Ciąg Fibonacciego – zadania maturalne

Źródło: Jason Leung, domena publiczna.

W e-materiale Ciąg Fibonacciego dowiedzieliśmy się, jak wykorzystać metodę rekurencyjną do wygenerowania kolejnych elementów ciągu Fibonacciego. W tym e-materiale rozwiążemy zadania maturalne związane z tym zagadnieniem.

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

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

Implementacje algorytmów wykorzystujących ciąg Fibonacciego w wybranych językach programowania znajdziesz w e-materiałach:

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

> **Twoje cele**
>
> - Przeanalizujesz rozwiązania zadania maturalnego sprawdzającego wiedzę na temat ciągu Fibonacciego.

- W wybranym języku oprogramowania zaimplementujesz program generujący elementy ciągu Fibonacciego, spełniające określone warunki.
- Rozwiążesz przykładowe zadanie maturalne dotyczące ciągu Fibonacciego.

Przeczytaj

Zadanie 1

Wiązka zadań *Binarny fraktal Fibonacciego*

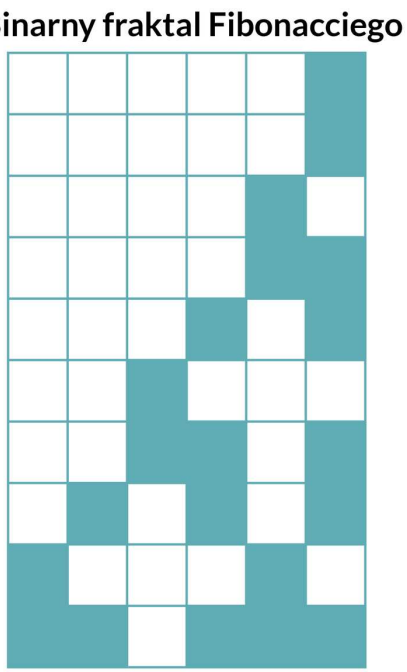
Ciąg Fibonacciego to ciąg liczb naturalnych określony rekurencyjnie w sposób następujący: $\mathrm{F}_1 = 1, \mathrm{F}_2 = 1$, a każdy następny element ciągu jest sumą dwóch poprzednich, czyli:

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

Binarny fraktal Fibonacciego to dwuwymiarowa tablica, zawierająca w kolejnych wierszach binarne zapisy kolejnych liczb Fibonacciego, gdzie każde zero w zapisie zastąpiono białym kwadratem, a każdą jedynkę czarnym kwadratem (Rys.1). Wszystkie binarne zapisy powinny składać się z jednakowej liczby cyfr, czyli do zapisów krótszych niż najdłuższy należy dodać zera wiodące.

Przykład binarnego fraktala dla pierwszych 10 liczb Fibonacciego:

n	F_n	zapis binarny F_n
1	1	000001
2	1	000001
3	2	000010
4	3	000011
5	5	000101
6	8	001000
7	13	001101
8	21	010101
9	34	100010
10	55	110111



Rys.1. Fraktal binarny dla pierwszych 10 liczb Fibonacciego.

Napisz program komputerowy, za pomocą którego uzyskasz odpowiedzi do poniższych zadań. Rysunek fraktala (zadanie 1.3) wykonaj, wykorzystując dostępne narzędzia

informatyczne. Odpowiedzi do poszczególnych zadań zapisz w pliku tekstowym o nazwie `wyniki.txt`, natomiast rysunek fraktala w pliku `fraktal.xxx`, gdzie `xxx` oznacza rozszerzenie pliku, w którym zapisany jest obraz fraktala.

Zadanie 1.1

Podaj wartości F_{10}, F_{20}, F_{30}, F_{40}. Zapisz każdą z liczb w osobnym wierszu.

Zadanie 1.2

Znajdź wszystkie liczby pierwsze wśród liczb F_1, F_2, ..., F_{40}. Zapisz każdą z liczb w osobnym wierszu.

Zadanie 1.3

Dla pierwszych 40 liczb Fibonacciego utwórz binarny fraktal Fibonacciego:

- Wypisz reprezentację binarną wszystkich liczb Fibonacciego od F_1 do F_{40}.
- Wyrównaj długości reprezentacji binarnych wszystkich liczb Fibonacciego od F_1 do F_{40} i na ich podstawie sporządź obraz binarnego fraktala Fibonacciego.

Zadanie 1.4

Podaj w zapisie binarnym wyrazy ciągu Fibonacciego z zakresu od F_1 do F_{40}, które w tym zapisie mają dokładnie 6 jedynek.

> Wiązka zadań została opracowana przez CKE i pojawiła się w zbiorze zadań maturalnych z informatyki jako zadanie 67. Cały zbiór można znaleźć na stronie internetowej Centralnej Komisji Egzaminacyjnej.
> **Praca domowa**
>
> Przedstaw rozwiązanie zadania w postaci programu w języku C++, Java lub Python.

Rozwiązanie

> Na egzaminie maturalnym uczniowie mają swobodę wyboru języka programowania. Z tego powodu rozwiązanie zadania przedstawimy w pseudokodzie. Twoim zadaniem jest napisanie kodu w języku, w którym programujesz.

Zadanie 1.1

Rozwiązanie pierwszego podpunktu sprowadza się do zapisania algorytmu wyznaczania kolejnych wyrazów ciągu Fibonacciego. Skorzystamy z iteracyjnej wersji algorytmu,

używającej pomocniczych zmiennych, przechowujących dwa wcześniejsze wyrazy ciągu. Jednak nic nie stoi na przeszkodzie, by wykorzystać wersję rekurencyjną algorytmu.

```
1 F1 ← 1
2 F2 ← 1
3 dla i = 3, 4, ..., 40 wykonuj:
4     pom ← F1
5     F1 ← F2
6     F2 ← pom + F1
7 
8     jeżeli i mod 10 = 0:
9         wypisz F2
```

1. Inicjujemy dwa pierwsze wyrazy ciągu Fibonacciego. **[linie kodu 1, 2]**
2. Zapisujemy pętlę `dla`, wyliczającą kolejne wyrazy ciągu, aż do czterdziestego. **[linie kodu 3-6]**
3. Jeżeli właśnie obliczyliśmy wyraz ciągu, którego numer jest podzielny przez 10, to wypisujemy go. **[linie kodu 8, 9]**

Zadanie 1.2

Zmodyfikujemy rozwiązanie pierwszego podpunktu – zamiast wypisywania wyrazów ciągu o konkretnych numerach, każdy z obliczonych wyrazów będziemy sprawdzać pod kątem „bycia na pierwszym miejscu". Zapiszmy funkcję stwierdzającą, czy liczba podana jako argument jest liczbą pierwszą:

```
1 funkcja czyPierwsza(liczba)
2     jeżeli liczba < 2:
3         zwróć fałsz
4     dla i = 2, 3, ..., pierwiastekCałkowity(liczba):
5         jeżeli liczba mod i = 0:
6             zwróć fałsz
7     zwróć prawdę
```

1. Zapisujemy nagłówek funkcji. **[linia kodu 1]**
2. Jeżeli liczba jest mniejsza od 2 (0 lub 1), nie jest liczbą pierwszą, zatem zwracamy fałsz. **[linie kodu 2, 3]**
3. Sprawdzamy podzielność argumentu funkcji przez kolejne liczby, aż do jego pierwiastka. W przypadku wystąpienia podzielności, zwracamy fałsz. **[linie kodu 4-6]**

4. Jeżeli pętlą zakończyła się bez zwrócenia fałszu, liczba dzieli się tylko przez 1 i samą siebie, zatem jest liczbą pierwszą i możemy zwrócić prawdę. **[linia kodu 7]**

```
1 F1 ← 1
2 F2 ← 1
3 dla i = 3, 4, ..., 40 wykonuj:
4     pom ← F1
5     F1 ← F2
6     F2 ← pom + F1
7 
8     jeżeli czyPierwsza(F2):
9         wypisz F2
```

Przedstawiona funkcja jest analogiczna do rozwiązania pierwszego podpunktu zadania. W linii 8 zamiast sprawdzania, czy dany numer wyrazu ciągu jest podzielny przez 10, sprawdzamy, czy jest liczbą pierwszą.

Zadanie 1.3

Podpunkt trzeci zadania podzielimy na dwie części. W pierwszej, korzystając z rozwiązania pierwszego podpunktu, wyliczymy pierwsze 40 wyrazów ciągu Fibonacciego, przekonwertujemy je do postaci binarnej i zapiszemy w tablicy. W drugiej, znając długość najdłuższego obliczonego wyrazu zapisanego binarnie, dopełnimy krótsze od niego wyrazy zerami z przodu i wypiszemy poprawione wyrazy.

Zapiszmy funkcję konwertującą wyrazy ciągu Fibonacciego do postaci binarnej. W związku z koniecznością dopisywania zer na początek krótszych liczb w dalszej części zadania, przekonwertowane liczby zapiszemy jako łańcuchy znaków:

```
1 funkcja konwertuj(liczba):
2     postacBinarna ← ""
3     dopóki liczba > 0, wykonuj:
4         jeżeli liczba mod 2 = 0:
5             postacBinarna ← "0" + postacBinarna
6         w przeciwnym wypadku:
7             postacBinarna ← "1" + postacBinarna
8         liczba ← liczba div 2
9     zwróć postacBinarna
```

1. Zapisujemy nagłówek funkcji oraz pusty łańcuch znaków, do którego będziemy dopisywać kolejne cyfry w trakcie konwersji. **[linie 1, 2]**
2. Dopóki argument funkcji jest większy od 0, dzielimy go całkowitoliczbowo przez 2, za każdym razem dopisując resztę z dzielenia na początek postaci binarnej. **[linie kodu 3-8]**
3. Zbudowany łańcuch znaków zwracamy. **[linia kodu 9]**

Możemy zapisać główną część kodu. Modyfikujemy rozwiązanie z podpunktu pierwszego, zapisując każdą przekonwertowaną liczbę w pomocniczej tablicy. Na koniec sprawdzamy długość ostatniego zapisanego w tablicy wyrazu i dla krótszych wyrazów dopisujemy z przodu 0, a następnie zapisujemy wyrazy do pliku wynikowego.

```
1  F1 ← 1
2  F2 ← 1
3
4  Binarne[0] ← "1"
5  Binarne[1] ← "1"
6  dla i = 2, 3, ..., 39 wykonuj:
7      pom ← F1
8      F1 ← F2
9      F2 ← pom + F1
10     Binarne[i] ← konwertuj(F2)
11 dla i = 0, 1, ..., 39 wykonuj:
12     różnica ← długość(Binarne[39]) - długość(Binarne[i])
13     jeżeli różnica > 0:
14         dla j = 0, 1, ..., różnica:
15             Binarne[i] ← "0" + Binarne[i]
16     wypisz Binarne[i]
```

1. Oprócz pierwszych wyrazów ciągu Fibonacciego deklarujemy tablicę łańcuchów znaków, która będzie przechowywać wyrazy w postaci binarnej. Dwa pierwsze elementy tablicy wypełniamy od razu, gdyż nie ma dla nich różnicy między systemem binarnym oraz dziesiętnym. **[linie kodu 1-5]**
2. W pierwszej pętli obliczamy kolejne wyrazy ciągu Fibonacciego oraz zapisujemy ich postać binarną do tablicy `Binarne[]`. **[linie kodu 6-10]**
3. Następnie, dla każdego z wyrazów w postaci binarnej, dopisujemy zera z przodu, aby zrównały się długością z najdłuższym obliczonym wyrazem, a następnie zapisujemy je do pliku wynikowego. **[linie kodu 11-16]**

Zadanie 1.4

W ostatnim podpunkcie zadania wykorzystamy zmodyfikowaną funkcję konwertującą z podpunktu trzeciego, przy okazji sprawdzając liczbę jedynek w postaci binarnej.

```
1  funkcja konwertuj(liczba):
2      postacBinarna ← ""
3      liczbaJedynek ← 0
4      dopóki liczba > 0, wykonuj:
5          jeżeli liczba mod 2 = 0:
6              postacBinarna ← "0" + postacBinarna
7          w przeciwnym wypadku:
8              postacBinarna ← "1" + postacBinarna
9              liczbaJedynek ← liczbaJedynek + 1
10         liczba ← liczba div 2
11     jeżeli liczbaJedynek = 6:
12         wypisz postacBinarna
```

Do funkcji wykorzystanej we wcześniejszym poleceniu dodajemy zmienną `liczbaJedynek`, zwiększaną za każdym razem, gdy w postaci binarnej liczby pojawi się jedynka. Na koniec konwersji sprawdzamy, czy jej wartość jest równa 6. Jeśli tak, wypisujemy postać binarną liczby. W przeciwnym wypadku funkcja kończy się bez żadnego skutku.

```
1  F1 ← 1
2  F2 ← 1
3  dla i ← 3, 4, ..., 40 wykonuj:
4      pom ← F1
5      F1 ← F2
6      F2 ← pom + F1
7      konwertuj(F2)
```

1. Inicjujemy dwa pierwsze wyrazy ciągu Fibonacciego. **[linie kodu 1, 2]**
2. Zapisujemy pętlę `dla`, wyliczającą kolejne wyrazy ciągu, aż do czterdziestego. **[linie kodu 3–6]**
3. Wywołujemy zmodyfikowaną funkcję konwertującą, która wypisuje liczby w postaci binarnej, tylko jeżeli mają 6 jedynek. **[linia kodu 9]**

Odpowiedzi do zadania

Zadanie 1.1

```
55
6765
832040
102334155
```

Zadanie 1.2

```
2
3
5
13
89
233
1597
28657
514229
```

Zadanie 1.3

```
00000000000000000000000001
00000000000000000000000001
00000000000000000000000010
00000000000000000000000011
00000000000000000000000101
00000000000000000000001000
00000000000000000000001101
00000000000000000000010101
00000000000000000000100010
00000000000000000000110111
00000000000000000001011001
00000000000000000010010000
00000000000000000011101001
00000000000000000101111001
00000000000000001001100010
00000000000000001111011011
00000000000000011000111101
00000000000000101000011000
00000000000001000001010101
00000000000001101001101101
00000000000010101011000010
```

```
00000000000010001010010111
00000000000011011111110001
00000000000101101010010000
00000000001001001010001001
00000000001110110100011001
00000000101111111101000010
00000001001101100101110011
00000001111110110001011010
00000011001011001000101000
00000101001000101011011101
00001000010011110100000101
00001101011100011111100010
00010101110000010011100111
00010001100110011001100100
00011100011110100011010000
00101110000100111001111001
01001010100011100000101001
01111000101000011101010010
11000011001011111011001011
```

Zadanie 1.4

```
101111001
10101011000010
1011010100100000
10010010100010001
```

Słownik

argument funkcji

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

czynnik pierwszy

liczba pierwsza, będąca dzielnikiem danej liczby

funkcja

wydzielona część kodu, nazywana także podprogramem, wykonująca określone operacje, możliwa do wywołania wiele razy podczas działania programu

instrukcja `dla`

fragment kodu, dla którego zdefiniowana zostaje zmienna iteracyjna; w każdym przejściu instrukcji wartość tej zmiennej jest modyfikowana w pewien określony sposób; w momencie, gdy instrukcja wykona się dla wszystkich podanych wartości zmiennej iteracyjnej, kończy swoje działanie

instrukcja warunkowa

element języka programowania, który pozwala na wykonanie rozgałęzienia w programie i tym samym na wykonanie różnych instrukcji, w zależności od tego czy zdefiniowane przez programistę wyrażenie logiczne jest prawdziwe czy fałszywe

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)

tablica

kontener uporządkowanych danych takiego samego typu; do poszczególnych elementów odnosimy się za pomocą unikatowych indeksów

Aplet

Polecenie 1

Rozważ następującą funkcję rekurencyjną, wyznaczającą n-ty wyraz ciągu Fibonacciego, a następnie zapoznaj się z apletem prezentującym jej kolejne wywołania.

```
funkcja fib(n)
    jeżeli n < 2
        zwróć n
    zwróć fib(n - 1) + fib(n - 2)
```


Zasób interaktywny dostępny pod adresem https://zpe.gov.pl/a/DQkOVc10H

Zadanie 2

Bartek zapisał na kartce 1000 par liczb, każda para w osobnej linii. Pierwsza liczba oznacza klucz danej linii, druga natomiast jej wartość.

W pliku `liczby.txt` znajduje się 1000 par liczb, każda w osobnej linii. Każdy klucz to liczba naturalna z przedziału <1, 2000>.

Plik o rozmiarze 9.12 KB w języku polskim

Polecenie 2

Napisz program, który dla pliku `liczby.txt` wyznaczy linie, których klucze są wyrazami ciągu Fibonacciego, a następnie wypisze tylko parzyste wartości tych linii. Wyniki zapisz w pliku `wynik.txt`. Pamiętaj, że linie numerujemy od 1.

```
1
```

Przykład 1

Dla przykładowych danych:

```
50 222
```

```
3 12
1 15
1 300
51 292
100 327
```

Plik wynikowy powinien wyglądać następująco:

```
12
300
```

Do oceny oddajesz:

- plik `wynik.txt`, zawierający odpowiedź do zadania (wartości – drugie liczby – linii z pliku `liczby.txt`, które są parzyste oraz których klucze – pierwsze liczby z linii – są liczbami ciągu Fibonacciego),
- plik(i) z komputerową realizacją zadania (kodem programu)·

Odpowiedź do zadania

Twoim zadaniem jest opracowanie rozwiązania zadania w wybranym przez siebie języku programowania: C++, Java lub Python. Odpowiedź do zadania dla danych z pliku znajdziesz na dole strony.

Rozwiązanie

Rozwiązanie zadania przedstawimy w postaci pseudokodu, ponieważ na egzaminie maturalnym można korzystać z wybranego języka programowania: C++, Java lub Python.

Na początku wczytujemy dane z pliku do tablicy. Następnie chcemy obliczyć elementy ciągu Fibonacciego, które wynoszą maksymalnie 2000. Dzięki temu nie będziemy musieli wykonywać tego za każdym razem, gdy będziemy sprawdzać klucz. Zapisujemy je w tablicy. Gdy jest ona gotowa, tworzymy dwie pętle, które sprawdzają, czy klucz linii (pierwsza liczba) jest elementem ciągu Fibonacciego, oraz czy wartość linii (druga liczba) jest liczbą parzystą. Jeżeli tak jest, wpisujemy wartość linii do pliku wynikowego i przerywamy pętlę wewnętrzną.

```
dla i = 0, 1, ..., 999 wykonuj:
    liczby[i][0] ← wczytaj pierwszą liczbę z linii z pliku "liczb
    liczby[i][1] ← wczytaj drugą liczbę z linii z pliku "liczby.t
    przejdź do nowej linii w pliku "liczby.txt"
```

```
5
6  wartość ← 1
7
8  fibonacci.dodaj(0)
9
10 dopóki wartość < 2000:
11     fibonacci.dodaj(wartość)
12     wartość ← wartość + fibonacci[długość(fibonacci) - 2]
13
14 fibonacci.dodaj(wartość)
15
16 dla i = 0, 1, ..., 999:
17     dla j = 0, 1, ..., długość(fibonacci):
18         jeżeli liczby[i][0] == fibonacci[j] oraz liczby[i][1] mod
19             wpisz liczby[i][1] do nowej linii w pliku "wynik.txt"
20             przerwij pętlę
```

> **Ważne!**
>
> Funkcja `dodaj(x)` oznacza dodanie elementu `x` na koniec zbioru danych (tablicy), na którym jest wywoływana, natomiast `mod` oznacza operację modulo.

Odpowiedź do zadania

Odpowiedź do zadania znajduje się w pliku:

Plik o rozmiarze 41.00 B w języku polskim

Sprawdź się

Zadanie 3 – Sumy wyrazów ciągu Fibonacciego

W pliku tekstowym `dane3.txt` załączonym do zadania znajduje się 50 wierszy, zawierających dodatnie liczby naturalne. Wynik zapisz do pliku `wynik3.txt`.

Plik o rozmiarze 222.00 B w języku polskim

Używając wybranego języka programowania, napisz program, który wczyta zawartość pliku, a następnie obliczy i wypisze, ile liczb z tego pliku jest sumą dokładnie trzech **różnych** wyrazów ciągu Fibonacciego. Pamiętaj, że zarówno drugi, jak i trzeci wyraz mają wartość 1, jednak są różnymi wyrazami.

W ramach ćwiczenia część danych została umieszczona w formie tabeli. Pamiętaj jednak, że podczas pisania implementacji na swoim komputerze musisz się upewnić, że dane zostaną prawidłowo wczytane z pliku tekstowego do programu.

Poprawny wynik dla danych z podanego pliku tekstowego znajduje się na końcu zadania.

Do oceny oddajesz:

- plik `wynik3.txt` z odpowiedzią (liczba naturalna oznaczająca to, ile liczb z pliku `dane3.txt` jest sumą dokładnie trzech różnych wyrazów ciągu Fibonacciego),
- plik(i) z komputerową realizacją zadania (kodem programu).

Pokaż ćwiczenia:

Ćwiczenie 1

C++

Twoje zadania

1. Program oblicza i wypisuje, ile liczb jest sumą dokładnie trzech różnych wyrazów ciągu Fibonacciego.

```cpp
#include <iostream>

using namespace std;

void ileLiczbPotrojnych(int tab[], int n) {
    int ilosc = 0;
    //uzupełnij funkcję ileLiczbPotrojnych()
    cout << ilosc << endl;
}

int main() {
    int liczby[] = {2, 4, 11, 35, 46, 49, 51, 54};
    ileLiczbPotrojnych(liczby, 8);
    return 0;
```

```
1
```

Ćwiczenie 2

Java

Twoje zadania

1. Program oblicza i wypisuje, ile liczb jest sumą dokładnie trzech różnych wyrazów ciągu Fibonacciego.

```java
public class Main {
    public static int[] liczby = {2, 4, 11, 35, 46, 49, 51, 54};

    public static void main(String[] args) {
        ileLiczbPotrojnych(liczby);
    }

    public static void ileLiczbPotrojnych(int[] tab) {
        int ilosc = 0;
        // uzupełnij tę funkcję
        System.out.println(ilosc);
    }
}
```

```
1
```

Ćwiczenie 3

Python

Twoje zadania

1. Program oblicza i wypisuje, ile liczb jest sumą dokładnie trzech różnych wyrazów ciągu Fibonacciego.

```python
def ileLiczbPotrojnych(tab):
    ilosc = 0
    #uzupełnij funkcję ileLiczbPotrojnych()
    print(ilosc)


liczby = [2, 4, 11, 35, 46, 49, 51, 54]
ileLiczbPotrojnych(liczby)
```

```
1
```

Odpowiedź do zadania

Plik o rozmiarze 2.00 B w języku polskim

Dla nauczyciela

Autor: Maurycy Gast
Przedmiot: Informatyka

Temat: Ciąg Fibonacciego – zadania maturalne

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:

2) do realizacji rozwiązania problemu dobiera odpowiednią metodę lub technikę algorytmiczną i struktury danych;

3) objaśnia dobrany algorytm, uzasadnia poprawność rozwiązania na wybranych przykładach danych i ocenia jego efektywność;

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.

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

2) stosuje zasady programowania strukturalnego i obiektowego w rozwiązywaniu problemów;

3) sprawnie posługuje się zintegrowanym środowiskiem programistycznym przy pisaniu, uruchamianiu i testowaniu programów;

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 rozwiązania zadania maturalnego sprawdzającego wiedzę na temat ciągu Fibonacciego.
- W wybranym języku oprogramowania zaimplementujesz program generujący elementy ciągu Fibonacciego, spełniające określone warunki.
- Rozwiążesz przykładowe zadanie maturalne dotyczące ciągu Fibonacciego.

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;
- oprogramowanie dla języka Java SE 8 (lub nowszej wersji), w tym Eclipse 4.4 (lub nowszej wersji);
- oprogramowanie dla języka Python 3 (lub nowszej wersji), w tym PyCharm lub IDLE.

Przebieg lekcji

Przed lekcją:

1. Uczniowie powtarzają najważniejsze informacje dotyczące ciągu Fibonaciego.
2. **Przygotowanie do zajęć.** Nauczyciel loguje się na platformie i udostępnia e-materiał: „Ciąg Fibonacciego – zadania maturalne". Nauczyciel prosi uczniów o zapoznanie się z treściami w sekcji „Przeczytaj".

Faza wstępna:

1. Chętna lub wybrana osoba referuje najważniejsze informacje dotyczące ciągu Fibonacciego. W razie potrzeby nauczyciel uzupełnia jej wypowiedź.
2. Nauczyciel prosi wybraną osobę o odczytanie tematu lekcji, a następnie określa cele i kryteria sukcesu.
3. Prowadzący prosi uczniów, aby zgłaszali swoje propozycje pytań do tematu. Jedna osoba może zapisywać je na tablicy. Gdy uczniowie wyczerpią swoje pomysły, a pozostały jakieś ważne kwestie do poruszenia, nauczyciel je dopowiada.

Faza realizacyjna:

1. **Praca z tekstem.** Uczniowie analizują rozwiązania zadania 1 i poszczególnych jego podpunktów zaprezentowane w sekcji „Przeczytaj". W parach implementują je w wybranych językach programowania. Chętne lub wybrane pary prezentują swoje rozwiązania. W razie potrzeby nauczyciel je omawia.
2. **Praca z multimedium.** Uczniowie analizują treść sekcji „Aplet", a następnie wykonują polecenie nr 2.
3. **Ćwiczenie umiejętności.** Uczniowie rozwiązują zadanie 3 z sekcji „Sprawdź się". Następnie dobierają się w pary i analizują nawzajem swoje rozwiązania. W kolejnym kroku chętne lub wybrane osoby prezentują swoje rozwiązania.

Faza podsumowująca:

1. 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.
2. Wybrany uczeń podsumowuje zajęcia, zwracając uwagę na nabyte umiejętności, omawia ewentualne problemy podczas rozwiązania ćwiczeń z programowania w języku Python.

Praca domowa:

1. Uczniowie wykonują Zadanie 2 z sekcji „Aplet".

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 języka Java SE 8 (lub nowszej wersji).
- Oficjalna dokumentacja techniczna dla oprogramowania Eclipse 4.4 (lub nowszej wersji).
- Oficjalna dokumentacja techniczna dla języka Python 3 (lub nowszej wersji).
- Oficjalna dokumentacja techniczna dla oprogramowania PyCharm lub IDLE.

Wskazówki metodyczne:

- Uczniowie mogą wykorzystać treści w sekcjach: „Przeczytaj", „Aplet", „Sprawdź się" jako materiał do lekcji powtórkowej.