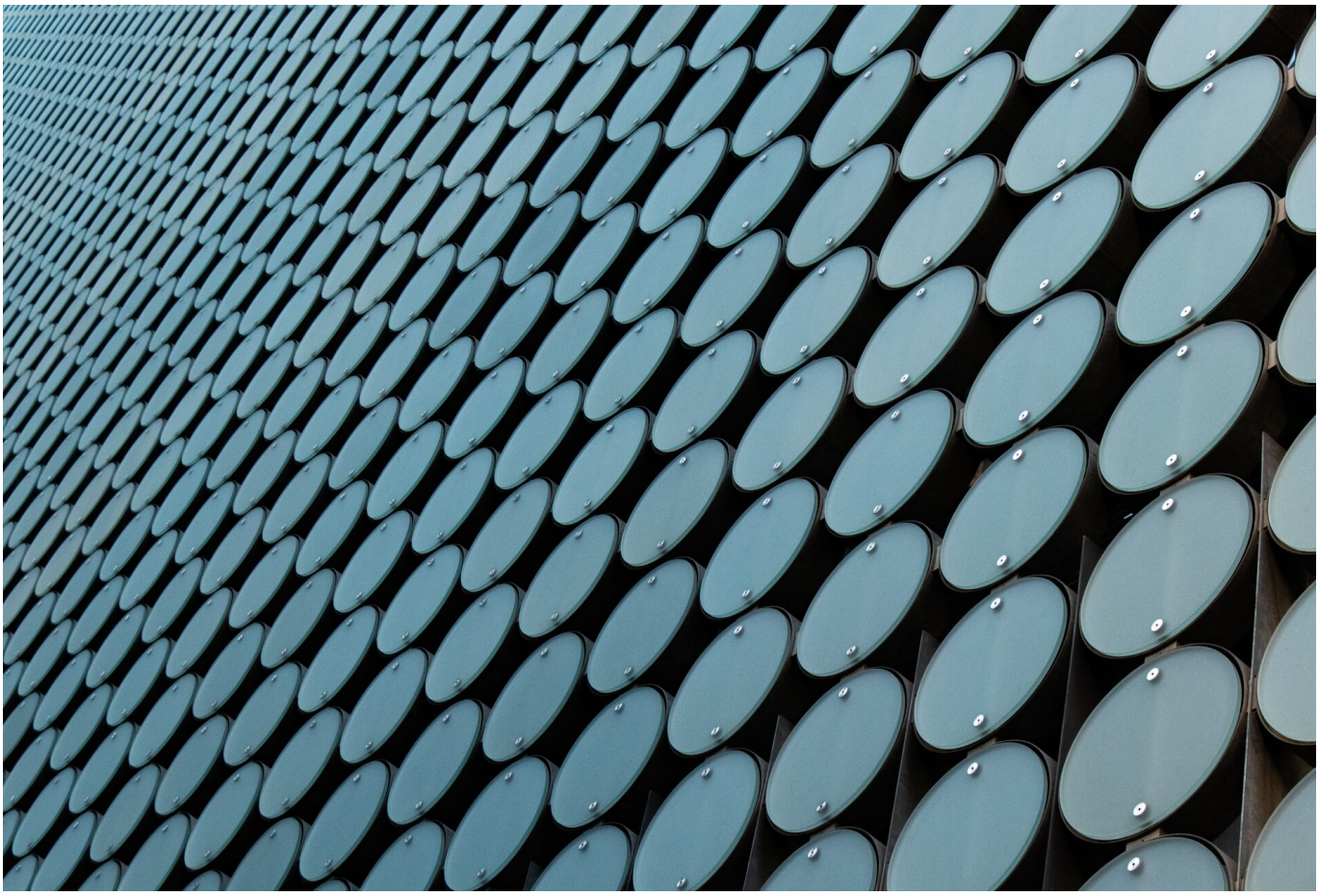


Analiza podejścia rekurencyjnego i iteracyjnego w języku C++

Analiza podejścia rekurencyjnego i iteracyjnego w języku C++

Źródło: Mitchell Luo, domena publiczna.

Wiesz już, czym charakteryzuje się podejście rekurencyjne, a czym iteracyjne. Czy potrafisz jednak na podstawie analizowanego problemu wskazać, które podejście wykorzystać? W tym e-materiale omówimy różnice w ich implementacji w języku C++.

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 pozostałych językach programowania znajdziesz w e-materiałach:

- 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**

- Przeanalizujesz rekurencyjny i iteracyjny sposób rozwiązywania problemów w języku C++.
- Porównasz rozwiązania iteracyjne i rekurencyjne tego samego problemu.
- Zaimplementujesz w języku w C++ program obliczający potęgę liczby.

Przeczytaj

Obliczanie wartości potęgi o podstawie i wykładniku naturalnym

Specyfikacja problemu:

Dane:

- `p` – podstawa potęgi; liczba naturalna, `p > 0`
- `w` – wykładnik potęgi; liczba naturalna

Wynik:

- `wynik` – wynik potęgowania; liczba naturalna

Sposób iteracyjny

Napiszemy funkcję obliczającą wartość potęgi o wykładniku naturalnym. Zaczniemy od nadania jej nazwy. Będziemy również potrzebować dwóch parametrów – jeden z nich będzie odpowiadać za wartość podstawy, a drugi za wartość wykładnika. Nazwiemy je odpowiednio `p` i `w`. Obu parametrom nadamy typ `int`.

```
1 int iteracja(int p, int w)
2 {
3 }
```

Teraz w ciele funkcji zadeklarujemy zmienną `wynik` o typie całkowitym. Przypiszemy jej wartość: `1`. Następnie utworzymy pętlę `while`, która będzie wykonywana, dopóki `w` jest większe od zera.

```
1 int wynik = 1;
2 while (w > 0)
3 {
4 }
```

W środku pętli będziemy mnożyć zmienną `wynik` przez `p`, a następnie dekrementować zmienną `w`.

```
wynik *= p;
w--;
```

Skoro w zmiennej `wynik` mamy już wyliczoną odpowiednią wartość, możemy ją zwrócić.

```
return wynik;
```

W ten sposób udało się nam napisać funkcję potęgującą sposobem iteracyjnym.

Gotowa funkcja wygląda następująco:

```
int iteracja(int p, int w)
{
    int wynik = 1;
    while (w > 0)
    {
        wynik *= p;
        w--;
    }
    return wynik;
}
```

Sposób rekurencyjny

Napiszmy funkcję obliczającą wartość potęgi metodą rekurencyjną. Funkcja będzie miała dwa parametry typu `int`, przechowujące informacje: podstawę i wykładnik potęgi. Nazwiemy je odpowiednio `p` i `w`.

```
int rekurencja(int p, int w)
{
}
```

We wnętrzu funkcji rozważymy dwa przypadki. Pierwszy to warunek przerywania rekurencji (warunek podstawowy), który zwróci wartość i zakończy wykonywanie funkcji. Do jego zdefiniowania użyjemy instrukcji warunkowej `if`, gdzie w teście logicznym sprawdzimy, czy zmienna `w` jest równa 0. W sytuacji, gdy taka zależność będzie zachodzić, funkcja zwróci 1. W pozostałych przypadkach funkcja zwróci iloczyn zmiennej `p` i siebie samej z argumentami `p` i `w - 1`.

```
if (w == 0) return 1;
else return p * rekurencja(p, w - 1);
```

Tym sposobem napisaliśmy funkcję potęgującą z wykorzystaniem rekurencji.

```
int rekurencja(int p, int w)
{
    if (w == 0) return 1;
    else return p * rekurencja(p, w - 1);
}
```

Jak możemy łatwo zauważyć, wymagała ona napisania znacznie krótszego kodu w porównaniu do wariantu iteracyjnego.

Dla pełnego zrozumienia sposobu działania funkcji rekurencyjnej przeanalizujemy działanie tej funkcji dla podstawy równej `2` i wykładnika o wartości `3`.

1. Zaczynamy ze zmienną `p = 2` i `w = 3`. Ponieważ `w` nie jest równe `0`, wywołujemy `p * rekurencja (p, w - 1)`, czyli `2 * rekurencja (2, 2)`.
2. W wywołaniu funkcji `rekurencja (2, 2)`, w dalszym ciągu `w` nie równa się `0`, więc ponownie wywołujemy funkcję: `2 * rekurencja (2, 1)`.
3. W przypadku wywołania funkcji `rekurencja (2, 1)`, `w` nadal nie równa się `0`, wywołujemy zatem po raz ostatni funkcję – tym razem będzie to `2 * rekurencja (2, 0)`.

Funkcja `rekurencja (2, 0)` zwróci nam wartość `1`. Teraz, gdy mamy już wartość z warunku podstawowego (początkowego), możemy obliczyć wszystkie użyte wywołania funkcji `potęga`. Składają się one razem na odpowiedź na zadane pytanie o wartość liczby `2` podniesionej do potęgi `3`.

1. `rekurencja (2, 1)` to `2 * rekurencja (2, 0)`, czyli 2.
2. `rekurencja (2, 2)` to `2 * rekurencja (2, 1)`, czyli 4.
3. `rekurencja (2, 3)` to `2 * rekurencja (2, 2)`, czyli 8.

Tym samym udało nam się przeanalizować działanie funkcji rekurencyjnej `rekurencja` oraz otrzymać poprawny wynik, czyli `8`.

Obliczanie silni

Specyfikacja problemu:

Dane:

- `liczba` – liczba naturalna

Wynik:

- `silnia` – iloczyn liczb od 1 do `liczba`; liczba naturalna

Sposób iteracyjny

```
int silnia_iteracyjna(int liczba)
{
    long long silnia = 1;

    for (int i = 1; i <= liczba; i++)

        silnia *= i;

    return silnia;
}
```

Sposób rekurencyjny

```
int silnia_rekurencyjna (int liczba)
{
    if (liczba < 2)
    {
        return 1;
    }
    return liczba * silnia_rekurencyjna(liczba - 1);
}
```

Słownik

argument funkcji

element składni w określonym języku programowania, który w wyniku wywołania podprogramu zostaje utożsamiony (skojarzony) z określonym parametrem podprogramu
dekrementacja

zmniejszenie wartości zmiennej o jeden
iteracja

powtarzanie tej samej operacji określoną liczbę razy lub do momentu, w którym zadany warunek zostanie spełniony

parametr funkcji

element składni w określonym języku programowania; umożliwia komunikację między podprogramem (funkcją) wywołanym a programem wywołującym; parametry określa się w nagłówku podprogramu (przy jego definicji)

rekurencja

technika programowania polegająca na odwoływaniu się procedur lub funkcji do samych siebie, aż do momentu spełnienia warunku podstawowego

stos

liniowa struktura danych, której dane są pobierane zgodnie z zasadą LIFO (od ang. *Last-In First-Out* - dane dołączone jako ostatnie są pobierane jako pierwsze)

warunek początkowy (podstawowy)

warunek, który spełniony nie spowoduje wywołania funkcji rekurencyjnej, tylko zwróci konkretną wartość

Prezentacja multimedialna

Problem 1

Napisz program w języku C++, który dla danych liczb naturalnych p, w oraz n obliczy wartość wyrażenia p^w, gdzie p i w użytkownik podaje z klawiatury na standardowe wejście. Sposób obliczenia wyrażenia będzie zależał od parametru n tzn. dla n > p program obliczy p^w iteracyjne, a dla n ≤ p rekurencyjnie.

Przetestuj działanie programu n = 100, p = 5 oraz w = 3.

Specyfikacja problemu:

Dane:

- p – liczba naturalna, p > 0
- w – liczba naturalna
- n – liczba naturalna; liczba wykorzystana do określenia sposobu, w jaki ma zostać obliczona potęga

Wynik:

- p^w – liczba naturalna

```cpp
#include <iostream>

using namespace std;

int iteracja(int p, int w)
{
    // tutaj dopisz kod
}

long long rekurencja(int p, int w)
{
    // tutaj dopisz kod
}

```

```
1
```

Polecenie 1

Porównaj swoje rozwiązanie z przedstawionym w prezentacji.

1

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PtA31hjm7

Napiszmy program, który pobierze od użytkownika podstawę oraz wykładnik, a następnie poda wartość podstawy podniesionej do potęgi o tym wykładniku.

Do obliczania wartości potęgi dla podstaw mniejszych niż n program ma używać funkcji iteracyjnej, a w pozostałych przypadkach – funkcji rekurencyjnej.

2

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PtA31hjm7

Zacznijmy od zdefiniowania funkcji iteracyjnej.

Użytkownik może podać dowolne wartości podstawy oraz wykładnika. Jakimi typami zmiennych powinniśmy zatem operować? W sytuacji, gdy podstawa jest liczbą całkowitą, a wykładnik naturalną, pierwszym wyborem będzie najczęściej typ `int`. Musimy jednak pamiętać o ograniczonej wielkości liczby przechowywanej przez ten typ. Dlatego w tym przypadku zalecane byłyby wartości `long` lub `long long`.

Podstawa może być jednak liczbą rzeczywistą – wtedy potrzebny byłby typ przechowujący liczby zmiennoprzecinkowe. Tu znów, ze względu na możliwą wielkość wyniku, najbezpieczniej byłoby wykorzystać typy `double` lub `long double`. Jeśli jednak specyfikacja zawęża zakres podstawy i wykładnika do niewielkich wartości, wystarczy wykorzystanie typu `float`.

Funkcja przyjmie dwie zmienne: `p` oraz `w`. W omawianym tu przypadku zakładamy, że operujemy na niewielkich wartościach podstawy (zmienna `p`) oraz wykładnika (zmienna `w`), więc zmienne są typu `int`.

```
1 int iteracja(int p, int w)
2 {
3 }
```

Materiał audio dostępny pod adresem:

3

https://zpe.gov.pl/b/PtA31hjm7

Zdefiniujmy zmienną typu `int`, której przypiszemy wartość 1. W dalszej części funkcji wartość ta będzie mnożona przez podstawę odpowiednią liczbę razy w celu uzyskania właściwego wyniku.

```
int odp = 1;
```

4

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PtA31hjm7

Następnie wstawiamy do funkcji pętlę `while`, która będzie wykonywana, dopóki `w` (czyli wartość wykładnika) jest większe od zera.

```
while (w > 0)
{
}
```

5

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PtA31hjm7

W każdym przejściu pętli `while` mnożymy zmienną `odp` przez `p` i dekrementujemy zmienną `w`, odpowiadającą za wartość wykładnika.

```
odp *= p;
w--;
```

6

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PtA31hjm7

Po wykonaniu pętli zmienna `odp` powinna przechowywać wartość podstawy podniesionej do odpowiedniej potęgi, która jest wynikiem działania funkcji.

Funkcja zwróci więc zmienną `odp`.

```
return odp;
```

7

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PtA31hjm7

Teraz napiszmy funkcję obliczającą wartość potęgi metodą rekurencyjną.

Ponieważ funkcja przyjmować będzie wyłącznie podstawy o wartości większej lub równej `n`, a `n` może mieć dowolną wartość podaną przez użytkownika, funkcja zwracać będzie typ `long long`. Funkcja, tak jak jej postać iteracyjna, jako argument przyjmować będzie dwie zmienne typu `int`.

```
long long rekurencja (int p, int w)
{
}
```

8

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PtA31hjm7

Wewnątrz funkcji standardowo zastosujemy instrukcję warunkową `if… else`. Jeżeli `w`, czyli wartość wykładnika, będzie równe `0`, funkcja ma zwrócić 1.

W przeciwnym razie funkcja zwróci `p * rekurencja (p, w – 1)`, czyli wartość podstawy pomnożoną przez wywołanie funkcji z wykładnikiem zmniejszonym o 1.

```
1  if (w == 0) return 1;
2     else return p *
   rekurencja(p, w - 1);
```

9

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PtA31hjm7

Po zdefiniowaniu funkcji przejdziemy do głównej części programu, w której poprosimy użytkownika o wprowadzenie wartości podstawy i wykładnika.

Musimy jednak zadeklarować zmienne, w których przechowamy podane przez użytkownika dane. Będą to dwie zmienne typu `int`.

```
1 int p;
2 int w;
```

10

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PtA31hjm7

Używając predefiniowanych strumieni wejścia i wyjścia, zasygnalizujemy użytkownikowi, że ma wprowadzić wartość podstawy i wykładnika.

Następnie pobierzemy od niego dane do zadeklarowanych wcześniej zmiennych.

```
cout << "Podaj podstawę: ";
cin >> p;
cout << "Podaj wykładnik: ";
cin >> w;
```

11

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PtA31hjm7

Teraz użyjemy instrukcji warunkowej `if…else`, gdzie w teście logicznym sprawdzimy, czy podana podstawa jest mniejsza od n. Jeżeli tak, użyjemy iteracyjnej funkcji potęgowania. Jeżeli nie, zastosujemy funkcję rekurencyjną.

```
if (p < n)
        cout << iteracja(p, w);
else
        cout << rekurencja(p, w);
```

12

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PtA31hjm7

Na koniec dodamy informację na temat wartości zmiennej n. Załóżmy, że wynosi 100. Nic jednak nie stoi na przeszkodzie, aby i jej wartość podał użytkownik.

```
int n = 100;
```

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PtA31hjm7

W ten sposób udało nam się stworzyć program, który w zależności od podanej podstawy oblicza wartość potęgi iteracyjnie lub rekurencyjnie.
Oto pełny kod programu:

```
#include <iostream>
using namespace std;

int iteracja(int a, int b)
{
    int odp = 1;
    while (b > 0)
    {
        odp *= a;
        b--;
    }
    return odp;
}

long long rekurencja(int a, int b)
{
    if (b == 0) return 1;
    else return a * rekurencja(a, b - 1);
}

int main ()
{
    int p;
    int w;
    int n = 100;
    cout << "Podaj podstawę: ";
    cin >> p;
    cout << "Podaj wykładnik: ";
```

```
    cin >> w;
    if (p < n)
        cout << iteracja(p, w);
    else
        cout << rekurencja(p, w);
    return 0;
}
```


Polecenie 2

Dodaj do swojego programu komentarze tak, żeby był zrozumiały dla osoby, która nie potrafi programować.

Sprawdź się

Pokaż ćwiczenia: 

Ćwiczenie 1

Uzupełnij kod programu w taki sposób, aby obliczał x^y, gdzie x i y są liczbami naturalnymi, a `y > 0`. Nie modyfikuj funkcji `main`.

Specyfikacja problemu:

Dane:

- x – liczba naturalna, którą podnosimy do potęgi
- y – liczba naturalna; potęga liczby x; gdzie `y > 0`

Wynik:

Program na wyjściu standardowym zwróci wartość zmiennej x do potęgi y.

Twoje zadania

1. Program podnosi wartość zmiennej x do potęgi y.

```
#include <iostream>

using namespace std;

int rekurencja (int a, int b)
{
    if (/*Tutaj wpisz test logiczny*/) return 1;
    else return a * rekurencja (/*Tutaj wstaw odpowiednie zmienne*/);
}

int iteracja(int a, int b)
{
    int odp = 1;
```

```

```

Ćwiczenie 2

Napisz program obliczający metodą iteracyjną wartość x^y, gdzie x i y są liczbami naturalnymi, a `y > 0`.

Specyfikacja problemu:

Dane:

- x – liczba naturalna, którą podnosimy do potęgi
- y – liczba naturalna, potęga liczby x; gdzie `y > 0`

Wynik:

Program na wyjściu standardowym zwróci wartość zmiennej x do potęgi y metodą iteracyjną.

Twoje zadania

1. Program podnosi wartość zmiennej x do potęgi y za pomocą funkcji napisanej metodą iteracyjną.

```
#include <iostream>

using namespace std;

int iteracja (int a, int b)
{
    //Tutaj wprowadź swój kod
}

int main () {
    int x = 3;
    int y = 3;

    cout << iteracja (x, y)
```

```

```

Ćwiczenie 3

Napisz program obliczający metodą rekurencyjną wartość x^y, gdzie x i y są liczbami naturalnymi, a `y > 0`.

Specyfikacja problemu:

Dane:

- x – liczba naturalna, którą podnosimy do potęgi
- y – liczba naturalna; potęga liczby x; gdzie `y > 0`

Wynik:

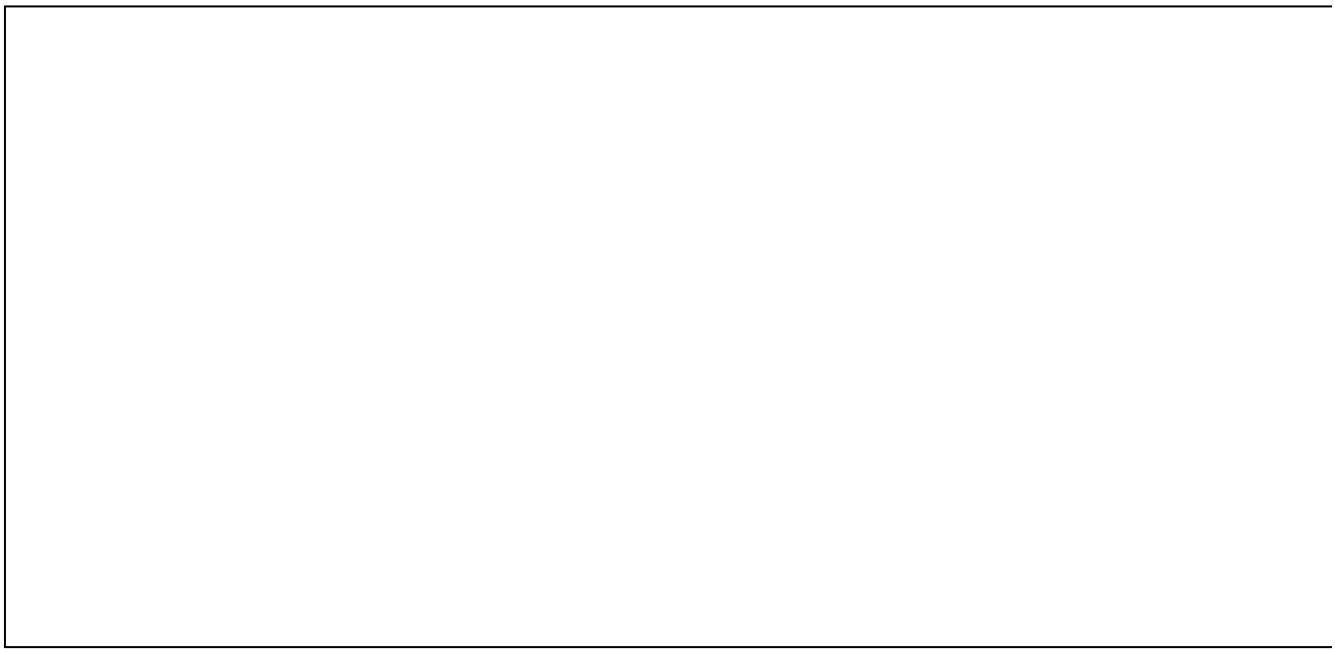
Program na wyjściu standardowym zwróci wartość zmiennej x do potęgi y metodą rekurencyjną.

Twoje zadania

1. Program podnosi wartość zmiennej x do potęgi y za pomocą funkcji napisanej metodą rekurencyjną.

```
#include <iostream>

using namespace std;

int rekurencja (int a, int b)
{
    //Tutaj wprowadź swój kod
}

int main () {
    int x = 2;
    int y = 3;

    cout << rekurencja (x, y)
```

```
1
```



Dla nauczyciela

Autor: Maurycy Gast
Przedmiot: Informatyka

Temat: Analiza podejścia rekurencyjnego i iteracyjnego w języku C++

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):

- Przeanalizujesz rekurencyjny i iteracyjny sposób rozwiązywania problemów w języku C++.
- Porównasz rozwiązania iteracyjne i rekurencyjne tego samego problemu.
- Zaimplementujesz w języku w C++ program obliczający potęgę liczby.

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. **Przygotowanie do zajęć.** Nauczyciel loguje się na platformie i udostępnia e-materiał: „Analiza podejścia rekurencyjnego i iteracyjnego w języku C++". Uczniowie zapoznają się z treściami w sekcji „Przeczytaj".

Faza wstępna:

1. Przedstawienie tematu i celów zajęć.

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".
2. **Praca z multimedium.** Nauczyciel wyświetla zawartość sekcji „Prezentacja multimedialna". Uczniowie w parach rozwiązują Problem 1. Następnie łączą się w grupy i porównują swoje rozwiązania. Chętna lub wybrana osoba przedstawia swoje rozwiązanie. Uczniowie dyskutują na jego temat.
3. **Ćwiczenie umiejętności.** Uczniowie wykonują indywidualnie ćwiczenie nr 1, a następnie porównują swoje odpowiedzi z kolegą lub koleżanką.
4. Liga zadaniowa – uczniowie pracując w parach, wykonują ćwiczenie nr 2 z sekcji „Sprawdź się", a następnie dzielą się swoimi wynikami przez porównywanie napisanego kodu z inną grupą, która również zakończyła zadanie.

Faza podsumowująca:

1. Na koniec zajęć nauczyciel prosi uczniów o rozwinięcie zdania: „Na dzisiejszych zajęciach nauczyłam/łem się jak…".

Praca domowa:

1. Kiedy wybrać metodę rekurencyjną, a kiedy iteracyjną? Spróbuj odpowiedzieć na pytanie.

Materiały pomocnicze:

- Oficjalna dokumentacja techniczna dla języka C++.
- Oficjalna dokumentacja techniczna dla kompilatora GCC/G++ 4.5 (lub nowszej wersji).
- Oficjalna dokumentacja techniczna dla oprogramowania Code::Blocks 16.01 (lub nowszej wersji), Orwell Dev-C++ 5.11 (lub nowszej wersji) lub Microsoft Visual Studio.

Wskazówki metodyczne:

- Uczniowie mogą wykorzystać treści w sekcjach: „Przeczytaj”, „Prezentacja multimedialna”, „Sprawdź się” jako materiał do lekcji powtórkowej.