nums: list, base: int):  1 usage
    steps = []
    total = 0

    if '.' in nums:
        dot_index = nums.index('.')
    else:
        dot_index = len(nums)

    for i, digit in enumerate(nums):
        if digit == '.':
            continue

        if i < dot_index:
            power = dot_index - i - 1

        else:
            power = -(i - dot_index)

        term = digit * (base ** power)
        total += term
        steps.append(f"{digit}*{base}^{power}={term}")

    print("Формула:", " + ".join(steps))
    return total
```

Рисунок 2. Нормалізація числа для розрахунків

Схема Горнера реалізується через акумулятор: для цілої частини — множення на основу та додавання; для дробової — додавання з діленням на основу, з формуванням формули для візуалізації.

```python
def horner_part(nums: list, base: int, fractional=False):  3 usages
    result = 0
    steps = []

    iterable = reversed(nums) if fractional else nums
    for n in iterable:
        if fractional:
            steps.append(f"({result} + {n} / {base})")
            result = (result + n) / base
        else:
            steps.append(f"({result} * {base} + {n})")
            result = result * base + n

    return result, steps

def horner_full(nums: list, base: int):  1 usage
    if '.' in nums:
        dot_index = nums.index('.')
        whole, frac = nums[:dot_index], nums[dot_index+1:]

        main_result, main_steps = horner_part(whole, base)
        frac_result, frac_steps = horner_part(frac, base, fractional=True)

        print(f"Формула:\n{' -> '.join(main_steps)} - \n{' -> '.join(frac_steps)}")
        return main_result + frac_result
    else:
        main_result, main_steps = horner_part(nums, base)
        print(f"Формула:\n{' -> '.join(main_steps)}")
        return main_result
```

Рисунок 3. Функція для переведення розширеним записом числа

Користувач обирає метод, після чого викликається відповідна функція.

```python
method = int(input("Виберіть метод переводу:\n 1 - схема Горнера\n 2 - розгорнутий запис числа\n"))
if method == 1:
    print(f"Результат: {horner_full(normalize(your_number, base_number), base_number)}")
elif method == 2:
    print(f"Результат: {expansion_method(normalize(your_number, base_number), base_number)}")
```

Рисунок 4. Функція для переведення розширеним записом числа

Тестування на однакових вхідних даних показує ідентичні результати для обох методів, підтверджуючи коректність.

```
Введіть ваше число для переводу: 5C2.52
Введіть СЧ вашого числа: 16
Виберіть метод переводу:
 1 - схема Горнера
 2 - розгорнутий запис числа
1
Формула:
(0 * 16 + 5) -> (5 * 16 + 12) -> (92 *
(0 + 2 / 16) -> (0.125 + 5 / 16)
Результат: 1474.3203125
```

```
Введіть ваше число для переводу: 5C2.52
Введіть СЧ вашого числа: 16
Виберіть метод переводу:
 1 - схема Горнера
 2 - розгорнутий запис числа
2
Формула: 5*16^2=1280 + 12*16^1=192 + 2*16^0=2 + 5*16^-1=0.3125 + 2*
Результат: 1474.3203125
```

Рисунок 5. Функція для переведення розширеним записом числа

Висновки. Розроблений Python-алгоритм автоматизує переведення чисел за допомогою схеми Горнера та розширеного запису числа, а головне демонструє покрокову реалізацію цих методів для забезпечення візуалізації обчислень, поєднуючи теорію та практику. Розроблений інструмент конвертації демонструє практичну реалізацію, дозволяючи користувачам візуалізувати достатньо складні процеси.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] Полін, Є.Л. (2012). *Арифметика ЕОМ частина І. Арифметика в форматах з фіксованою крапкою*. Одеса, Україна: ВМВ.
[2] Фісун, М.Т., Цибенко, Б.О. (2010). *Структури та організація даних в ЕОМ: Навчальний посібник*. Миколаїв, Україна: Вид-во ЧДУ ім. Петра Могили.
[3] Akinwunmi, A.O., Ajayi, O.O., Folorunso, T.A. (2024). Efficiency analysis and optimization techniques for base conversion algorithms in computational systems. *International Journal of Innovative Science and Research Technology, 9*(8), 66–72.

Onishchenko T.V., Buts O.V.

Automation of converting numbers from an arbitrary positional number system to the decimal system using Horner’s method and expanded number representation

Abstract. *The paper presents a Python-based algorithm developed using Horner's method and expanded number representation. The algorithm demonstrates a step-by-step implementation of these methods, providing visualization of calculations for educational purposes. The theoretical foundations of the methods are investigated, including the necessary mathematical apparatus, complexity analysis, and handling of fractional parts.*

Keywords: *number systems; Horner's method; expanded number representation; Python; automation of calculations.*