FIKS 8. Ročník, 2. kolo, úloha 2

# Klece – řešení

## Postup:

Vstup si rozdělíme podle řádků na počet zvířat (*n*), samotná zvířata, která ještě jednotlivě uložíme do arraye rozdělením podle mezery (*zvirata*) a maximální rozdíl dravosti (*p*). Array *zvířata* je zároveň potřeba převést na *int* (popř. jinou číselnou proměnnou), protože dalším krokem je tento array seřadit* podle velikosti. Seřazený array potom vložíme do for loopu a projdeme každou jeho hodnotu. Vytvoříme si také proměnnou *rozdil,* které ještě před loopem dáme hodnotu *zvirata[0]*. V samotném loopu pak budeme porovnávat, zda je rozdíl hodnot iterovaného itemu z arraye *zvirata* a proměnné *rozdil* větší než maximální rozdíl dravosti *p*. V případě že ne, pokračujeme do další iterace. Pokud ale rozdíl větší bude, znamená to, že do této klece už další zvíře nedáme. Změníme tedy hodnotu proměnné *rozdil* na zvíře *zvirata[iterace+1]* a index této iterace zapíšeme do arraye, kterému můžeme říkat třeba *bodyZlomu*. Po konci loopu rozdělíme *zvirata* podle indexů zapsaných v *bodyZlomu* do výsledného arraye. Každý z jeho sub-arrayů bude jedna klec a v ní zapsaná budou zvířata podle dravosti.

***Algoritmus řazení popsán zvlášť pro přehlednost***

Řekněme, že pro řešení naší úlohy využijeme mergesort, ten funguje následovně:
Rozdělíme array na podseznamy, z nichž každý bude obsahovat jeden prvek. Opakovaně budeme porovnávat a seřazovat jednotlivé dílčí seznamy, dokud nám nezbyde jen jeden. To bude náš finální seřazený seznam.

## Čas a paměť:

Časovou náročnost tohoto řešení uvažujeme podle sorting algoritmu, který použijeme (v případě mergesortu je to O(n log(n)), zbytek běží v O(n). Pro paměťovou náročnost platí to samé (u mergesortu O(n)) **-> časová náročnost algoritmu bude O(n log(n)), paměťová náročnost O(n)**