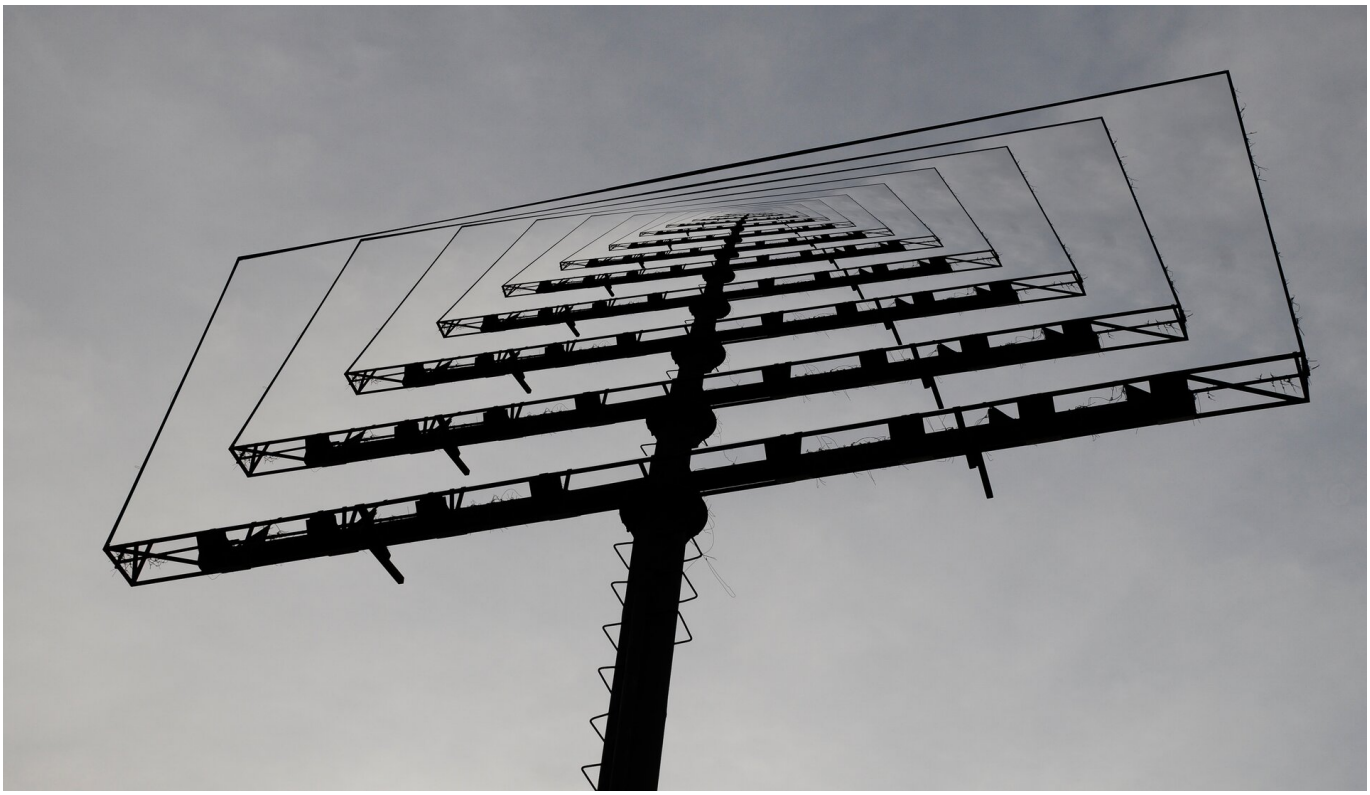


Rekurencja w języku C++

Rekurencja w języku C++

Źródło: schrupp, domena publiczna.

Rekurencja jest metodą rozwiązywania problemów, która korzysta z rozwiązań innych problemów. Jej wyjątkowość polega na tym, że za „innymi problemami” kryje się problem właściwy, ale rozwiązywany dla innych rozmiarów tych samych danych. Przykładem może być obliczanie silni liczby. Żeby policzyć 56!, najpierw możemy wyliczyć 55!. Z kolei w celu ustalenia, ile wynosi 55!, można najpierw obliczyć 54! itd.

W e-materiale Rekurencja przedstawiliśmy podstawowe informacje dotyczące omawianego zagadnienia. Kolejnym krokiem będzie implementacja w wybranym języku programowania. Ten e-materiał poświęcony jest implementacji w języku C++.

Więcej przykładów, ćwiczeń i rozwiązań znajdziesz w:

- Rekurencja – ćwiczenia,
- Rekurencja w zadaniach.

Ciekawi cię, jak wyglądają implementacje w innych językach programowania? Możesz się z nimi zapoznać w dwóch pozostałych e-materiałach z tej serii:

- Rekurencja w języku Java,
- Rekurencja w języku Python.

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.

Twoje cele

- Przeanalizujesz rekurencyjne podejście do algorytmu Euklidesa i poznasz zalety jego stosowania.
- Wyjaśnisz, czym rekurencja różni się od iteracji.
- Rozwiążesz kilka zadań z wykorzystaniem rekurencji.

Przeczytaj

W e-materiale Algorytm Euklidesa poznaliśmy algorytm służący do znajdowania największego wspólnego dzielnika dwóch liczb. Znamy także iteracyjną realizację tego algorytmu (omówiona ona została w e-materiale Algorytm Euklidesa w języku C++) – tym razem zbadamy podejście rekurencyjne.

Algorytm Euklidesa z odejmowaniem – implementacja algorytmu w języku C++

Zaimplementujmy z użyciem rekurencji algorytm Euklidesa w języku C++.

W funkcji `NWD()`, która przyjmuje dwie zmienne typu `int`, użylibyśmy instrukcji warunkowej `if`, w której warunek będzie taki sam, jak w przypadku pętli `while` w wykonaniu iteracyjnym, czyli `a != b`.

Jeżeli warunek zostanie spełniony, funkcja będzie wywoływana rekurencyjnie do momentu, gdy obie wartości nie będą równe. W przypadku gdy wartość `a` będzie większa od wartości `b`, funkcja zwróci `NWD(a - b, b)`, a w odwrotnym przypadku `NWD(a, b - a)`.

Jeżeli warunek nie zostanie spełniony, funkcja zwróci `a`. Oczywiście mogłaby też zwrócić `b` – jeżeli warunek nie zostanie spełniony, będzie to oznaczać, że wartość obu zmiennych jest identyczna.

Nasza funkcja wygląda następująco:

```
int NWD(int a, int b) {
    if (a != b) {
        if (a > b) return NWD(a - b, b);
        return NWD(a, b - a);
    }

    return a;
}
```

Funkcja kończy swoje działanie po zwróceniu wartości, więc nie musimy stosować instrukcji `else`.

Algorytm Euklidesa z dzieleniem – implementacja algorytmu w języku C++

Zadeklarujemy funkcję `NWD()` zwracającą typ `int`. Przyjmie ona parametry `a` i `b` – obydwa typu `int`. W ciele funkcji umieścimy instrukcję warunkową `if`, której testem logicznym będzie `b != 0`. Jeżeli zależność zachodzi, funkcja zwróci samą siebie z argumentami `b` i `a % b`. W przeciwnym razie funkcja zwróci zmienną `a`. Kod wygląda następująco:

```
1 int NWD(int a, int b) {
2     if (b != 0)
3         return NWD(b, a % b);
4 
5     return a;
6 }
```

Istotne jest, żeby funkcja zwróciła przypadek podstawowy w momencie, gdy `b` przyjmie wartość `0`. W innym przypadku w następnym wywołaniu funkcji pojawiłby się argument, który doprowadziłby do wykonania operacji niedozwolonej, czyli szukania reszty z dzielenia przez `0`.

Słownik

> **NWD**
>
> skrót pojęcia największy wspólny dzielnik dwóch liczb całkowitych – największa liczba naturalna, która dzieli dwie rozpatrywane liczby bez reszty
>
> **przypadek podstawowy**
>
> przypadek, w którym funkcja rekurencyjna zwraca konkretną wartość, a nie wywołanie samej siebie

Prezentacja multimedialna

Polecenie 1

Przeanalizuj prezentację, a następnie prześledź etapy wykonywania obu funkcji rekurencyjnych dla podanych przez siebie liczb.

1

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P3LDgPUST

Załóżmy, że poproszono nas o napisanie programu, który obliczy NWD dwóch podanych przez użytkownika liczb. Wprowadzona zostanie także trzecia wartość, która odpowiadać będzie za typ algorytmu Euklidesa, który ma zostać użyty do obliczenia NWD.

Jest to odpowiednio:

1. algorytm Euklidesa z dzieleniem,
2. algorytm Euklidesa z odejmowaniem.

Program ma stosować algorytm w sposób rekurencyjny.

2

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P3LDgPUST

Zaczniemy od stworzenia funkcji rekurencyjnej dla algorytmu Euklidesa z dzieleniem. Będzie ona zwracać typ `int`, a jej argumentami będą dwie zmienne typu `int`, `a` i `b`.

```
int NWD_mod(int a, int b)
{

}
```

3

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P3LDgPUST

Następnie umieścimy w ciele funkcji instrukcję warunkową `if`. W jej teście logicznym sprawdzimy, czy zmienna b jest różna od zera.

Jeżeli warunek zostanie spełniony, funkcja zwróci samą siebie, z argumentami b i resztą z dzielenia a przez b.

W przeciwnym razie – czyli w przypadku podstawowym – funkcja zwróci wartość zmiennej a.

```
if (b != 0)
    return NWD_mod(b, a % b);

return a;
```

4

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P3LDgPUST

Teraz zajmiemy się funkcją rekurencyjną odpowiadającą za algorytm Euklidesa z odejmowaniem. Podobnie jak w przypadku funkcji `NWD_mod()`, funkcja ta będzie zwracać typ `int` oraz przyjmie dwie zmienne typu `int`, które nazwiemy a i b.

```
int NWD_mod2(int a, int b)
{

}
```

5

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P3LDgPUST

W ciele funkcji zapiszemy instrukcję warunkową `if`, w której sprawdzimy, czy zmienne a i b mają różne wartości.

Jeżeli warunek zostanie spełniony, to w zależności od tego, która ze zmiennych przechowuje wyższą wartość, funkcja zwróci `NWD_mod2(a - b, b)` lub `NWD_mod2(a, b - a)`.

W przypadku podstawowym funkcja zwróci wartość zmiennej a.

```
if (a != b) {
    if (a > b) return NWD_mod2(a - b, b);
    return NWD_mod2(a, b - a);
}

return a;
```

6

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P3LDgPUST

Teraz w głównej części programu zadeklarujemy zmienne, w których przechowywać będziemy

podane przez użytkownika liczby oraz wybrany typ algorytmu.

```
int a;
int b;
int typ;
```

W celu pobrania danych wykorzystywać będziemy musieli bibliotekę `iostream`. Jeżeli jeszcze jej nie dołączyliśmy, zrobimy to, używając na początku programu następującego kodu:

```
#include <iostream>
```

7

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P3LDgPUST

Następnie poprosimy użytkownika o wprowadzenie trzech liczb. Znajdziemy NWD dwóch pierwszych liczb, natomiast trzecia z nich odpowiadać będzie za typ algorytmu Euklidesa.

Poprosimy użytkownika o wprowadzenie danych w następujący sposób:

```
cout << "Proszę wprowadzić dwie liczby oraz typ: ";
cin >> a >> b >> typ;
```

8

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P3LDgPUST

Gdy już udało nam się uzyskać od użytkownika wszystkie potrzebne informacje, stworzymy

instrukcję warunkową `if`, sprawdzimy, czy `typ` podany przez użytkownika mieści się w zbiorze {1, 2}.

W przypadku, w którym warunek ten nie będzie spełniony, zwrócimy tekst `Niepoprawny typ.`

```
1 if (typ == 1 || typ == 2)
  {
2
3 } else cout <<
  "Niepoprawny typ.";
```

9

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P3LDgPUST

Gdy warunek `typ == 1 || typ == 2` zostanie spełniony, za pomocą kolejnych instrukcji warunkowych `if` sprawdzimy, którego algorytmu mamy użyć, aby podać odpowiedź.

Dla `typ == 1` użyjemy algorytmu Euklidesa z dzieleniem, ale dla `typ == 2` algorytmu z odejmowaniem.

```
1 if (typ == 1)
2     cout << NWD_mod(a, b);
3 if (typ == 2)
4     cout << NWD_mod2(a,
  b);
```

10

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P3LDgPUST

Tym samym skończyliśmy pisać program. Kod powinien wyglądać następująco:

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

int NWD_mod(int a, int b)
{
    if (b != 0)
        return NWD_mod(b, a % b);

    return a;
}

int NWD_mod2(int a, int b)
{
    if (a != b) {
        if (a > b) return NWD_mod2(a - b, b);
        return NWD_mod2(a, b - a);
    }

    return a;
}

int main() {
    int a;
    int b;
    int typ;

    cout << "Prosze wprowadzic dwie liczby oraz typ: ";
    cin >> a >> b >> typ;

    if (typ == 1 || typ == 2) {
        if (typ == 1)
            cout << NWD_mod(a, b);
        if (typ == 2)
            cout << NWD_mod2(a, b);
```

```
33      } else cout <<
"Niepoprawny typ.";
34  }
```

Problem 1

Napisz program, który policzy n-ty element ciągu.

Przetestuj jego działanie dla ciągu o wzorze:

$$a_n = a_{n-1} + r$$

oraz $a_0 = 1$, $r = 5$, $n = 3$.

Specyfikacja problemu:

Dane:

- n – liczba naturalna
- r – liczba naturalna
- a_0 – liczba naturalna

Wynik:

Program wyświetla n-ty wyraz ciągu wyrażonego podanym wzorem.

```
1
```

```
1
```

Problem 2

Napisz program, który policzy największy wspólny dzielnik (NWD) wskazanych liczb.

Przetestuj jego działanie dla n = 24 oraz k = 28.

Rekurencyjny wzór na wyznaczanie NWD:

$$nwd\Big(k,\ n\Big) \ = \ \begin{cases} n & dla\ k=0 \\ nwd(n\ mod\ k,\ k) & dla\ k>0 \end{cases}$$

Specyfikacja problemu:

Dane:

- n – liczba naturalna
- k – liczba naturalna

Wynik:

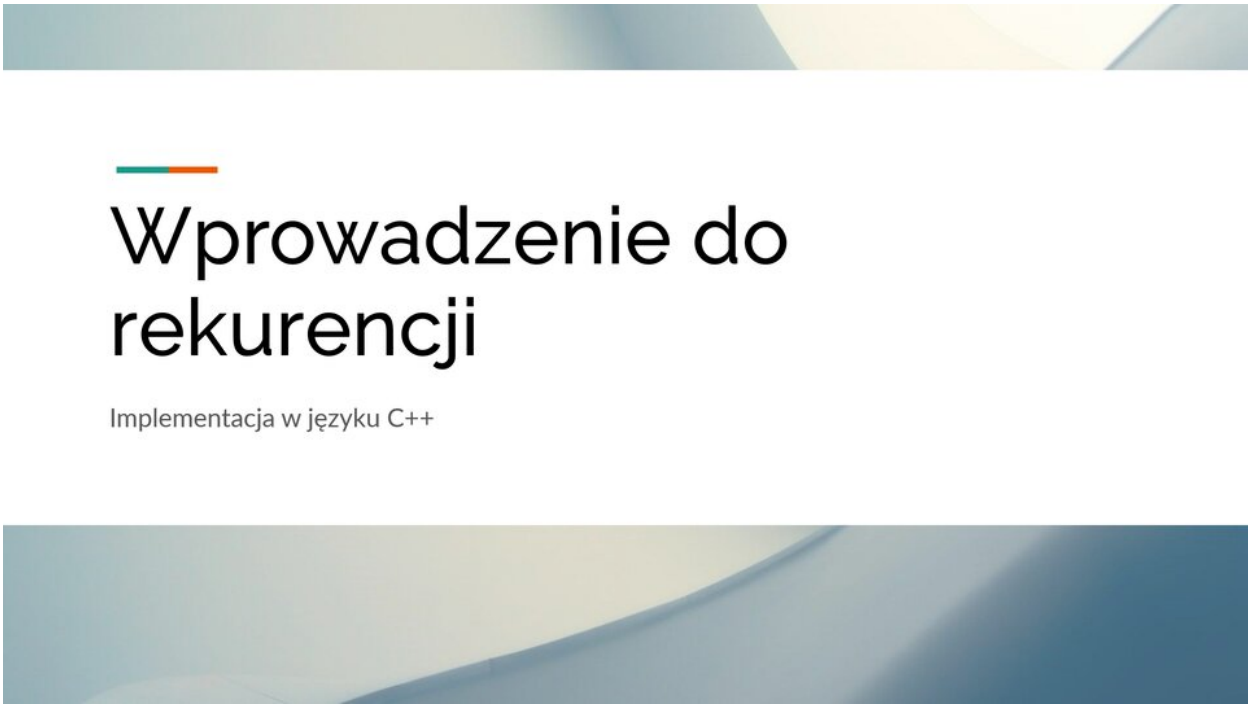
Program wyświetla n-ty wyraz ciągu wyrażonego podanym wzorem.

```
1
```

```
1
```

Polecenie 2

Porównaj swoje rozwiązanie z filmem.



Film dostępny pod adresem /preview/resource/Re3VZS47Sn4yC

Film nawiązujący do treści materiału: Wprowadzenie do rekurencji.

Kod programu zaprezentowanego w filmie:

Plik o rozmiarze 317.00 B w języku polskim

Sprawdź się

Pokaż ćwiczenia: 

Ćwiczenie 1

Napisz program, który obliczy sumę n kolejnych liczb naturalnych z użyciem rekurencji.

Przetestuj działanie programu dla `n = 10`.

Specyfikacja:

Dane:

- n – liczba naturalna

Wynik:

Program wyświetla sumę kolejnych liczb naturalnych z przedziału `<1, n>`.

Twoje zadania

1. Program wypisuje na standardowe wyjście wartość zwracaną przez funkcję `suma` dla podanych danych wejściowych.

Ćwiczenie 2

Napisz program, który wygeneruje n-te słowa Fibonacciego oddzielone znakiem nowej linii. Kolejne wartości n podane są w tablicy.

Słowa Fibonacciego tworzone są według następującej zasady:

$$F_n := \begin{cases} b & dla\ n = 1 \\ a & dla\ n = 2 \\ F_{n-1} \times F_{n-2} & dla\ n > 2 \end{cases}$$

Gdzie $x \times y$ oznacza konkatenację słowa x ze słowem y – łączymy słowo x oraz y, przy czym słowo x występuje na pierwszej pozycji.

Przykład:

$F_1 = b$

$F_2 = a$

$F_3 = ab$

$F_4 = aba$

$F_5 = abaab$

Przetestuj działanie programu dla tablicy n o następującej zawartości: {3, 1, 9, 10, 6}.

Specyfikacja:

Dane:

- n – tablica liczb naturalnych dodatnich

Wynik:

Program wyświetla w kolejnych wierszach słowa Fibonacciego o właściwych numerach.

Twoje zadania

1. Program wypisuje n-te słowa Fibonacciego.

Ćwiczenie 3

Napisz program, który – używając funkcji rekurencyjnej – skróci ułamek.
Przetestuj swój program dla ułamka o liczniku `8`, przechowywanym w zmiennej `x`
i mianowniku `40` zawartym w zmiennej `y`.

Specyfikacja:

Dane:

- `x` – licznik, liczba całkowita
- `y` – mianownik, liczba całkowita

Wynik:

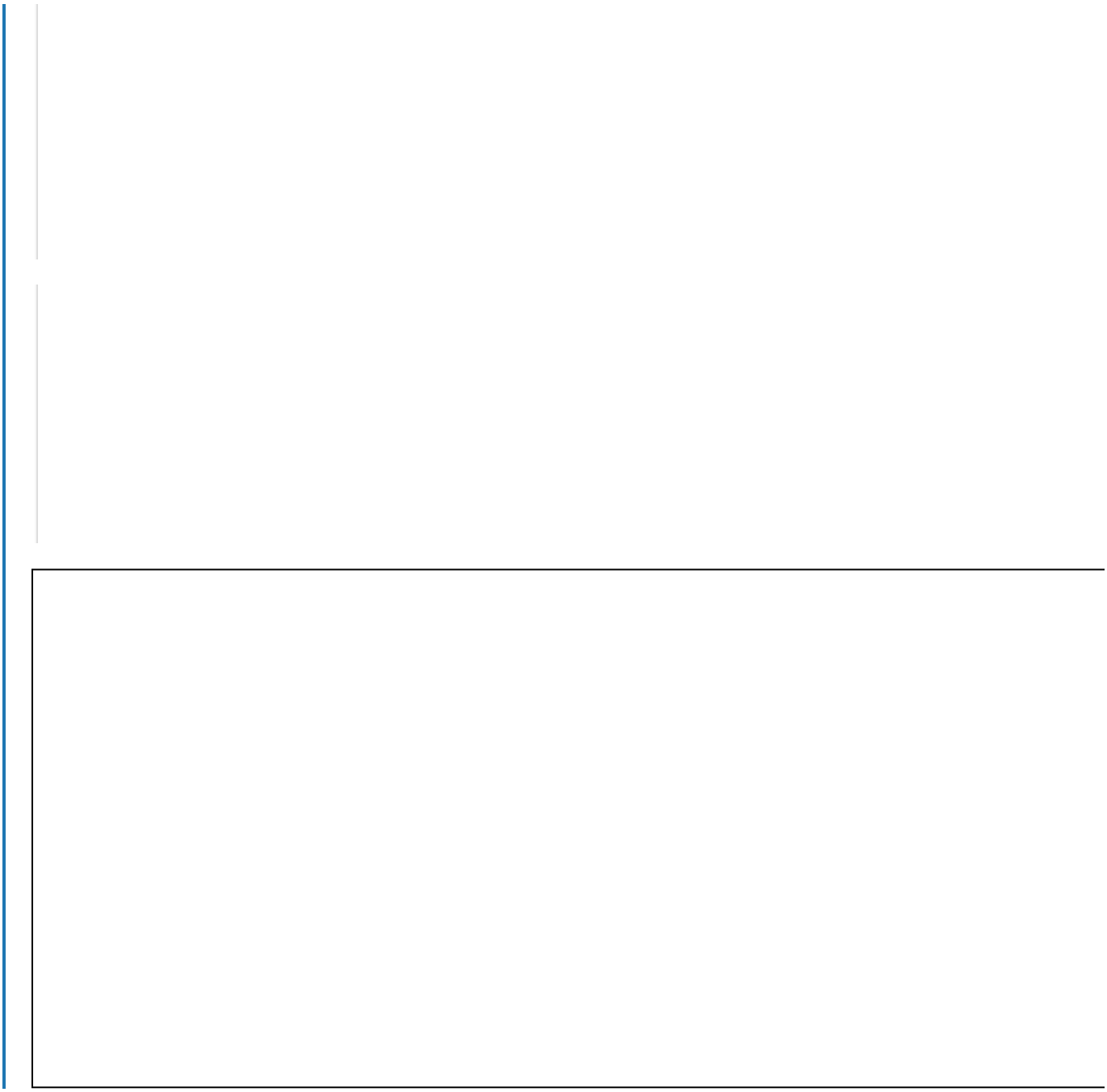
- `licznik` – liczba całkowita
- `mianownik` – liczba całkowita

Przykładowe wyjście:

```
1/5
```

Twoje zadania

1. Program skraca ułamek, w którym zmienna `x` stanowi licznik, a `y` mianownik, oraz drukuje wynik na standardowe wyjście w postaci: wartość licznika po skróceniu, znak `/`, wartość mianownika po skróceniu.



Dla nauczyciela

Autor: Maurycy Gast
Przedmiot: Informatyka

Temat: Rekurencja w języku C++

Grupa docelowa:

Szkoła ponadpodstawowa, liceum ogólnokształcące, technikum, zakres 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 + II. Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:

1) zapisuje za pomocą listy kroków, schematu blokowego lub pseudokodu, i implementuje w wybranym języku programowania, algorytmy poznane na wcześniejszych etapach oraz algorytmy:

a) algorytm Euklidesa w wersji iteracyjnej i rekurencyjnej wraz z zastosowaniami,

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 rekurencyjne podejście do algorytmu Euklidesa i poznasz zalety jego stosowania.
- Wyjaśnisz, czym rekurencja różni się od iteracji.
- Rozwiążesz kilka zadań z wykorzystaniem rekurencji.

Strategie nauczania:

- konstruktywizm;
- konektywizm.

Metody i techniki nauczania:

- dyskusja;
- rozmowa nauczająca z wykorzystaniem multimedium i ćwiczeń interaktywnych;
- metody aktywizujące.

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;
- oprogramowanie dla języka C++, w tym kompilator GCC/G++ 4.5 (lub nowszej wersji) i Code::Blocks 16.01 (lub nowszej wersji), Orwell Dev-C++ 5.11 (lub nowszej wersji) lub Microsoft Visual Studio.

Przebieg lekcji

Przed lekcją:

1. **Przygotowanie do zajęć.** Nauczyciel loguje się na platformie i udostępnia e-materiał: „Rekurencja w języku C++”. Nauczyciel prosi uczniów o zapoznanie się z treściami w sekcji „Przeczytaj”.

Faza wstępna:

1. Przedstawienie tematu i celów zajęć.
2. Rozpoznawanie wiedzy uczniów.

Faza realizacyjna:

1. **Praca z multimedium.** Uczniowie w zespołach dwuosobowych zapoznają się z treścią polecenia nr 1: „Przeanalizuj prezentację, a następnie prześledź etapy wykonywania obu funkcji rekurencyjnych dla podanych przez siebie liczb." z sekcji „Prezentacja multimedialna" i wspólnie analizują kolejne kroki rozwiązania postawionego problemu.
2. **Ćwiczenie umiejętności.** Uczniowie, pracując w parach, wykonują ćwiczenie nr 1 z sekcji „Sprawdź się". Nauczyciel sprawdza poprawność pisanych kodów, porównuje je i omawia wraz z uczniami. Wskazuje najbardziej efektywne rozwiązanie.
3. Uczniowie samodzielnie wykonują ćwiczenie nr 2 w sekcji „Sprawdź się". Chętne lub wybrane osoby przedstawiają rozwiązania i je omawiają.
4. Uczniowie wykonują ćwiczenie nr 3 z sekcji „Sprawdź się".

Faza podsumowująca:

1. Nauczyciel ponownie wyświetla na tablicy temat i cele lekcji zawarte w sekcji „Wprowadzenie". W kontekście ich realizacji następuje omówienie ewentualnych problemów z rozwiązaniem ćwiczeń z sekcji „Sprawdź się".
2. Na koniec zajęć z programowania w C++ nauczyciel prosi uczniów o rozwinięcie zdania: „Na dzisiejszych zajęciach nauczyłam/łem się jak…".

Praca domowa:

1. Uczniowie wykonują polecenie nr 2 z sekcji „Prezentacja multimedialna".

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.
- 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.