Fibonacciho posloupnost

Je posloupnost $\mathit{fib}_0, \mathit{fib}_1, \mathit{fib}_2, \ldots$ přirozených čísel definovaná

$$\mathit{fib}_0 = 0$$

$$\mathit{fib}_1 = 1$$

a pro každé $n > 1$: $\mathit{fib}_n = \mathit{fib}_{n-2} + \mathit{fib}_{n-1}$

Začátek Fibonacciho posloupnosti:

$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765, \ldots$

Fibonacciho posloupnost

Napište funkci `fib` vracející n-tý člen Fibonacciho posloupnosti. Zapište ji nejprve naivním způsobem, zopakováním rekurentní definice.

Kolikátý člen posloupnosti tato funkce spočítá do 30 s na vašem počítači?

Fibonacciho funkci můžeme zapsat rekurzivně, ale efektivněji. Vyjádříme ji pomocí funkce `f`, která má dva argumenty navíc: $\texttt{fib}(n) = \texttt{f}(0, 1, n)$.
V nich jsou vždy dva sousední členy posloupnosti.

```
typedef unsigned int ui;
typedef unsigned long int uli;

uli f (uli a, uli b, ui n) {
  // DOPLŇTE pomocnou funkci f
}

uli fib_r (ui n) { return f(0, 1, n); }
```

Vypište tabulku členů Fibonacciho posloupnosti od fib_0 do fib_{93}.

Poznámka: Fibonacciho posloupnost je rovna součtu dvou geometrických posloupností po členech. Kvocienty těchto dvou posloupností jsou

$\varphi = \frac{1+\sqrt{5}}{2} \approx 1{,}618$ a $\psi = \frac{1-\sqrt{5}}{2} \approx -0{,}618$.

$$
\begin{array}{rclllll}
\left(\frac{1}{\sqrt{5}}\ \varphi^n\right)_{n=0}^{\infty} & = & \frac{2\sqrt{5}}{10}, & \frac{5+\sqrt{5}}{10}, & \frac{5+3\sqrt{5}}{10}, & \frac{10+4\sqrt{5}}{10}, & \frac{15+7\sqrt{5}}{10}, \ \ldots \\
\left(-\frac{1}{\sqrt{5}}\ \psi^n\right)_{n=0}^{\infty} & = & \frac{-2\sqrt{5}}{10}, & \frac{5-\sqrt{5}}{10}, & \frac{5-3\sqrt{5}}{10}, & \frac{10-4\sqrt{5}}{10}, & \frac{15-7\sqrt{5}}{10}, \ \ldots \\
\hline
(fib_n)_{n=0}^{\infty} & = & 0, & 1, & 1, & 2, & 3, \ \ldots
\end{array}
$$

Všimněme si, že $|\varphi| > 1$, $|\psi| < 1$. Tedy první geometrická posloupnost rychle roste a druhá geometrická posloupnost rychle konverguje k nule. Fibonacciho posloupnost se proto asymptoticky (pro velká n) chová jako první geometrická posloupnost. Říkáme, že *roste exponenciálně*.

Napište funkci `fib` vracející n-tý člen Fibonacciho posloupnosti – jako číslo typu `double`. Použijte funkce `pow` a `round` z knihovny `math`.

Řazení výběrem

Algoritmus *Select sort* řadí konečnou posloupnost $a_0, a_1, \dots, a_{n-1}$ takto:

- Jednoprvková posloupnost je už seřazená a algoritmus na ní končí.
- Jinak se v posloupnosti najde největší prvek,
- vymění se s posledním prvkem
- a seřadí se posloupnost $a_0, a_1, \dots, a_{n-2}$.

Řazení výběrem

Máme definován typ záznamů

```
typedef struct {
    unsigned int rec_id;  // klíč pro řazení
    char rec_data;        // příklad dat v záznamu
} Rec;                    // typ záznamu
```

a chceme pole takovýchto záznamů seřadit vzestupně podle klíče `rec_id`.

Napište rekurzivní definici funkce

```
void selectSort(Rec a[], int n)
```

která v poli `a` seřadí prvky `a[0]`, . . . , `a[n-1]`.

Řazení výběrem

Kolik elementárních kroků, v závislosti na délce řazené posloupnosti, algoritmus řazení výběrem provede?

Je algoritmus stabilní?

Binární vyhledávání

Algoritmus binárního vyhledávání prvku s v rostoucí posloupnosti $a_l, \ldots, a_r$ využívá jejího uspořádání.

Když pro nějaký index m ležící mezi l a r je

$s < a_m$, pak se prvek s nemůže nalézat napravo od a_m,

$s > a_m$, pak se prvek s nemůže nalézat nalevo od a_m,

$s = a_m$, pak je prvek s nalezen (známe jeho index m)

Napište rekurzivní definici funkce

```
int binsearch (int a[], int l, int r, int s)
```

Jejím prvním parametrem je pole `a` celých čísel, která tvoří *rostoucí* posloupnost.

Funkce vyhledá číslo `s` v úseku `a[l]`,...,`a[r]` a vrátí jeho index.

Pokud se zde číslo `s` nevyskytuje, funkce vrátí -1.