TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Algoritmizace a programování

1. semestrální práce

Zadání č. 5

2014 / 15

Lukáš Jeníček

Obsah

1 Zadání

Zapište program, který do vzestupně setříděné posloupnosti kladných čísel bude zařazovat postupně další prvky a to pouze v případě, že se příslušný prvek ve vytvářené posloupnosti doposud nevyskytuje. Při vyhledávání pro vložení dalšího prvku využijte setříděnosti posloupnosti – aplikujte binární vyhledávání.

Specifikace vstupu:

Program má umožnit při jednom spuštění zpracování libovolného počtu zadání. Před zadáním další úlohy nechť program vypíše dotaz uživateli, zda pokračovat ve zpracování či nikoli – odpověď uživatele bude znak `'a'` nebo `'n'` (malými nebo velkými písmeny). Program má skončit v případě, že odpověď uživatele je `'n'`. Při načítání počáteční posloupnosti nechť program nejdříve načte počet a poté jednotlivé hodnoty posloupnosti. Poté má program načítat další čísla do zadání záporné hodnoty a tato čísla postupně zařazovat do posloupnosti.

Načtením záporného čísla zpracování aktuální úlohy končí, program má vypsat výslednou posloupnost

2 Vypracování

Analýza úlohy

Před samotnou realizací daného programu jsem se zamyslel nad jednotlivými funkcemi programu a začal vymýšlet algoritmy. Vzhledem k malému rozsahu úlohy jsem zvolil jednu hlavní metodu Main, ve které jsou použity další dvě metody s návratovým typem boolean, které slouží k ověření hodnot v podmínkách cyklů.

Vzhledem k zadání jsem si také musel nastudovat, jak funguje binární vyhledávání a jak jej v programu použít.

Postup a realizace

Po analýze zadání a vytvoření algoritmu binárního vyhledávání jsem se pustil do tvorby UI (User Interface = uživatelské rozhraní). Vzhledem k tomu, že uživatel mám mít možnost zadat

libovolné množství prvků, zdálo se mi vhodnější použít místo datového typu Array (pole) generickou kolekci (List). K zadání jsem tedy přistoupil z trochu jiného pohledu, ale myslím si, že je díky tomu program lépe čitelný a nemusím definovat velikost pole (což je problém, když nevím, kolik prvků uživatel zadá) – tímto se program stal „blbuvzdorným“ neboť vždy musíme počítat, že se uživatel bude snažit všelijakým způsobem funkčnost programu otestovat a právě zde by mohl narazit na problém (mohlo by se stát, že by při zadávání překročil definovanou velikost pole).

Ukázky kódu

```
public static boolean isEnd(String s)
{
    if (s.equals("a") || s.equals("A"))
        return true;
    else
        return false;
}
```

Zde máme metodu isEnd s návratovým typem boolean. Jedná se o jednu z metod, která je použita v podmínce konstrukce while a slouží k vyhodnocení vstupu od uživatele. Pokud uživatel zadá písmeno `'a'` nebo `'A'`, tak metoda vrátí TRUE a ukončí se celý program.

```
/**
 * Metoda vyhodnotí, jestli se zadané číslo nachází v poli.
 * Pokud ano, navrátí jeho index. Pokud se číslo v poli nenechází, vrací výjimku
 * @param a Vzestupně setřídně pole čísel
 * @param lowest Nejnižší index, od kterého začne vyhledávání
 * @param highest Nejvyšší index pole
 * @param number Hledané číslo
 * @return Index, na které se nachází hledané číslo
 */
public static int binarySearch(int[] a, int lowest, int highest, int number)
{
    if (lowest == highest && a[lowest] != number)
        throw new IllegalArgumentException("Nelze nalézt vhodné umístění (špatné vstupní hodnoty)");

    int middle = (highest - lowest) / 2 + lowest;

    if (a[middle] == number)
        return middle;
    else if (a[middle] < number)
        return binarySearch(a, middle + 1, highest, number);
    else if (a[middle] > number)
        return binarySearch(a, lowest, Math.max(lowest, middle - 1), number);
    else
        throw new IllegalArgumentException("Zadané číslo se v poli nenachází");
}
```

Metoda pro binární vyhledávání v mém programu nakonec není využita, ale i tak ji zde uvádím. Výstupem této metody je index, na kterém se zadané číslo nachází. Binární vyhledávání je

výkonný kód pro vyhledání daného prvku v poli, neboť využívá tzv. půlení intervalu (provede se vždy méně operací než při klasickém procházení pole pomocí konstrukce FOR).

3 Závěr

Je pravdou, že jsem původní zadání svým způsobem obešel. Ale algoritmus binárního vyhledávání jsem uvedl a mám pocit, že mé řešení je z pohledu ne začátečnického programátora optimálnější a pro ostatní, kteří do kódu budou nahlížet, přehlednější. Funkčnost zůstala zachování a díky tomu, že pracuji s generickou kolekcí, tak není zbytečně vytěžována paměť definováním velikosti pole (velikost totiž neznáme a museli bychom odhadovat zbytečně vysoká čísla).