Programování I (PRM044) – Cvičení 9

Obsah cvičení

- chyby v domácích úlohách
- řetězce
- optimalizace funkce počítající hodnotu sinu
- rekurze

Příklady

1. Napište proceduru `otoc(var s: string)`, která otočí zadaný řetězec.
2. Napište funkci `sinus(x: real): real`, která vrátí sinus parametru `x`. Hodnotu vypočítá jako aproximaci součtu řady $\frac{x^1}{1!} - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots$.
3. Napište rekurzivní funkci, která vypočítá faktoriál zadaného čísla.
4. Že nekonečná rekurze k ničemu není? Napište proceduru, která vypíše text písničky Pes jitrničku sežral.

Domácí úkol

1. Zjistěte, zda je zadaný řetězec palindrom (zní stejně, ať se čte normálně či pozpátku). Pro jednoduchost předpokládejte, že obsahuje pouze malá písmena a neobsahuje interpunkční znaménka. Příklady řetězců, kdy by měl program vypsat ANO: „zeman seno dones na mez“, „v elipse spi lev“, „kuna nese nanuk“, „jelenovi pivo nelej“. Tip: Nejprve si obsah řetězce znak po znaku překopírujte do pole, avšak bez mezer.
2. Napište funkci `podretezec(var podret, ret: string)`, která vrátí pozici prvního výskytu podřetězce `podret` v řetězci `ret`, při neúspěšném hledání vrátí hodnotu `0`.
3. Napište funkci, která spočítá počet výskytů zadaného podřetězce v zadaném řetězci.
4. Napište rekurzivní funkci, která vypočítá hodnotu n-tého Fibonacciho čísla.
5. Napište funkci, která dostane setříděnou posloupnost celých čísel `posl` reprezentovanou polem a číslo `cislo` a vrátí pořadí čísla `cislo` v posloupnosti `posl`. Funkce by měla mít logaritmickou časovou složitost. Tip: podívejte se na prostřední číslo posloupnosti a porovnejte jej s hledaným číslem, na základě výsledku porovnání pak hledání buď ukončete, nebo rekurzivně hledejte v pravé/levé polovině původní posloupnosti. Napište funkci tak, aby byla
   (a) rekurzivní,
   (b) nerekurzivní.
6. Napište rekurzivní proceduru, která pro dané n vypíše všechny možné permutace množiny čísel $\{1, \ldots, n\}$. Příklad: pro $n = 3$ vypíše: `(1, 2, 3); (1, 3, 2); (2, 1, 3); (2, 3, 1); (3, 1, 2); (3, 2, 1)`.
7. Napište funkci, které se předá den, měsíc a rok a ona vrátí pořadové číslo dne v roce. Příklad: pro `den` = 1, `mesic` = 3, `rok` = 2004 funkce vrátí 61 (jako 31 + 29 + 1 – rok 2004 byl přestupný).
8. Napište funkci, které předáte dvě proměnné typu `Datum` (definujte jako record) a ona vrátí počet dní, které uběhly mezi těmito dvěma daty.
9. Napište funkci, které předáte proměnnou typu `Datum` a ona vrátí den v týdnu, na který dané datum připadá.

Další příklady

1. Napište funkci, která dostane pole čísel a vrátí součet všech čísel na lichých pozicích.
2. Napište funkci, která dostane pole čísel a vrátí součet všech sudých čísel v poli obsažených.
3. Napište funkce, které dostanou pole čísel a vrátí
   (a) maximální prvek pole,
   (b) minimální prvek pole,
   (c) průměr všech prvků pole.
4. Napište proceduru, která otočí zadané pole.
5. Napište funkci, která ze vstupu přečte posloupnost čísel a vrátí, kolikrát se v posloupnosti vyskytla dvě čísla za sebou. Úlohu si vhodně dodefinujte (tj. jak má být posloupnost ukončená, zda pro `aaaa` vrátí 3 nebo 2,...).
6. Na vstupu je dána posloupnost celých čísel. Zjistěte, kolik různých čísel se v posloupnosti vyskytuje.
7. Napište proceduru, která vypíše všechna trojciferná čísla, ve kterých se žádná číslice neopakuje.
8. Napište rekurzivní proceduru, která pro zadané číslo uloží do zadaného pole rozklad tohoto čísla na prvočísla.
9. Napište program, který na výstup vypíše délky stran všech pravoúhlých trojúhelníků s celočíselnými stranami, jejichž obvod je menší než 10 000.
10. Napište funkci, která pro zadané n vrátí $\sum_{k=1}^{n} k^5$. Neodvozujte vzoreček pro součet takové řady, ale použijte cyklus.
11. Definujte datový typ `Matice2`, který bude reprezentovat matici, jejíž rozměry nebudou pevné (mohou tedy existovat dvě proměnné typu `Matice2` a každá z jimi reprezentovaných matic bude mít jiné rozměry). Napište proceduru `vypeckuj(var m: Matice2; var pecka: Matice2)`, která vrátí matici `pecka`, která vznikne z matice `m` odstraněním prvního a posledního řádku a prvního a posledního sloupce.
12. Na vstupu je posloupnost nezáporných celých čísel ukončená číslem -1 rozdělená čísly nula na úseky kladných čísel, pro jednoduchost můžeme předpokládat, že závěrečnou -1 bezprostředně předchází nula.
    (a) Najděte třetí největší číslo z druhého nejdelšího úseku. Vstupní posloupnost smíte číst jen jednou.
    (b) Vypište všechna čísla nejdelšího úseku. Vstupní posloupnost můžeme přečíst dvakrát.
13. Program přečte ze vstupu posloupnost znaků ve tvaru „a+b=“, kde a a b jsou celá kladná čísla. Poté vypíše hodnotu součtu $a + b$. Předpokládejme, že a a b se vejdou do typu `integer`. Program musí kontrolovat správnost tvaru zadané posloupnosti. Použijte stejnou myšlenku jako u Hornerova schématu.
14. Napište funkci `otoc`, která dostane číslo `n` typu `integer` a vrátí číslo (taktéž `integer`), které vznikne z `n` otočením cifer (tj. poslední cifra bude první, předposlední druhá, atd.). Př.: Dostane-li funkce číslo 8307, vrátí 7038.