Temat: Rekurencja – wywołujemy samych siebie.

**Rekurencja** to odwołanie się np. funkcji do samej siebie. Aby wyznaczyć n-tą wartość trzeba najpierw wyznaczyć wszystkie „wcześniejsze" wartości. Aby algorytm rekurencyjny mógł się zatrzymać, jego kolejne odwołania do siebie samego muszą zależeć od pewnego warunku, który zmienia się z każdym kolejnym wywołaniem. Algorytm ten musi mieć ściśle zdefiniowany warunek zakończenia wywołań rekurencyjnych.
Każde wywołanie funkcji rekurencyjnej związane jest z zapamiętaniem kopii wszystkich zmiennych funkcji oraz z umieszczeniem na stosie adresu miejsca w programie do którego system ma powrócić po zakończeniu funkcji.

**1. Definicja rekurencyjna silni liczby naturalnej**
w zapisie matematycznym
n!=1 dla n=0
n!= (n-1)!*n dla n>0

w języku c++
```
double silnia  (int n)
{ if (n= =0)     return 1;
     else          return silnia(n-1)*n;
}
```

**2. Definicja rekurencyjna potęgi o wykładniku naturalnym liczby rzeczywistej**
w zapisie matematycznym
$a^0=1$ dla n=0
$a^n=a^{n-1}*a$ dla $n \in N_+$

w języku c++
```
float potega_rek(float x, int n)
{ if (n= = 0) return 1;
         else  return potega_rek(x,  n-1) *x;
}
```

**3. Definicja rekurencyjna ciągu Fibonacciego**
w zapisie matematycznym
$a_1=a_2=1$
$a_n=a_{n-1}+a_{n-2}$, n>2

w języku c++
```
int fibo (int n)
{  if (n ==1 || n ==2) return 1;
  else return fibo(n-1)+fibo(n-2);
}
```

**Zadanie**
Napisz program, który obliczy i wyświetli na ekranie 7 wyraz ciągu określonego rekurencyjnie w następujący sposób:

$$\begin{cases} a_1 = 0 \\ a_2 = 3 \\ a_n = 2 \cdot a_{n-1} - 3 \cdot n - a_{n-2} \quad , n > 2 \end{cases}$$

Plik nazwij: **ciag7.cpp.**