Ciąg Fibonacciego w języku C++

Źródło: Jason Leung, domena publiczna.

Ciąg Fibonacciego to ciąg liczb naturalnych opisany w sposób rekurencyjny: pierwszy element jest równy 0, drugi równy jest 1, a każdy następny jest sumą dwóch poprzednich. Ciąg ten po raz pierwszy został omówiony w 1202 roku przez Leonarda z Pizy, zwanego również Fibonaccim. Więcej informacji znajdziesz w e-materiale Ciąg Fibonacciego.

Z tego e-materiału dowiesz się, jak w języku C++ zaimplementować algorytm służący do obliczania wyrazów ciągu Fibonacciego.

Implementację ciągu Fibonacciego w pozostałych językach programowania przedstawiamy w e-materiałach:

- Ciąg Fibonacciego w języku Java,
- Ciąg Fibonacciego w języku Python.

Więcej zadań? Zajrzyj do e-materiału Ciąg Fibonacciego – zadania maturalne.

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 rekurencyjny algorytm generujący kolejne elementy ciągu Fibonacciego zapisany za pomocą języka C++.
> - Napiszesz programy wyznaczające elementy ciągu Fibonacciego w sposób rekurencyjny oraz iteracyjny.
> - Rozwiążesz kilka zadań związanych z tematem e-materiału.

Przeczytaj

Idea rozwiązania rekurencyjnego

Mechanizm rekurencji polega na wykorzystaniu funkcji, która wywołuje samą siebie aż do momentu, gdy przeznaczone jej do rozwiązania zadanie zostanie wykonane. Funkcja taka może być typu `void`, ale może również zwracać obliczoną wartość.

Zobaczmy, jak wygląda funkcja rekurencyjna, której zadaniem jest wypisanie wszystkich liczb naturalnych dodatnich mniejszych lub równych od podanego parametru.

Specyfikacja problemu:

Dane:

- n – liczba naturalna dodatnia

Wynik:

Program wypisuje malejący ciąg liczb naturalnych dodatnich mniejszych lub równych n.

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

void wypiszNaturalne(int n) {
    if (n < 1) {
        return;
    }

    cout << n << endl;
    wypiszNaturalne(n - 1);
```

```
}
```

Ponieważ zadaniem funkcji jest wypisanie liczb na standardowe wyjście, nie ma potrzeby, aby zwracała ona jakąkolwiek wartość. Zdefiniowaliśmy zatem funkcję typu `void`. Parametr funkcji `n` przechowuje wartość największej liczby naturalnej, którą program musi wypisać.

Jako pierwsza w funkcji pojawia się instrukcja warunkowa, która zatrzyma działanie funkcji, bez wywołania kolejnej instancji, jeżeli parametr `n` osiągnie wartość mniejszą niż jeden. Warunków powodujących zakończenie działania funkcji rekurencyjnej może być więcej niż jeden.

Zaraz pod instrukcją warunkową wypisujemy wartość parametru `n`, a następnie wywołujemy funkcję rekurencyjną z argumentem zmniejszonym o 1.

Prześledźmy działanie funkcji, gdy wywołamy ją z argumentem `n` równym 3.

```
if (3 < 1) {
    return;
}

cout << 3 << endl;
wypiszNaturalne(3 - 1);
```

Czy 3 jest mniejsze niż 1? Nie. Dlatego pomijamy instrukcje warunkową i wypisujemy wartość 3, a następnie wywołujemy tę samą funkcję z argumentem zmniejszonym o 1, czyli `n = 2`.

```
if (2 < 1) {
    return;
}

cout << 2 << endl;
wypiszNaturalne(2 - 1);
```

Liczba 2 nie jest mniejsza od 1, więc polecenie zapisane w instrukcji warunkowej nadal nie zostaje wykonane. Wpisujemy więc liczbę 2 i wywołujemy funkcję z argumentem równym 1.

```
if (1 < 1) {
    return;
}

cout << 1 << endl;
wypiszNaturalne(0);
```

Czy 1 jest mniejsze niż 1? Ponownie nie, więc wypisujemy wartość 1 i wywołujemy funkcję po raz kolejny, tym razem z argumentem równym 0.

```
if (0 < 1) {
    return;
}
```

Ponieważ liczba 0 jest mniejsza niż 1, wyrażenie logiczne w instrukcji warunkowej przybierze wartość `true`. W rezultacie zostanie wykonane polecenie `return`, które zakończy działanie funkcji. Wynikiem działania programu będą liczby 3, 2, 1.

Jak widać, każde wywołanie funkcji zatrzymuje pracę funkcji bieżącej. Czeka ona, aż funkcja przez nią wywołana zakończy działanie (i zwróci wartość, jeśli miałaby ją zwracać). Przy wywoływaniu kolejnych funkcji rekurencyjnych dzieje się tak samo.

Jeżeli wywołana funkcja rekurencyjna spełni warunek zakończenia pracy i – gdy tego oczekiwano – zwróci wartość zamiast wywoływać siebie samą, funkcja, która ją wywołała, może podjąć działanie. Zwraca ona wartość poprzedniczce, która również podejmuje działanie, zwraca wartość funkcji ją wywołującej i tak dalej. Proces ten trwa do momentu, gdy pracę wznowi pierwsza, wywołana przez program funkcja. Zwraca ona ostateczny wynik realizacji algorytmu rekurencyjnego.

Aby opisany mechanizm działał poprawnie, potrzebny jest co najmniej jeden warunek, po spełnieniu którego funkcja nie zostanie wywołana po raz kolejny, lecz zakończy się (oraz ewentualnie zwróci wartość). Jeżeli taki warunek się nie pojawi, funkcja będzie wywoływać się w nieskończoność, co spowoduje zawieszenie programu lub błąd przepełnienia stosu.

Przy każdym kolejnym wywołaniu funkcji jej parametr lub jeden z parametrów muszą zmienić wartość. Nie może też dojść do sytuacji, w której wartość parametru lub parametrów powtórzą się między wywołaniami. Jeżeli tak się stanie, program może nigdy nie zakończyć działania.

Ciąg Fibonacciego

Ciąg Fibonacciego to ciąg liczb naturalnych, w którym pierwsze dwa elementy (o indeksie zero i jeden) to 0 oraz 1, a każdy kolejny element jest sumą dwóch poprzednich. Elementem ciągu o indeksie dwa jest zatem 0 + 1 = 1, o indeksie trzy zaś 1 + 1 = 2 i tak dalej.

Definicja ciągu Fibonacciego ma następującą postać:

$$F_n = \begin{cases} 0 & \text{dla } n = 0 \\ 1 & \text{dla } n = 1 \\ F_{n-1} + F_{n-2} & \text{dla } n > 1 \end{cases}$$

Rekurencja

Problem 1
Napisz w języku C++ program, który w sposób rekurencyjny znajdzie i wypisze element ciągu Fibonacciego o indeksie n. Przetestuj swój program dla elementu ciągu o indeksie 3.

Specyfikacja problemu:

Dane:

- n – indeks elementu ciągu Fibonacciego, liczba naturalna

Wynik:

Program wypisuje element ciągu Fibonacciego o indeksie n.

```
1
```

```
1
```

Polecenie 1

Porównaj swoje rozwiązanie z prezentacją.

1

Napiszemy program w języku C++, który w sposób rekurencyjny znajdzie i wypisze element ciągu Fibonacciego o indeksie n. Implementację algorytmu zaczniemy od stworzenia funkcji rekurencyjnej.

2

Tym razem funkcja rekurencyjna będzie zwracać wartość. Musi mieć zatem możliwość zwrócenia obliczonych wartości do funkcji, która ją wywołała. Z tego powodu typem funkcji nie będzie `void`. Liczby Fibonacciego są liczbami naturalnymi. Typem funkcji mógłby więc być `int`, `long int` lub `long long`. Zdecydujemy się na `long long`, ponieważ dzięki temu będziemy mogli obliczyć dalsze elementy ciągu Fibonacciego, zmniejszając ryzyko, że będą wykraczać poza zakres wartości danego typu.

3

Kolejnym krokiem jest zapisanie warunków, po których spełnieniu zostanie zatrzymane wywoływanie kolejnych funkcji i zwrócona będzie wartość. Pierwszym elementem ciągu Fibonacciego, gdy `n = 0`, jest liczba 0. Natomiast drugim elementem, gdy `n = 1`, jest liczba 1. Wstawmy więc odpowiednie instrukcje warunkowe, które rozpatrzą te dwie sytuacje.

4

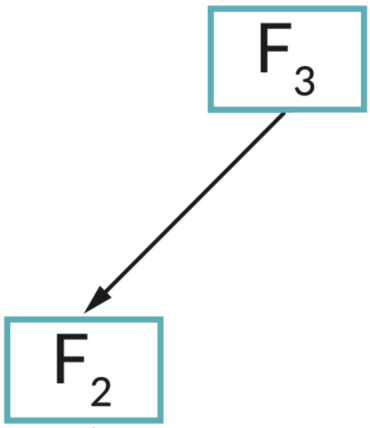
5

Jeżeli jeden z warunków zostanie spełniony, kolejna instancja funkcji nie zostanie wywołana i wartość zostanie zwrócona. Co stanie się w przypadku, gdy wartość parametru `n` jest większa niż 1? Wtedy każdy kolejny element jest sumą dwóch poprzednich.

6

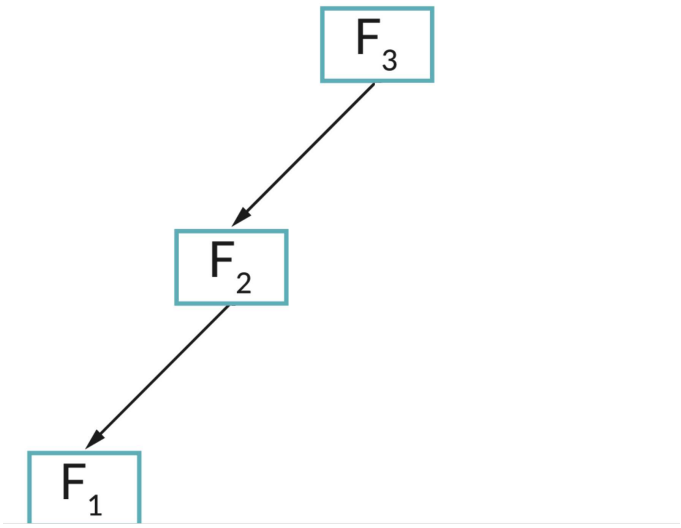
Spróbujmy sprawdzić działanie programu. Wywołajmy funkcję z argumentem `n` równym 3. Szukamy zatem elementu ciągu Fibonacciego o indeksie 3.

7



Najpierw jest wywoływana funkcja `fibonacci(3 - 1)` i zatrzymywane jest działanie bieżącej funkcji. Realizowana jest funkcja `fibonacci(2)` i dopiero gdy zwróci wartość, przechodzimy do kolejnego wywołania – `fibonacci(1)`. Z tego powodu w drzewie wywołań rekurencyjnych widzisz jedynie lewe poddrzewo (funkcja `fibonacci(2)` musi się zakończyć).

8



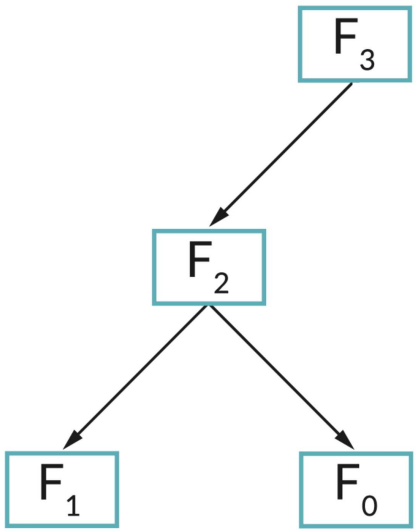
W tym przypadku również możemy zignorować instrukcje warunkowe, ponieważ wyrażenia `2 == 0` oraz `2 == 1` zwrócą wartość `false`.

Funkcja `fibonacci(2)` wywoła zatem samą siebie najpierw z argumentem równym 1.

9

Jeden nie jest równe 0, ale jest równe 1. Nasza funkcja `fibonacci(1)` kończy zatem działanie i zwraca wartość 1.

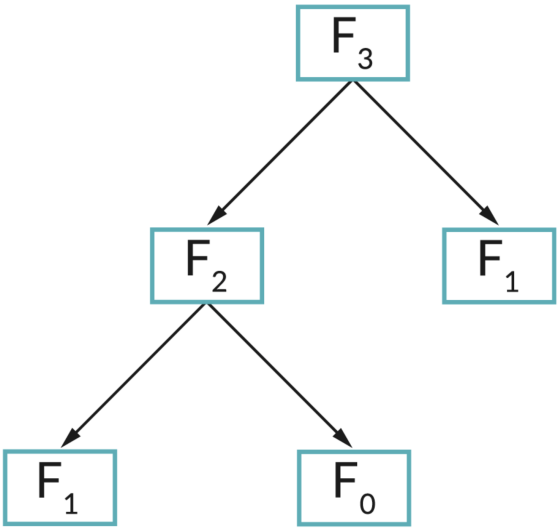
10



Teraz następuje powrót do funkcji `fibonacci(2)` i z niej wywoływana jest druga funkcja `fibonacci(0)` (prawe poddrzewo), która oczywiście zwróci wartość 0.

`1 + 0 = 1` więc funkcja `fibonacci(2)` zwraca wartość 1:

11



Funkcja `fibonacci(2)` zwróciła wartość, więc możemy przejść do kolejnej funkcji, `fibonacci(1)`. Wiemy z poprzednich wywołań, że zwraca ona wartość 1.

12

Wstawmy otrzymane wartości do funkcji pierwotnie wywołanej.

Po wstawieniu wartości zwróconych przez wywołane funkcje, funkcja `fibonacci(3)` zwraca wartość 2. Tak więc elementem ciągu Fibonacciego o indeksie równym 3 jest 2.

Oto cały program:

13

Iteracja

Napiszemy program w języku C++, który w sposób iteracyjny znajdzie i wypisze element ciągu Fibonacciego o indeksie `n`. Implementację algorytmu zaczniemy od stworzenia funkcji iteracyjnej.

```
long long iter_fibonacci(int n) {

}
```

Ta funkcja również będzie zwracać wartość. Jej typ to `long long`, ponieważ wartości kolejnych wyrazów ciągu Fibonacciego rosną wykładniczo.

Zapisujemy warunki, po których spełnieniu zostanie zatrzymane działanie funkcji i zwróci ona wartość. Wiemy, że pierwszy wyraz ciągu Fibonacciego – zatem wyraz o indeksie 0 – ma wartość 0. Natomiast drugi wyraz – czyli wyraz o indeksie 1 – wynosi 1. Zapiszemy odpowiednie instrukcje warunkowe.

```
long long iter_fibonacci(int n) {
    if (n == 0) {
        return 0;
    } else if (n == 1) {
        return 1;
    }
}
```

Musimy brać również pod uwagę przypadek, gdy wartość parametru `n` jest większa niż 1. W takiej sytuacji każdy kolejny element jest sumą dwóch poprzednich. Dla dalszego działania funkcji potrzebujemy zmiennych pomocniczych. Inicjalizujemy trzy zmienne pomocnicze: `a`, `b` (oznaczające wartości dwóch początkowych wyrazów ciągu) oraz `i` (indeks kolejnego elementu ciągu Fibonacciego). Przypisujemy im odpowiednio wartości 0, 1 i 2.

```
long long iter_fibonacci(int n) {
    if (n == 0) {
```

```
        return 0;
    } else if (n == 1) {
        return 1;
    }

    long long a = 0;
    long long b = 1;
    int i = 2;
}
```

Zapisujemy pętlę while, która będzie wykonywać się tak długo, dopóki wartość zmiennej `i` jest mniejsza od wartości zmiennej `n` lub jej równa.

```
long long iter_fibonacci(int n) {
    if (n == 0) {
        return 0;
    } else if (n == 1) {
        return 1;
    }

    long long a = 0;
    long long b = 1;
    int i = 2;

    while (i <= n) {

    }
}
```

W każdym obiegu pętli `while` najpierw obliczamy i-ty wyraz ciągu, czyli sumę dwóch poprzednich wyrazów ciągu. Wynik przypisujemy do zmiennej `suma`. Następnie aktualizujemy wartości zmiennych `a` i `b`. Zmiennej `a` przypisujemy wartość zmiennej `b`, a zmiennej `b` wartość zmiennej `suma`. Na końcu zwiększamy wartość zmiennej `i` o 1.

```
```

```
long long iter_fibonacci(int n) {
    if (n == 0) {
        return 0;
    } else if (n == 1) {
        return 1;
    }

    long long a = 0;
    long long b = 1;
    int i = 2;

    while (i <= n) {
        long long suma = a + b;
        a = b;
        b = suma;
        i = i + 1;
    }

    return b;
}
```

Po zakończeniu pętli wartość zmiennej bjest zwracana jako wynik działania funkcji.

Oto cały kod programu:

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

long long iter_fibonacci(int n) {

    if (n == 0) {
        return 0;
    } else if (n == 1) {
        return 1;
    }
```

```
11
12      long long a = 0;
13      long long b = 1;
14      int i = 2;
15
16      while (i <= n) {
17          long long suma = a + b;
18          a = b;
19          b = suma;
20          i = i + 1;
21      }
22
23      return b;
24  }
25
26  int main() {
27      cout << iter_fibonacci(3) << endl;
28      return 0;
29  }
```

Słownik

> **argument funkcji**
>
> element składni w określonym języku programowania, który w wyniku wywołania podprogramu zostaje utożsamiony (skojarzony) z określonym parametrem podprogramu

> **ciąg Fibonacciego**
>
> ciąg liczb naturalnych, określony rekurencyjnie, w którym pierwsze dwa elementy to 0 oraz 1, a każdy kolejny element jest sumą dwóch poprzednich

> **instrukcja warunkowa**
>
> element języka programowania, który pozwala na wykonanie odpowiedniej instrukcji, w zależności od tego, czy wyrażenie logiczne zapisane jako warunek

okaże się prawdziwe, czy fałszywe

nieskończona pętla

pętla, która nigdy nie spełnia warunku zakończenia i nie zostanie zakończona

parametr funkcji

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

rekurencja

wywoływanie funkcji przez siebie samą, aż do momentu zrealizowania wyznaczonego celu

Symulacja interaktywna

Polecenie 1

Spirala Fibonacciego zbudowana jest z ćwiartek okręgów, których promienie są kolejnymi liczbami ciągu Fibonacciego. Przeanalizuj symulację interaktywną ukazującą proces jej powstawania.

Zasób interaktywny dostępny pod adresem https://zpe.gov.pl/a/D1G1oN9ZR

Polecenie 2

Przygotuj notatkę ze swoimi spostrzeżeniami dotyczącymi symulacji.

Sprawdź się

Pokaż ćwiczenia: 

Ćwiczenie 1

Napisz program, który obliczy liczbę wywołań rekurencyjnych funkcji `fibonacci()` dla obliczenia elementu ciągu Fibonacciego o indeksie n. Przetestuj działanie swojego programu dla n o następujących wartościach: 0, 1, 2, 7, 9, 11.

Specyfikacja:

Dane:

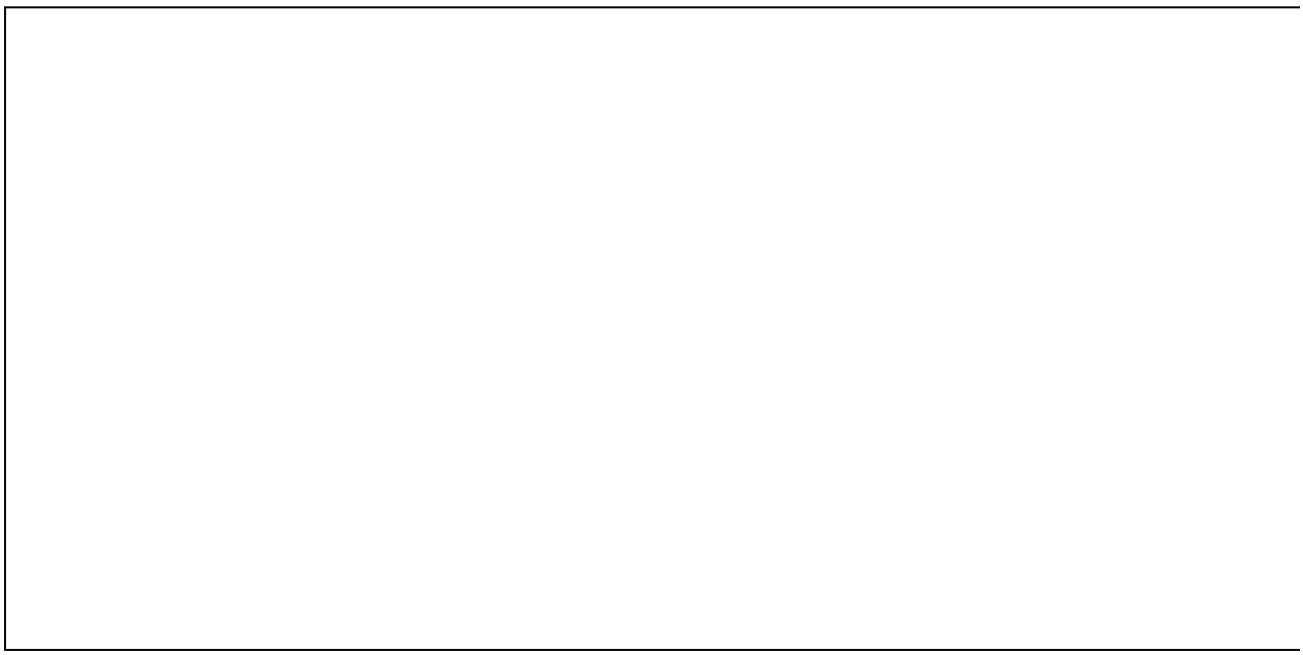
- n – liczba naturalna, indeks elementu ciągu Fibonacciego

Wynik:

Program wyświetla liczbę wywołań rekurencyjnych funkcji `fibonacci` dla obliczenia elementu ciągu Fibonacciego o indeksie n.

Twoje zadania

1. Funkcja `ileWywolan()` zwraca liczbę wywołań rekurencyjnych funkcji `fibonacci` dla obliczenia elementu ciągu Fibonacciego o indeksie n. Program wypisuje liczby wywołań dla kilku różnych wartości n. Wyniki oddzielone są pojedynczym znakiem spacji.



Ćwiczenie 2

Napisz program, który obliczy element ciągu Fibonacciego o indeksie n, a następnie sprawdzi, czy ten element jest liczbą pierwszą i wypisze odpowiedni komunikat. Przetestuj działanie swojego programu dla elementu ciągu Fibonacciego o indeksie równym 11.

Specyfikacja:

Dane:

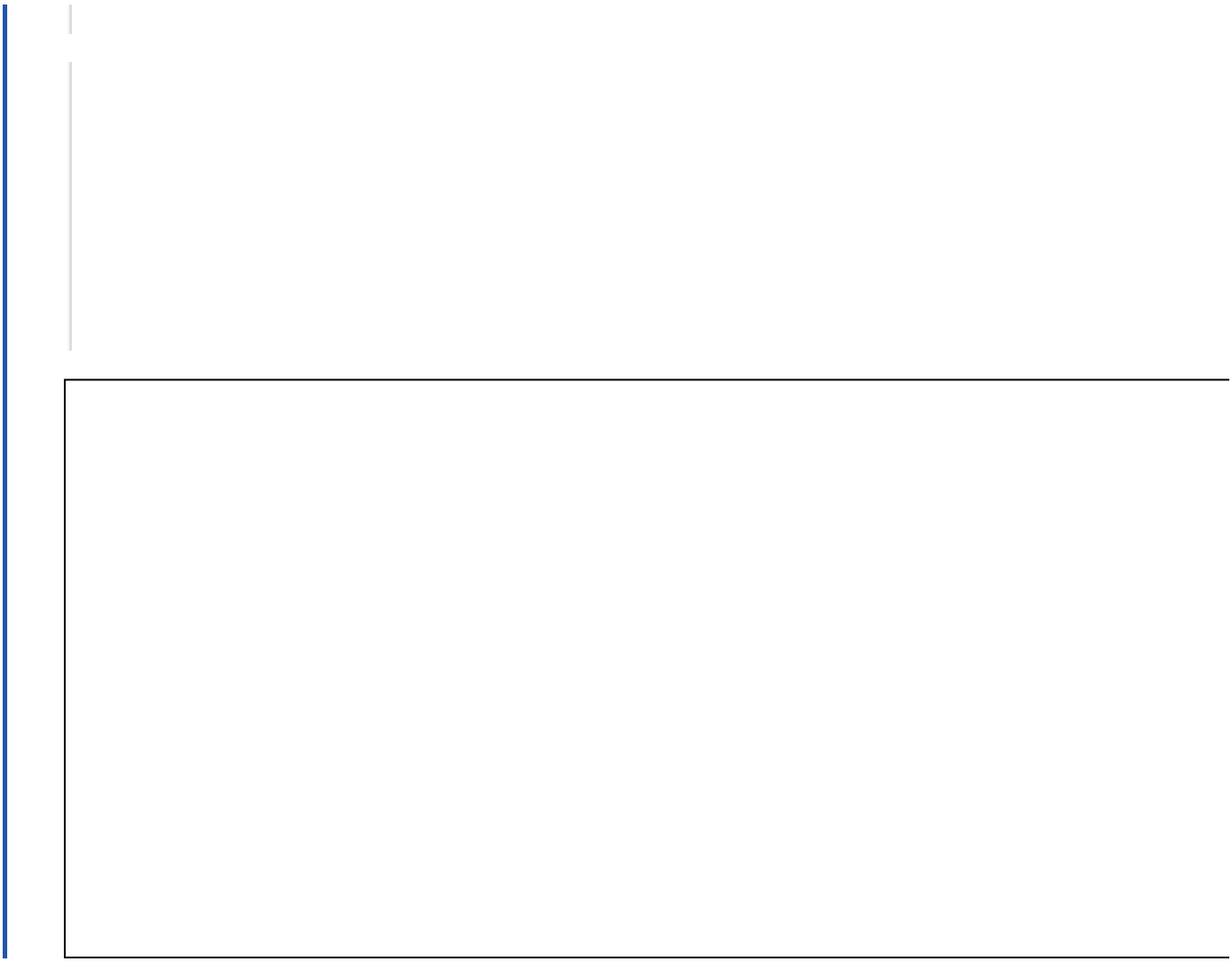
- n – liczba naturalna, indeks elementu ciągu Fibonacciego

Wynik:

Program wyświetla element ciągu Fibonacciego o indeksie n oraz komunikat „To jest liczba pierwsza”, jeżeli element ciągu Fibonacciego o indeksie n jest liczbą pierwszą lub „To nie jest liczba pierwsza”, jeżeli element o indeksie n nie jest liczbą pierwszą.

Twoje zadania

1. Program wypisuje element ciągu Fibonacciego o indeksie n oraz po znaku spacji komunikat „To jest liczba pierwsza”, jeżeli ten element ciągu jest liczbą pierwszą lub „To nie jest liczba pierwsza”, jeżeli ten element ciągu nie jest liczbą pierwszą.



Ćwiczenie 3

Napisz program, który obliczy liczbę wywołań rekurencyjnych funkcji `fibonacci()` z argumentami 2, 1 oraz 0 przy obliczaniu elementu ciągu Fibonacciego o indeksie n. Przetestuj działanie swojego programu dla elementu ciągu o indeksie równym 13.

Specyfikacja:

Dane:

- n – liczba naturalna, indeks elementu ciągu Fibonacciego

Wynik:

Program wyświetla liczbę wywołań rekurencyjnych `fibonacci(2)`, `fibonacci(1)` oraz `fibonacci(0)` przy obliczaniu elementu ciągu Fibonacciego o indeksie n.

Twoje zadania

1. Program wypisuje trzy wartości: liczbę wywołań `fibonacci(2)`, `fibonacci(1)` oraz `fibonacci(0)` przy obliczaniu elementu ciągu Fibonacciego o indeksie równym n.

Dla nauczyciela

Autor: Zespół autorski Contentplus.pl sp. z o.o.

Przedmiot: Informatyka

Temat: Ciąg Fibonacciego w języku C++

Grupa docelowa:

Liceum ogólnokształcące i technikum, liceum ogólnokształcące, technikum, zakres podstawowy

Podstawa programowa:

Cele kształcenia – wymagania ogólne

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

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.

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 algorytm generujący kolejne elementy ciągu Fibonacciego zapisany za pomocą języka C++.
- Napiszesz programy wyznaczające elementy ciągu Fibonacciego w sposób rekurencyjny oraz iteracyjny.
- Rozwiążesz kilka zadań związanych z tematem e-materiału.

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. Uczniowie powtarzają informacje zawarte we wcześniejszych e-materiałach dotyczących ciągu Fibonacciego oraz rekurencji.

Faza wstępna:

1. Przedstawienie tematu zajęć oraz wspólne z uczniami ustalenie kryteriów sukcesu.
2. **Rozpoznanie wiedzy uczniów.** Uczniowie tworzą pytania dotyczące tematu zajęć, na które odpowiedzą w trakcie lekcji.

Faza realizacyjna:

1. **Praca z tekstem.** Uczniowie przystępują do cichego czytania tekstu e-materiału. Indywidualnie zapoznają się z treścią w sekcji „Przeczytaj".
2. Uczniowie wykonują polecenie, które znajduje się w sekcji „Przeczytaj".
3. Uczniowie w parach wykonują ćwiczenie nr 1.
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 raz jeszcze wyświetla na tablicy temat lekcji i cele zawarte w sekcji „Wprowadzenie". W odniesieniu do ich realizacji dokonuje szczegółowej oceny rozwiązania zastosowanego przez wybranego ucznia.Nauczyciel zadaje pytania podsumowujące, np.
   – czym jest rekurencja?
   – na czym polega mechanizm rekurencji?
   – jak nazywamy element określony wraz z deklaracją funkcji, który pozwoli na odniesienie się do wartości przekazanej przy wywołaniu funkcji?
   – co to jest argument funkcji?
2. Wybrany uczeń podsumowuje zajęcia z programowania w C++, zwracając uwagę na nabyte umiejętności.

Praca domowa:

1. Uczniowie wykonują ćwiczenie nr 3 z sekcji „Sprawdź się".

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", „Symulacja interaktywna", „Sprawdź się" jako materiał do lekcji powtórkowej.