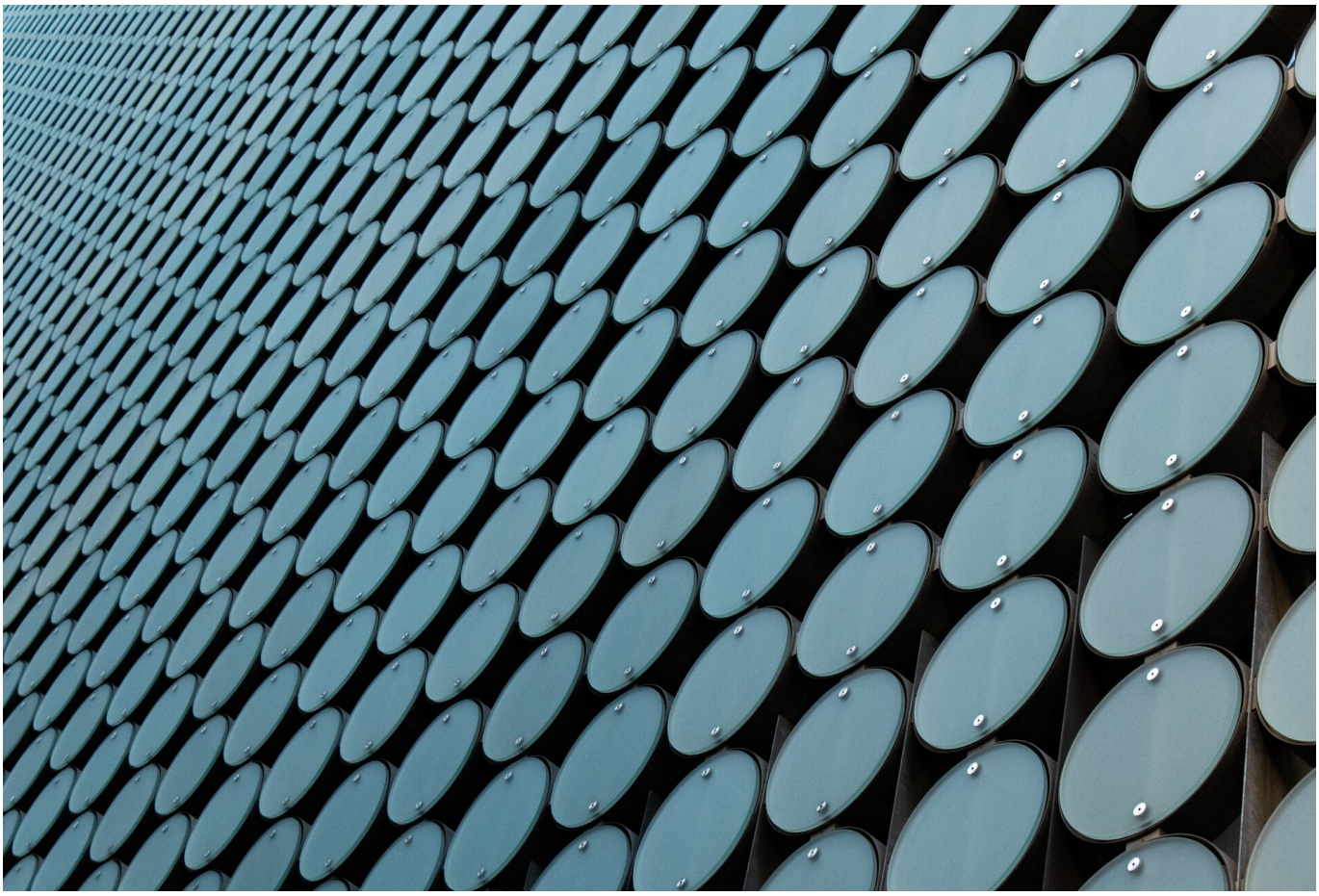


Analiza podejścia rekurencyjnego i iteracyjnego

Źródło: Mitchell Luo, domena publiczna.

> W tym e-materiale powtarzamy wiadomości ze szkoły podstawowej.

Wiesz już, czym charakteryzuje się podejście rekurencyjne, a czym iteracyjne. Czy potrafisz – na podstawie analizowanego problemu – wskazać, które z nich należy zastosować?

Więcej informacji o iteracji i rekurencji znajdziesz w e-materiałach:

- Algorytmy iteracyjne,
- Rekurencja,
- Rekurencja a iteracja,
- Algorytmy iteracyjne i liczbowe – potęgowanie liczb.

O tym, jak zagadnienie rekurencji wyjaśnia matematyka, przeczytasz w e-materiałach:

- Ciąg określony rekurencyjnie,
- Ciąg geometryczny określony rekurencyjnie,
- Wzór ogólny ciągu określonego rekurencyjnie,
- Ciąg arytmetyczny określony wzorem rekurencyjnym.

Analizę algorytmów iteracyjnych i rekurencyjnych w wybranych językach programowania znajdziesz w e-materiałach:

- Analiza podejścia rekurencyjnego i iteracyjnego w języku C++,
- Analiza podejścia rekurencyjnego i iteracyjnego w języku Java,
- Analiza podejścia rekurencyjnego i iteracyjnego w języku Python.

Więcej zadań? Przejdź do e-materiału Analiza podejścia rekurencyjnego i iteracyjnego – zadania maturalne.

> **Twoje cele**
>
> - Porównasz dwie techniki programowania - rekurencyjną i iteracyjną.
> - Przeanalizujesz algorytmy obliczające wartość potęgi techniką rekurencyjną i iteracyjną.
> - Zaimplementujesz algorytmy obliczające wartość silni dwiema technikami: rekurencyjną i iteracyjną.

Przeczytaj

Rekurencja a iteracja

Przypomnijmy sobie najważniejsze wiadomości na temat rekurencji i iteracji.

Iteracja to technika programowania polegająca na wielokrotnym wykonywaniu danego ciągu instrukcji. W programie iterację realizujemy za pomocą instrukcji iteracyjnych – czyli pętli. Instrukcje w pętli wykonywane są do momentu, gdy zostanie spełniony warunek wykonywania pętli.

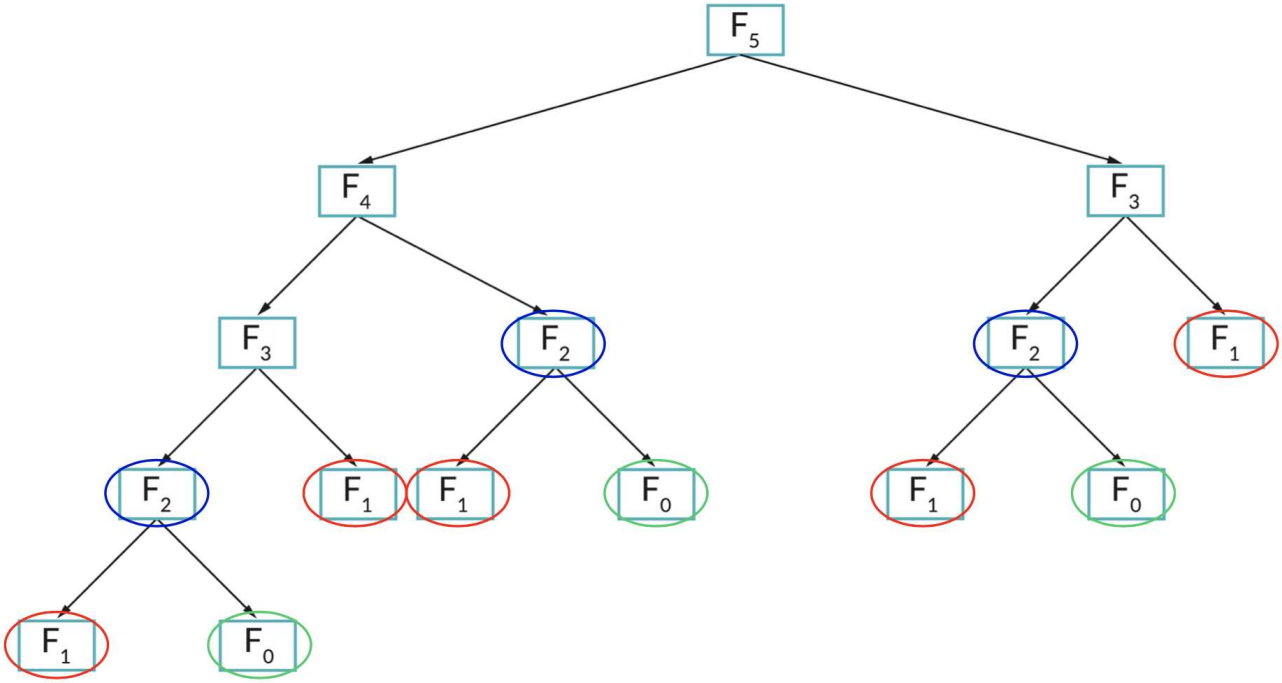
W przypadku rekurencji funkcja wywołuje samą siebie ze zmienionym parametrem. Parametr ten określa poziomy wywołań rekurencyjnych funkcji, to znaczy stopień zagłębienia się wywołań. Gdy zostaje spełniony warunek początkowy rekurencji, zwracana jest konkretna wartość. Dzięki temu ciąg wywołań rekurencyjnych ma swój koniec. Gdy funkcja się zakończy lub zwróci wartość, następuje powrót do poprzedniego poziomu (w miejsce po wywołaniu funkcji) i wykonywane są kolejne zapisane instrukcje.

Jakie są zatem **wady i zalety** tych dwóch metod - rekurencyjnej i iteracyjnej? Sprawdźmy, którą z nich zastosować w jakiej sytuacji.

Którą technikę wybrać?

Mimo że użycie rekurencji przy rozwiązywaniu niektórych problemów zapewnia krótki i czytelny kod, to algorytmy iteracyjne często są bardziej wydajne pod względem czasu i pamięci.

Stosowanie rekurencji w wielu przypadkach jest nieefektywne, często powtarzane są wiele razy te same obliczenia. Na przykład rekurencyjny algorytm obliczania elementu ciągu Fibonacciego (przedstawiony w e-materiale Ciąg Fibonacciego) ma złożoność $O(2^n)$, podczas gdy złożoność algorytmu iteracyjnego to $O\left(n\right)$. Wynika to z faktu, że w celu wyznaczenia wybranego elementu ciągu metodą rekurencyjną wielokrotnie są obliczane te same, poprzedzające go wartości:



Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

Ponadto użycie rekurencji wymaga zajęcia dużej ilości pamięci. Jest to spowodowane koniecznością rezerwacji miejsca na zmienne, argumenty wywołania i wartości zwracane z każdym wywoływaniem funkcji. To sprawia, że program w wersji rekurencyjnej może być znacznie wolniejszy od tego napisanego metodą iteracyjną.

Aby porównać te dwie techniki programowania, napiszmy funkcje obliczające potęgę danej liczby, korzystając z metody iteracyjnej, a następnie rekurencyjnej. Zacznijmy od podejścia iteracyjnego.

Obliczanie potęgi $podstawa^{wykładnik}$ metodą iteracyjną

Specyfikacja:

Dane:

- `wykładnik` – wykładnik potęgi, liczba naturalna
- `podstawa` – podstawa potęgi, liczba całkowita

Wynik:

- `wynik` – wynik potęgowania $podstawa^{wykładnik}$, liczba całkowita

Oto funkcja zapisana za pomocą pseudokodu w wersji iteracyjnej:

```
funkcja iteracyjnie(podstawa, wykładnik)
    wynik ← 1
    dopóki wykładnik > 0 wykonuj
```

```
        wynik ← wynik * podstawa
        wykładnik ← wykładnik - 1
    zwróć wynik
```

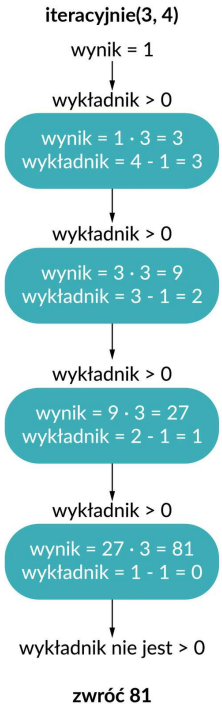
Parametrami przyjmowanymi przez funkcję są `podstawa`, czyli element potęgowany, oraz jej `wykładnik`.

Wynik potęgowania przechowywany jest w zmiennej `wynik`. Inicjujemy ją wartością 1, dzięki czemu bezpośrednio otrzymamy wynik dla wykładnika równego 0.

Dopóki `wykładnik` jest większy od 0, dopóty `wynik` mnożymy przez podstawę. W ten sposób otrzymamy iloczyn tylu czynników `podstawa`, ile wynosi `wykładnik`.

Na koniec zwracany jest `wynik`, w którym zapisany będzie wynik potęgowania.

Zaprezentowany schemat przedstawia podejście iteracyjne obliczania potęgi danej liczby dla podstawy równej 3 i wykładnika równego 4. Przedstawione tu zostały kolejne iteracje pętli.



> **Ważne!**
>
> Przedstawiony iteracyjny algorytm obliczania potęgi według definicji jest algorytmem o relatywnie niskiej złożoności czasowej. Istnieje jednak efektywniejsze rozwiązanie iteracyjne wykorzystujące schemat Hornera - szybkie potęgowanie liczb. Omówienie tego algorytmu znajdziesz w e-materiale Algorytmy iteracyjne i liczbowe – potęgowanie liczb.

Obliczanie potęgi $podstawa^{wykładnik}$ metodą rekurencyjną

Specyfikacja:

Dane:

- `wykładnik` – wykładnik potęgi, liczba naturalna
- `podstawa` – podstawa potęgi, liczba całkowita

Wynik:

- `wynik` – wynik potęgowania $podstawa^{wykładnik}$, liczba całkowita

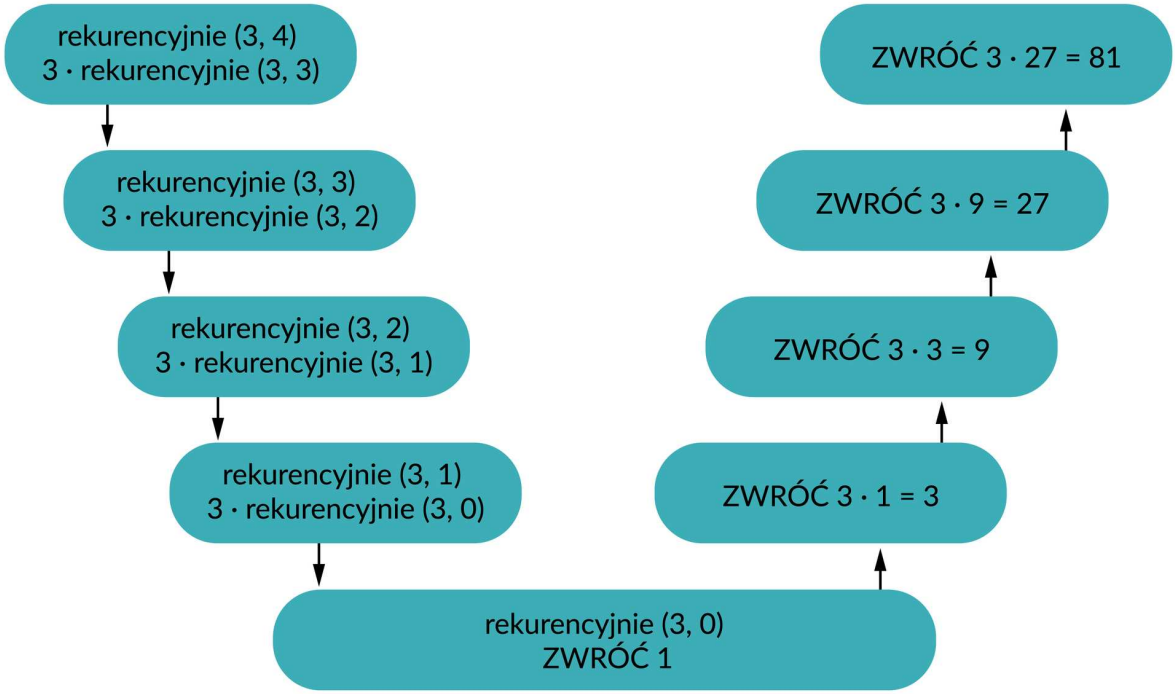
Oto zapisana za pomocą pseudokodu funkcja realizująca potęgowanie techniką rekurencyjną:

```
funkcja rekurencyjnie(podstawa, wykładnik)
    jeżeli wykładnik = 0
        zwróć 1
    zwróć podstawa * rekurencyjnie(podstawa, wykładnik - 1)
```

Funkcja `rekurencyjnie()` przyjmuje dwa parametry - `podstawa` oraz `wykładnik`. Jeżeli `wykładnik` jest równy 0, od razu zwracana jest wartość 1. Dzieje się tak dlatego, że niezależnie od podstawy liczba podniesiona do potęgi 0 daje wynik równy 1.

W przeciwnym wypadku mnożymy podstawę przez wynik potęgowania, którego wykładnik zmniejszony jest o 1. Oto matematyczny przykład dla 3^4:

$$3^4 = 3 \cdot 3^3$$



Funkcja wywołuje rekurencyjnie samą siebie z drugim parametrem zmniejszonym o 1. Jeżeli parametr ten wynosi 0, spełniony zostaje warunek początkowy rekurencji. Funkcja nie wywołuje wtedy rekurencyjnie kolejnej funkcji, a zwraca konkretną wartość, w tym przypadku 1. Zwrócona wartość (pobrana ze stosu) zostaje przypisana jako wartość funkcji w miejscu jej wywołania i wykonywane są obliczenia, a przeznaczony na nią obszar pamięci na stosie zostaje zwolniony.

Szybkie potęgowanie liczb – definicja rekurencyjna

Istnieje szybszy sposób potęgowania liczb od potęgowania według definicji. Jest on oparty na schemacie Hornera.

Zwróćmy uwagę, że potęgowanie 2^6 według definicji:

$$2^6 = 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2$$

możemy obliczyć inną metodą. Przedstawia się ona następująco:

$$2^6 = \left(2^3\right)^2 = \left(2 \cdot 2^2\right)^2$$

Stosując tę metodę, zamiast pięciu mnożeń wykonalibyśmy trzy. Oszczędność liczby mnożeń wydaje się niewielka, jednak dla większych wykładników znacząco skracamy czas przeprowadzania tego działania. Załóżmy, że znamy wartość potęgi 2^{25}. Wtedy chcąc wykonać działanie 2^{50}, dokonujemy tylko jednego mnożenia!

$$2^{50} = \left(2^{25}\right)^2$$

Obliczając tę potęgę zgodnie z powszechnie znaną definicją, znając wartość 2^{25}, musielibyśmy wykonać jeszcze 25 dodatkowych mnożeń.

$$2^{50} = 2^{25} \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot \ldots \cdot 2$$

Opierając się na powyższej analizie, możemy ustalić definicję rekurencyjną szybkiego potęgowania liczb ($a \neq 0$):

$$a^n = \begin{cases} 1 & \text{dla } n = 0 \\ a \cdot a^{n-1} & \text{dla } n \bmod 2 = 1 \\ (a^{n/2})^2 & \text{dla } n \bmod 2 = 0 \end{cases}$$

Na podstawie definicji konstruujemy algorytm rekurencyjny:

Specyfikacja problemu:

Dane:

- `n` – wykładnik potęgi, liczba naturalna
- `a` – podstawa potęgi, liczba całkowita różna od zera

Wynik:

- `wynik` – wynik potęgowania a^n, liczba całkowita

```
funkcja potęgowanie(a, n)
    jeżeli n = 0
        zwróć 1
    jeżeli n mod 2 = 1
        zwróć a * potęgowanie(a, n-1)
    w przeciwnym razie
        potęga ← potęgowanie(a, n/2)
    zwróć potęga * potęga
```

> **Ważne!**
>
> Zaprezentowana metoda ma swoją wersję iteracyjną. Jej omówienie znajdziesz w e-materiale Algorytmy iteracyjne i liczbowe – potęgowanie liczb

Analiza złożoności czasowej

Złożoność czasowa algorytmu potęgowania liczb według następującej definicji:

$$a^n = \underbrace{a \cdot a \cdot \ldots \cdot a}_{n}$$

wyrażona jako liczba wykonanych mnożeń wynosi $O(n)$ - zarówno w wersji iteracyjnej jak i rekurencyjnej.

W przypadku algorytmu szybkiego potęgowania możemy zauważyć, że każde pojedyncze lub każde dwa kolejne wywołania rekurencyjne funkcji zmniejszają wykładnik dwukrotnie. Zatem zostanie wykonanych maksymalnie $2 \log_2 n$ mnożeń. Złożoność algorytmu szybkiego potęgowania liczb wynosi więc $O(\log_2 n)$.

Słownik

iteracja

technika programowania polegająca na wielokrotnym powtarzaniu zapisanego ciągu instrukcji aż do spełnienia określonego warunku

rekurencja

technika programowania, w której funkcja wywołuje samą siebie aż do napotkania przypadku podstawowego, czyli takiego, którego nie da się rozłożyć na mniejsze problemy; przykład przypadku podstawowego dla obliczania piątego wyrazu ciągu Fibonacciego:

$$fib(4) = fib(3) + fib(2) = (fib(2) + fib(1)) + (fib(1) + fib(0)) =$$

$$= ((fib(1) + fib(0)) + fib(1)) + fib(1)) + (fib(1) + fib(0)) =$$

$$= ((1 + 0) + 1) + (1 + 0) = 3$$

warunek początkowy rekurencji

podana wprost wartość definiowanej wielkości, niezbędna, aby rekurencja się zakończyła

Schemat interaktywny

Polecenie 1

Za pomocą schematu interaktywnego przygotuj algorytm obliczający silnię z liczby n **techniką rekurencyjną**. Program przetestuj dla n równego 6.

Specyfikacja:

Dane:

- n – liczba naturalna

Wynik:

- `wynik` – wartość silni zadanej liczby n, liczba naturalna

Polecenie 2

Za pomocą schematu interaktywnego przygotuj algorytm obliczający silnię z liczby n **techniką iteracyjną**. Program przetestuj dla n równego 7.

Specyfikacja:

Dane:

- n – liczba naturalna

Wynik:

- `wynik` – wartość silni zadanej liczby n, liczba naturalna

Sprawdź się

Pokaż ćwiczenia: 

Ćwiczenie 1

Przeanalizuj specyfikację i algorytm zapisany w pseudokodzie, a następnie wykonaj polecenie.

Specyfikacja:

Dane:

- `wykładnik` – wykładnik potęgi; liczba naturalna
- `podstawa` – podstawa potęgi; liczba całkowita

Wynik:

- `wynik` – wynik potęgowania $podstawa^{wykładnik}$; liczba całkowita

```
funkcja iteracyjnie(podstawa, wykładnik)
    wynik ← 1
    dopóki wykładnik > 0 wykonuj
        wynik ← wynik * podstawa
        wykładnik ← wykładnik - 1
    zwróć wynik
```

Wskaż, ile iteracji pętli zostanie wykonanych przy wywołaniu funkcji `iteracyjnie(5, 8)`.

- ◯ 5
- ◯ 6
- ◯ 7
- ◯ 8

Ćwiczenie 2

Przeanalizuj specyfikację i algorytm zapisany w pseudokodzie, a następnie wykonaj polecenie.

Specyfikacja:

Dane:

- `podstawa` – podstawa potęgi; liczba całkowita
- `wykładnik` – wykładnik potęgi; liczba naturalna

Wynik:

- `wynik` – wynik potęgowania $podstawa^{wykładnik}$; liczba całkowita

```
funkcja rekurencyjnie(podstawa, wykładnik)
    jeżeli wykładnik = 0
        zwróć 1
    w przeciwnym razie
        zwróć podstawa * rekurencyjnie(podstawa, wykładnik - 1)
rekurencyjnie(11, 12)
```

Wskaż, ile razy zostanie wywołana funkcja `rekurencyjnie()`.

- ◯ 12
- ◯ 11
- ◯ 10
- ◯ 13

Ćwiczenie 3

Zapisz za pomocą pseudokodu algorytm wyznaczający sumę n kolejnych liczb naturalnych dodatnich metodą rekurencyjną.

Specyfikacja:

Dane:

- n – liczba naturalna dodatnia

Wynik:

- `suma` – suma n kolejnych liczb naturalnych dodatnich; liczba naturalna

```
1
```

Ćwiczenie 4

Zapisz za pomocą pseudokodu algorytm wyznaczający iloczyn n kolejnych dodatnich liczb naturalnych metodą iteracyjną.

Specyfikacja:

Dane:

- n – liczba naturalna dodatnia

Wynik:

- `wynik` – iloczyn n kolejnych dodatnich liczb naturalnych; liczba naturalna dodatnia

```
1
```

Ćwiczenie 5

Przyporządkuj odpowiednie cechy do podejścia rekurencyjnego i iteracyjnego.

podejście rekurencyjne

podejście iteracyjne

- większy rozmiar kodu programu
- jeden podproblem może być niepotrzebnie wielokrotnie rozwiązywany
- wymaga zapamiętania wielu informacji niezbędnych do odtworzenia stanu sprzed wywołania
- w treści funkcji wywoływana jest ta sama funkcja ze zmienionym parametrem
- używana jest zmienna sterująca
- mniejszy rozmiar kodu programu
- polega na powtarzaniu w pętli tych samych instrukcji
- zakończenie następuje po spełnieniu warunku początkowego

Ćwiczenie 6

Wskaż zdania prawdziwe. Zaznacz wszystkie poprawne odpowiedzi.

- [ ] Rekurencję zrealizujemy jedynie za pomocą instrukcji pętli.
- [ ] Gdy funkcja rekurencyjna zwróci wartość pobraną ze stosu, przeznaczony na nią obszar pamięci na stosie zostaje zwolniony.
- [ ] Parametr funkcji rekurencyjnej określa poziomy wywołań rekurencyjnych funkcji, czyli stopień zagłębienia się wywołań.
- [ ] Stosowanie iteracji wymaga zazwyczaj większej ilości pamięci w porównaniu do podejścia rekurencyjnego.

Ćwiczenie 7

Zdefiniowany jest następujący ciąg:

$$a_n = \begin{cases} 2 & \text{gdy} \quad n < 3 \\ a_{n-2} \cdot a_{n-1} & \text{gdy} \quad n \mod 2 = 0 \\ a_{n-1} \mod 4 & \text{gdy} \quad n \mod 2 = 1 \end{cases}$$

Zapoznaj się ze specyfikacją oraz zapisaną w pseudokodzie funkcją rekurencyjną realizująca algorytm obliczania wartości n-tego elementu zdefiniowanego ciągu, a następnie wykonaj polecenie.

Specyfikacja:

Dane:

- n – indeks elementu ciągu; liczba naturalna dodatnia

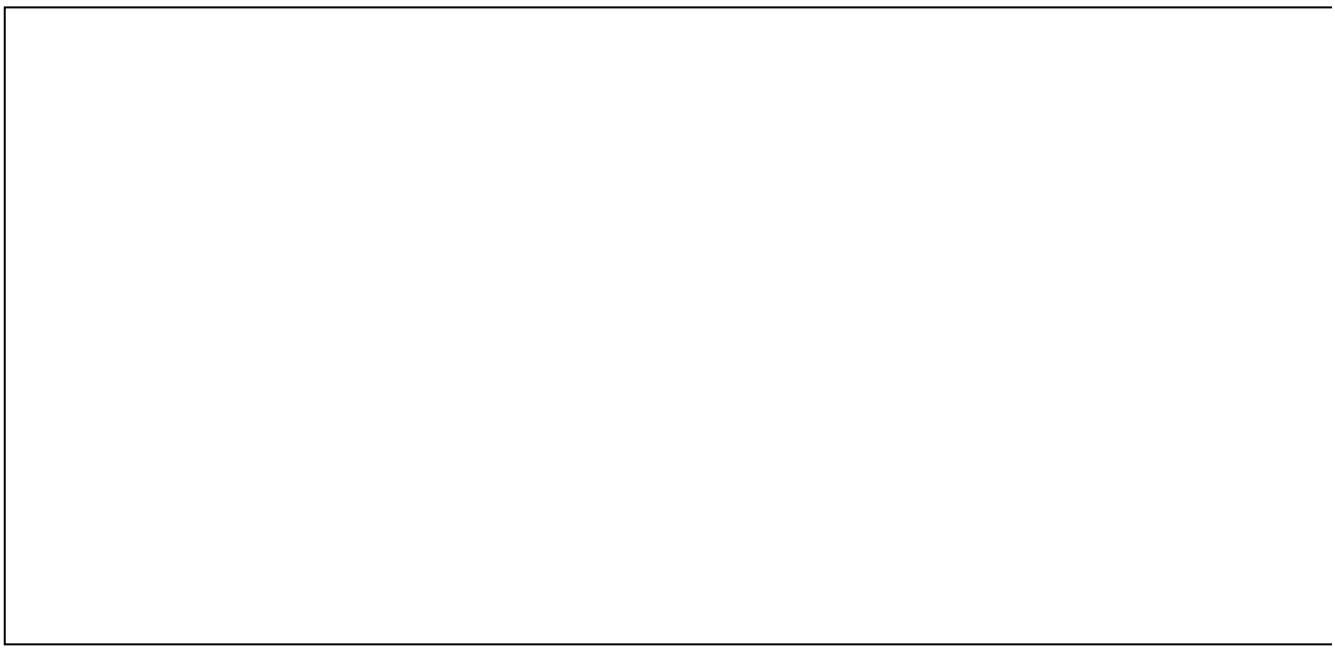
Wynik:

- a_n – wartość n-tego elementu zdefiniowanego ciągu; liczba naturalna dodatnia

```
funkcja rekurencyjna(n)
    jeżeli n < 3
        zwróć 2
    w przeciwnym razie
        jeżeli n mod 2 = 0
            zwróć rekurencyjna(n - 2) * rekurencyjna(n - 1)
        w przeciwnym razie
            zwróć rekurencyjna(n - 1) mod 4
```

Na podstawie przedstawionej funkcji napisz własną realizującą ten sam algorytm, używając podejścia iteracyjnego.

```
1
```



Ćwiczenie 8

Przeanalizuj algorytm zapisany w pseudokodzie, a następnie wykonaj polecenie.

```
funkcja rekurencyjna(n)
    jeżeli n < 2
        zwróć 2
    w przeciwnym razie
        jeżeli n mod 2 = 0
            zwróć rekurencyjna(n - 2) * rekurencyjna(n - 1)
        w przeciwnym razie
            zwróć rekurencyjna(n - 1) mod 4
```

Wskaż, ile razy zostanie wywołana funkcja `rekurencyjna()` dla n równego 7.

- ◯ 8
- ◯ 17
- ◯ 10
- ◯ 19

Dla nauczyciela

Autor: Maurycy Gast
Przedmiot: Informatyka

Temat: Analiza podejścia rekurencyjnego i iteracyjnego

Grupa docelowa:

Szkoła ponadpodstawowa, liceum ogólnokształcące, technikum, zakres podstawowy i rozszerzony

Podstawa programowa:

Cele kształcenia – wymagania ogólne

I. Rozumienie, analizowanie i rozwiązywanie problemów na bazie logicznego i abstrakcyjnego myślenia, myślenia algorytmicznego i sposobów reprezentowania informacji.

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera oraz innych urządzeń cyfrowych: układanie i programowanie algorytmów, organizowanie, wyszukiwanie i udostępnianie informacji, posługiwanie się aplikacjami komputerowymi.

Treści nauczania – wymagania szczegółowe

I. Rozumienie, analizowanie i rozwiązywanie problemów.

Zakres podstawowy. Uczeń:

2) stosuje przy rozwiązywaniu problemów z różnych dziedzin algorytmy poznane w szkole podstawowej oraz algorytmy:

e) obliczania wartości elementów ciągu metodą iteracyjną i rekurencyjną, w tym wartości elementów ciągu Fibonacciego.

Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:

8) dyskutuje na temat roli myślenia komputacyjnego i jego metod, takich jak: abstrakcja, reprezentacja danych, dekompozycja problemu, redukcja, myślenie rekurencyjne, podejście heurystyczne w rozwiązywaniu problemów z różnych dziedzin.

I + II. Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:

3) objaśnia, a także porównuje podstawowe metody i techniki algorytmiczne oraz struktury danych, wykorzystując przy tym przykłady problemów i algorytmów, w szczególności:

b) rekurencję (do generowania ciągów liczb, potęgowania, sortowania liczb, generowania fraktali),

Kształtowane kompetencje kluczowe:

- kompetencje cyfrowe;
- kompetencje osobiste, społeczne i w zakresie umiejętności uczenia się;
- kompetencje matematyczne oraz kompetencje w zakresie nauk przyrodniczych, technologii i inżynierii.

Cele operacyjne (językiem ucznia):

- Porównasz dwie techniki programowania – rekurencyjną i iteracyjną.
- Przeanalizujesz algorytmy obliczające wartość potęgi techniką rekurencyjną i iteracyjną.
- Zaimplementujesz algorytmy obliczające wartość silni dwiema technikami: rekurencyjną i iteracyjną.

Strategie nauczania:

- konstruktywizm;
- konektywizm.

Metody i techniki nauczania:

- dyskusja;
- rozmowa nauczająca z wykorzystaniem multimedium i ćwiczeń interaktywnych;
- ćwiczenia praktyczne.

Formy pracy:

- praca indywidualna;
- praca w parach;
- praca w grupach;
- praca całego zespołu klasowego.

Środki dydaktyczne:

- komputery z głośnikami, słuchawkami i dostępem do internetu;

- zasoby multimedialne zawarte w e-materiale;
- tablica interaktywna/tablica, pisak/kreda.

Przebieg lekcji

Przed lekcją:

1. Uczniowie przypominają sobie charakterystykę rekurencji oraz iteracji.
2. **Przygotowanie do zajęć.** Nauczyciel loguje się na platformie i udostępnia e-materiał: „Analiza podejścia rekurencyjnego i iteracyjnego". Uczniowie mają zapoznać się z treściami w sekcji „Przeczytaj".

Faza wstępna:

1. Nauczyciel wyświetla temat i cele zajęć zawarte w sekcji „Wprowadzenie". Prosi uczniów, by na podstawie wiadomości zdobytych przed lekcją zaproponowali kryteria sukcesu.

Faza realizacyjna:

1. **Praca z tekstem.** Nauczyciel ocenia, na podstawie informacji na platformie, stan przygotowania uczniów do zajęć. Jeżeli jest ono niewystarczające, prosi wybraną osobę o przedstawienie najważniejszych informacji z sekcji „Przeczytaj". Następnie uczniowie analizują przykłady z sekcji „Przeczytaj". Próbują na swoich komputerach przełożyć je na języki, w których programują.
2. **Praca z multimedium.** Nauczyciel wyświetla zawartość sekcji „Schemat interaktywny", czyta treść polecenia nr 1: „Za pomocą schematu interaktywnego przygotuj algorytm obliczający silnię z liczby n techniką rekurencyjną. Program przetestuj dla n równego 6" i omawia kolejne kroki rozwiązania.
3. Nauczyciel dzieli uczniów na grupy czteroosobowe. Uczniowie wykonują ćwiczenia nr 1-8 z sekcji „Sprawdź się", a następnie porównują swoje odpowiedzi z inną grupą.

Faza podsumowująca:

1. Nauczyciel zadaje pytania podsumowujące, np.:
    - co oznacza pojęcie rekurencja?
    - czym jest iteracja?
    - czym jest przypadek podstawowy?
2. Nauczyciel wyświetla na tablicy temat lekcji i cele zawarte w sekcji „Wprowadzenie". W kontekście ich realizacji podsumowuje przebieg zajęć, a także wskazuje mocne i słabe strony pracy uczniów.

Praca domowa:

1. Uczniowie wykonują polecenie nr 2 z sekcji „Schemat interaktywny".

Wskazówki metodyczne:

- Treści w sekcji „Przeczytaj” można wykorzystać jako podsumowanie i utrwalenie wiedzy uczniów.