

Rekurencja

Wykład: rekursja, funkcje rekurencyjne, wywołanie samej siebie, wyznaczanie poszczególnych liczb Fibonacciego, potęgowanie, algorytm Euklidesa

REKURENCJA

Rekurencja (z łac. *recurrere*), zwana także rekursją (z ang. *recursion*) to termin oznaczający w programowaniu sytuację, kiedy funkcja w celu zwrócenia prawidłowego wyniku wywołuje samą siebie (a dokładnie tworzy swoje kopie aż do napotkania tzw. przypadku podstawowego, dla którego funkcja może wyznaczyć wynik).



FUNKCJA REKURENCYJNA

```
function s(n:integer):integer;  
begin  
  if (n>1) then  
    s:=n*s(n-1);  
  else  
    s:=1;  
end;
```

$$s(4) = 4 * s(3)$$

?

FUNKCJA REKURENCYJNA

```
function s(n:integer):integer;  
begin  
  if (n>1) then  
    s:=n*s(n-1);  
  else  
    s:=1;  
end;
```

$$s(4) = 4 * s(3) \quad ?$$

$$s(3) = 3 * s(2) \quad ?$$

FUNKCJA REKURENCYJNA

```
function s(n:integer):integer;  
begin  
  if (n>1) then  
    s:=n*s(n-1);  
  else  
    s:=1;  
end;
```

$$s(4) = 4 * s(3) \quad ?$$

$$s(3) = 3 * s(2) \quad ?$$

$$s(2) = 2 * s(1) \quad ?$$

FUNKCJA REKURENCYJNA

```
function s(n:integer):integer;  
begin  
  if (n>1) then  
    s:=n*s(n-1);  
  else  
    s:=1;  
end;
```

$$s(4) = 4 * s(3) \quad ?$$

$$s(3) = 3 * s(2) \quad ?$$

$$s(2) = 2 * s(1) \quad ?$$

$$s(1) = 1 \quad 1$$

FUNKCJA REKURENCYJNA

```
function s(n:integer):integer;  
begin  
  if (n>1) then  
    s:=n*s(n-1);  
  else  
    s:=1;  
  end;  
end;
```

$$s(4) = 4 * s(3) \quad ?$$

$$s(3) = 3 * s(2) \quad ?$$

$$s(2) = 2 * s(1) \quad 2$$

$$s(1) = 1 \quad 1$$

FUNKCJA REKURENCYJNA

```
function s(n:integer):integer;  
begin  
  if (n>1) then  
    s:=n*s(n-1);  
  else  
    s:=1;  
  end;  
end;
```

$$s(4) = 4 * s(3) \quad ?$$

$$s(3) = 3 * s(2) \quad 6$$

$$s(2) = 2 * s(1) \quad 2$$

$$s(1) = 1 \quad 1$$

FUNKCJA REKURENCYJNA

```
function s(n:integer):integer;  
begin  
  if (n>1) then  
    s:=n*s(n-1);  
  else  
    s:=1;  
end;
```

$$s(4) = 4 * s(3) \quad 24$$

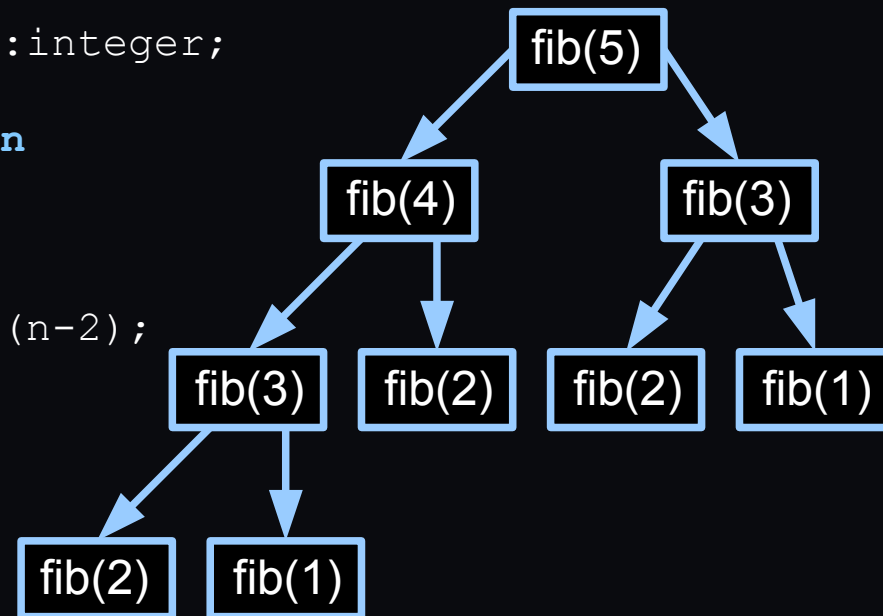
$$s(3) = 3 * s(2) \quad 6$$

$$s(2) = 2 * s(1) \quad 2$$

$$s(1) = 1 \quad 1$$

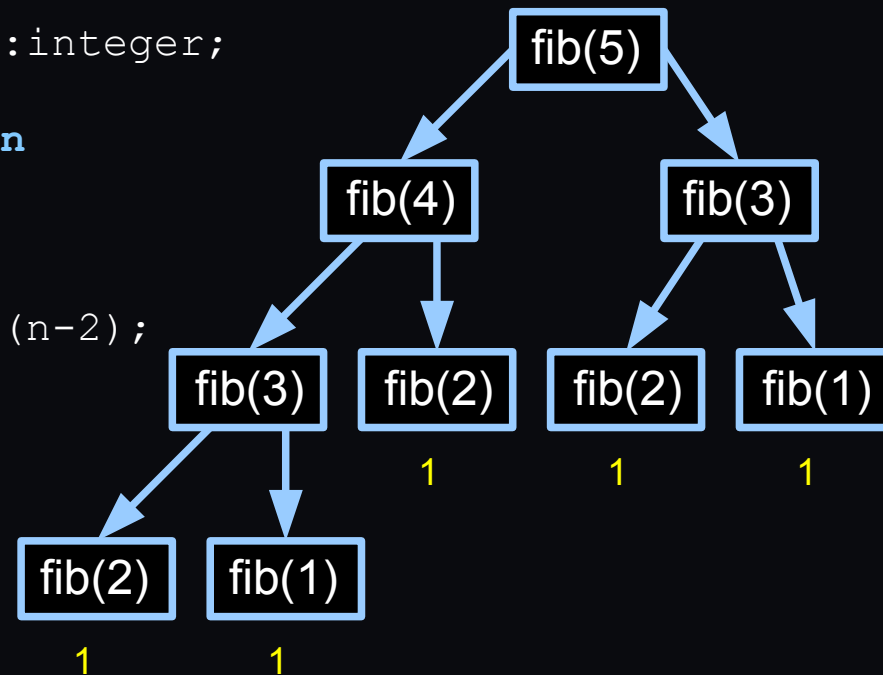
FUNKCJA WYZNACZAJĄCA N-TY WYRAZ CIĄGU FIBONACCIEGO

```
function fib(n:integer):integer;  
begin  
  if (n=1) or (n=2) then  
    fib:=1  
  else  
    if n>2 then  
      fib:=fib(n-1)+fib(n-2);  
  end;
```



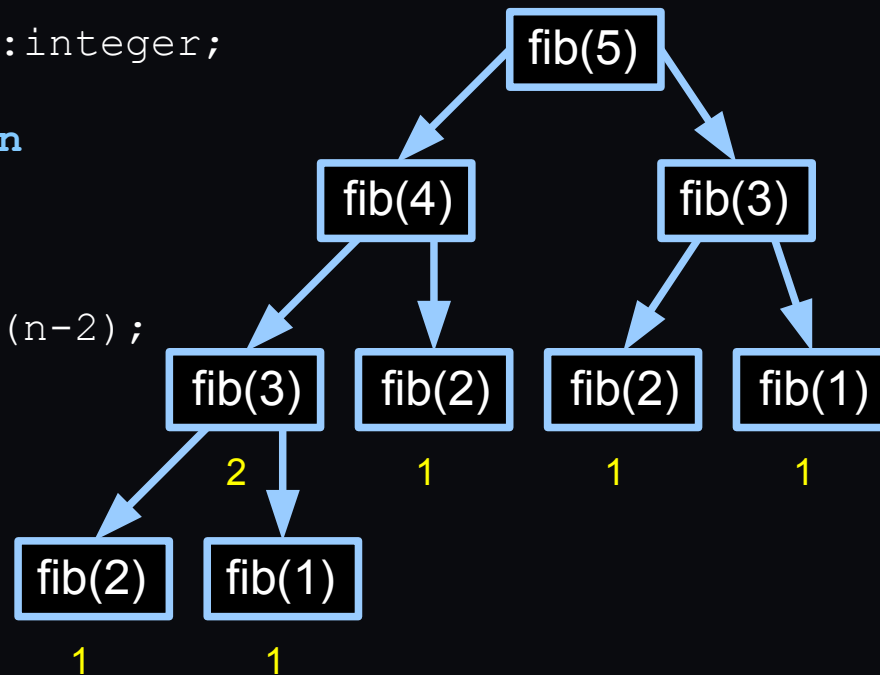
FUNKCJA WYZNACZAJĄCA N-TY WYRAZ CIĄGU FIBONACCIEGO

```
function fib(n:integer):integer;  
begin  
  if (n=1) or (n=2) then  
    fib:=1  
  else  
    if n>2 then  
      fib:=fib(n-1)+fib(n-2);  
    end;
```



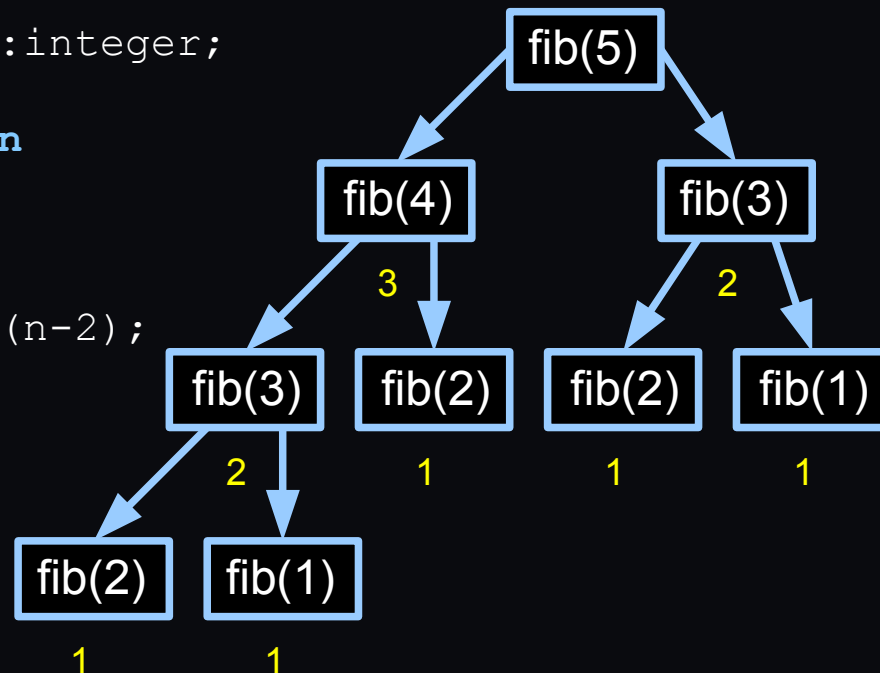
FUNKCJA WYZNACZAJĄCA N-TY WYRAZ CIĄGU FIBONACCIEGO

```
function fib(n:integer):integer;  
begin  
  if (n=1) or (n=2) then  
    fib:=1  
  else  
    if n>2 then  
      fib:=fib(n-1)+fib(n-2);  
  end;
```



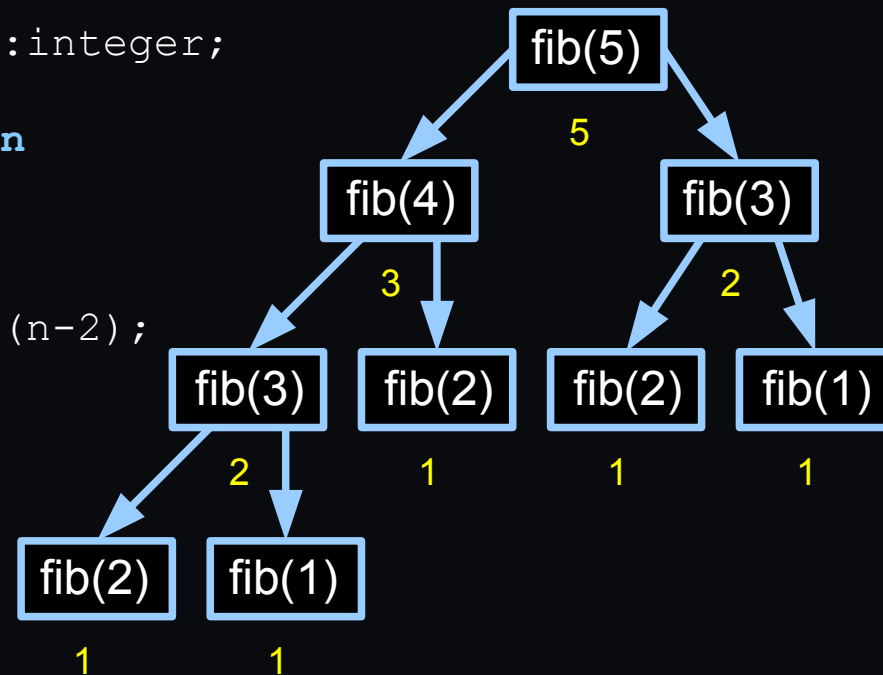
FUNKCJA WYZNACZAJĄCA N-TY WYRAZ CIĄGU FIBONACCIEGO

```
function fib(n:integer):integer;  
begin  
  if (n=1) or (n=2) then  
    fib:=1  
  else  
    if n>2 then  
      fib:=fib(n-1)+fib(n-2);  
    end;
```



FUNKCJA WYZNACZAJĄCA N-TY WYRAZ CIĄGU FIBONACCIEGO

```
function fib(n:integer):integer;  
begin  
  if (n=1) or (n=2) then  
    fib:=1  
  else  
    if n>2 then  
      fib:=fib(n-1)+fib(n-2);  
    end if  
  end if  
end;
```



POTĘGOWANIE REKURENCYJNE

```
function potega(p,w:integer):integer;  
begin  
  if w=0 then potega:=1  
  else potega:=p*potega(p,w-1); potega(2,3)=2*potega(2,2) ?  
end;
```

Obliczmy 2^3

p - podstawa potęgi
w - wykładnik potęgi

POTĘGOWANIE REKURENCYJNE

```
function potega(p,w:integer):integer;  
begin  
  if w=0 then potega:=1  
  else potega:=p*potega(p,w-1);  
  end;
```

potega(2,3) = 2 * potega(2,2) ?

potega(2,2) = 2 * potega(2,1) ?

Obliczmy 2^3

p - podstawa potęgi
w - wykładnik potęgi

POTĘGOWANIE REKURENCYJNE

```
function potega(p,w:integer):integer;  
begin  
  if w=0 then potega:=1  
  else potega:=p*potega(p,w-1);  
  end;
```

Obliczmy 2^3

p - podstawa potęgi
w - wykładnik potęgi

$$\text{potega}(2,3) = 2 * \text{potega}(2,2) \quad ?$$

$$\text{potega}(2,2) = 2 * \text{potega}(2,1) \quad ?$$

$$\text{potega}(2,1) = 2 * \text{potega}(2,0) \quad ?$$

POTĘGOWANIE REKURENCYJNE

```
function potega(p,w:integer):integer;  
begin  
  if w=0 then potega:=1  
  else potega:=p*potega(p,w-1);  
  end;
```

Obliczmy 2^3

p - podstawa potęgi
w - wykładnik potęgi

$$\text{potega}(2,3) = 2 * \text{potega}(2,2) \quad ?$$

$$\text{potega}(2,2) = 2 * \text{potega}(2,1) \quad ?$$

$$\text{potega}(2,1) = 2 * \text{potega}(2,0) \quad ?$$

$$\text{potega}(2,0) = 1 \quad 1$$

POTĘGOWANIE REKURENCYJNE

```
function potega(p,w:integer):integer;  
begin  
  if w=0 then potega:=1  
  else potega:=p*potega(p,w-1);  
  end;
```

Obliczmy 2^3

p - podstawa potęgi
w - wykładnik potęgi

$$\text{potega}(2,3) = 2 * \text{potega}(2,2) \quad ?$$

$$\text{potega}(2,2) = 2 * \text{potega}(2,1) \quad ?$$

$$\text{potega}(2,1) = 2 * \text{potega}(2,0) \quad 2$$

$$\text{potega}(2,0) = 1 \quad 1$$

POTĘGOWANIE REKURENCYJNE

```
function potega(p,w:integer):integer;  
begin  
  if w=0 then potega:=1  
  else potega:=p*potega(p,w-1);  
  end;
```

Obliczmy 2^3

p - podstawa potęgi
w - wykładnik potęgi

$$\text{potega}(2,3) = 2 * \text{potega}(2,2) \quad ?$$

$$\text{potega}(2,2) = 2 * \text{potega}(2,1) \quad 4$$

$$\text{potega}(2,1) = 2 * \text{potega}(2,0) \quad 2$$

$$\text{potega}(2,0) = 1 \quad 1$$

POTĘGOWANIE REKURENCYJNE

```
function potega(p,w:integer):integer;  
begin  
  if w=0 then potega:=1  
  else potega:=p*potega(p,w-1);  
  end;
```

Obliczmy 2^3

p - podstawa potęgi
w - wykładnik potęgi

Wynik = 8

$$\text{potega}(2,3) = 2 * \text{potega}(2,2)$$

8

$$\text{potega}(2,2) = 2 * \text{potega}(2,1)$$

4

$$\text{potega}(2,1) = 2 * \text{potega}(2,0)$$

2

$$\text{potega}(2,0) = 1$$

1

ALGORYTM EUKLIDESA

Algorytm Euklidesa – algorytm znajdowania największego wspólnego dzielnika (NWD) dwóch liczb naturalnych

Algorytm iteracyjnie:

```
function nwd(a,b:integer):integer;  
  begin  
    while (a<>b) do  
      begin  
        if (a>b) then a:=a-b else b:=b-a;  
      end;  
      nwd:=a;  
    end;
```

ALGORYTM EUKLIDESA

Algorytm rekurencyjnie:

```
function NWD(a,b:integer):integer;  
begin  
if (b=0) then NWD:=a else NWD:=NWD(b,a mod b);  
End;
```

$NWD(25, 10) = NWD(10, 5)$?

Wyznaczmy NWD 25 i 10

ALGORYTM EUKLIDESA

Algorytm rekurencyjnie:

```
function NWD(a,b:integer):integer;  
begin  
if (b=0) then NWD:=a else NWD:=NWD(b,a mod b);  
End;
```

Wyznaczmy NWD 25 i 10

$$\text{NWD}(25, 10) = \text{NWD}(10, 5) \quad ?$$



$$\text{NWD}(10, 5) = \text{NWD}(5, 0) \quad ?$$

ALGORYTM EUKLIDESA

Algorytm rekurencyjnie:

```
function NWD(a,b:integer):integer;  
begin  
if (b=0) then NWD:=a else NWD:=NWD(b,a mod b);  
End;
```

Wyznaczmy NWD 25 i 10

$$\text{NWD}(25, 10) = \text{NWD}(10, 5) \quad ?$$

$$\text{NWD}(10, 5) = \text{NWD}(5, 0) \quad ?$$

$$\text{NWD}(5, 0) = 5 \quad 5$$

ALGORYTM EUKLIDESA

Algorytm rekurencyjnie:

```
function NWD(a,b:integer):integer;  
begin  
if (b=0) then NWD:=a else NWD:=NWD(b,a mod b);  
End;
```

Wyznaczmy NWD 25 i 10

$$\text{NWD}(25, 10) = \text{NWD}(10, 5) \quad ?$$

$$\text{NWD}(10, 5) = \text{NWD}(5, 0) \quad 5$$

$$\text{NWD}(5, 0) = 5 \quad 5$$

ALGORYTM EUKLIDESA

Algorytm rekurencyjnie:

```
function NWD(a,b:integer):integer;  
begin  
if (b=0) then NWD:=a else NWD:=NWD(b,a mod b);  
End;
```

Wyznaczmy NWD 25 i 10

NWD = 5

$$\text{NWD}(25, 10) = \text{NWD}(10, 5)$$

5

$$\text{NWD}(10, 5) = \text{NWD}(5, 0)$$

5

$$\text{NWD}(5, 0) = 5$$

5