Lekcja 10 (PR)

Temat: Wyznaczanie wyrazów ciągu Fibonacciego.

Cele lekcji:

1. Uczeń zna pojęcie rekurencji
2. Uczeń zna zasady dotyczące ciągu
3. Podaje definicję ciągu
4. Stosuje ciąg Fibonacciego do rozwiązywania zadań

Przebieg lekcji:

1. Zapoznanie z celami lekcji
2. Definicja rekurencyjna ciągu Fibonacciego

Ciąg Fibonacciego – ciąg liczb naturalnych określony rekurencyjnie w sposób następujący:

Pierwszy wyraz jest równy 0, drugi jest równy 1, każdy następny jest sumą dwóch poprzednich.

Formalnie:

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

Kolejne wyrazy tego ciągu nazywane są **liczbami** Fibonacciego. Zaliczanie zera do elementów ciągu Fibonacciego zależy od umowy – część autorów definiuje ciąg od F1=F2=1

Pierwsze dwadzieścia wyrazów ciągu Fibonacciego to:

F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}	F_{16}	F_{17}	F_{18}	F_{19}
0	1	1	2	3	5	8	13	21	34	55	89	144	233	377	610	987	1597	2584	4181

Ciąg został omówiony w roku 1202 przez Leonarda z Pizy, zwanego Fibonaccim, w dziele *Liber abaci* jako rozwiązanie zadania o rozmnażaniu się królików. Nazwę „ciąg Fibonacciego” spopularyzował w XIX w. Édouard Lucas.

3. Programowanie:
   a) program bez rekurencji

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

void fibonacci(int n)
{
     long long a = 0, b = 1;

     for(int i=0;i<n;i++)
     {
            cout<<b<<" ";
            b += a; //pod zmienną b przypisujemy wyraz następny czyli a+b
            a = b-a; //pod zmienną a przypisujemy wartość zmiennej b
     }
}

int main()
{
    int n;

    cout<<"Podaj ile chcesz wypisać wyrazów ciągu fibonacciego: ";
    cin>>n;

    fibonacci(n);

    system("pause");
    return 0;
}
```

b) Program rekurencyjny

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

int fib(int n)
{
        if(n<3)
                return 1;

        return fib(n-2)+fib(n-1);
}

int main()
{

        int n;

        cout<<"Podaj nr wyrazu ciągu: ";
        cin>>n;

        cout<<n<<" wyraz ciągu ma wartość "<<fib(n)<<endl;

        system("pause");
        return 0;
}
```

Przeanalizujmy powyższy algorytm dla

$$n = 5$$

:

$$wynik = fib(5) = \underbrace{\underbrace{\underbrace{fib(3)}_{\underbrace{fib(1)}_{1}+\underbrace{fib(2)}_{1}}+\underbrace{fib(2)}_{1}}_{fib(4)} + \underbrace{\underbrace{fib(1)}_{1}+\underbrace{fib(2)}_{1}}_{fib(3)}}_{} = 5$$

4. Zadania do ćwiczenia – podręcznik str. 42.

Zadanie 1.34. Podaj specyfikację zadania i skonstruuj algorytmy rekurencyjne w postaci programów wyznaczające n-ty wyraz podanego ciągu liczbowego. Podaj definicje rekurencyjne ciągów:

a) (4, 7, 10, 13, 16, 19, 22, 25, 28...),

b) (2, 4, 8, 16, 32, 64, 128, 256, 512...),

c) (0,2, −0,6, 1,8, −5,4, 16,2, −48,6, 145,8...),

d) (−10, 5, −2,5, 1,25, −0,625, 0,3125...),

e) (3, 5, 4, 6, 5, 7, 6, 8, 7...),

f) (1,5, 1, 0,5, −0,5, −2, −4,5, −8,5, −15...),

g) (−3, 1, −4, −5, 19, −96, −1825, 175199...),

h) (−2, 2,5, 3, −5, 7,5, −4,5, −0,5, 8, −12,5...),

i) (−1, 0, 0,5, 1,5, 1,5, 1, −0,5, −2, −3...),

j) (0, 1, −1, 2, −2, 3, −3, 4, −4...).

Zadanie 1.35. Wygeneruj ciągi liczbowe (podając wartości co najmniej siedmiu kolejnych wyrazów) na podstawie podanej definicji rekurencyjnej. Podaj specyfikacje i skonstruuj rekurencyjne algorytmy w postaci programów, wyznaczające n-ty wyraz zdefiniowanego ciągu.

a) $\begin{cases} a_1 = 2 \\ a_n = 3a_{n-1} + \frac{1}{2} \quad dla \quad n > 1 \end{cases}$

b) $\begin{cases} a_1 = 0{,}5 \\ a_2 = 1 \\ a_n = 3a_{n-2} + a_{n-1} \quad dla \quad n > 2 \end{cases}$

c) $\begin{cases} a_1 = 2 \\ a_2 = -4 \\ a_n = a_{n-2} + 2a_{n-1} + 0{,}5 \quad dla \quad n > 2 \end{cases}$

d) $\begin{cases} a_1 = -1{,}5 \\ a_2 = 0 \\ a_3 = 1{,}5 \\ a_n = 2a_{n-3} + a_{n-2} - a_{n-1} \quad dla \quad n > 3 \end{cases}$