Univerzita Karlova

Přírodovědecká fakulta

Studijní program: Aplikovaná geografie

Studijní obor: Fyzická geografie a geoinformatika

Technická dokumentace programu na Převod čísla na římské číslice a zpět

Předmět: Úvod do programování

Tomáš Seidler

Praha, 2026

Zadání

Převod čísla na římské číslice a zpět – Vstupní hodnotou je celé číslo v desítkové soustavě zadané uživatelem. Nalezněte jeho reprezentaci římskými číslicemi tak, aby respektovala pravidla použitá pro zápis v této nepoziční soustavě. Reprezentaci římskými číslicemi následně převeďte zpět do desítkové soustavy pro ověření.

Rozbor problému

Musíme se řídit podle pravidel pro tvorbu římských čísel. Pro převod arabských na římské můžeme od zadaného čísla postupně odečítat největší číslo (hodnota největšího možného římského symbolu), které se do něho vejde a podle odečtené hodnoty přidávat římský symbol. Tím, že jdeme od největšího možné hodnoty po nejmenší, splníme rovnou pravidlo psaní římských číslic od největší po nejmenší. Když přidáme jako symboly IV, IX a další, tak zajistíme že symboly se budou moci opakovat pouze třikrát, což splňuje pravidlo maximálně 3 symbolů vedle sebe a rovnou i odečítací pravidlo (pro odečítání můžeme použít pouze I, X, C). V případě převodu římských na arabské můžeme procházet římskou číslici z leva do prava (od největší po nejmenší) a postupně přičítat hodnotu daného římského symbolu. Pokud se bude jednat například o číslo IV nebo IX, vyřešíme to podmínkou, pokud je následující symbol větší, aktuální odečtem a až přijde řada na další (následující) symbol, tak se přičte celý přičte (IV: 0 - 1 = 1 –> -1 + 5 = 4). Tím splňujeme odečítací pravidlo. Pokud je římské číslo zapsané správně, neměl by nastat problém (vzhledem k tomu že přichází od jiné časti mého programu, tak bude správně zapsaný). Rozsah převodu na římská čísla bude od 1 do 3999, neboť způsoby vyjadřování vyšších čísel nebyly používány jednotně.

Zvolený algoritmus

Používáme dvě funkce, které po správném zadání vstupu voláme z hlavní části programu. První funkce *„arabic_to_roman“* používá cyklus *for*, který postupně zjišťuje kolikrát se arabská hodnota římských symbolů vejde do zadaného čísla (ve sestupně seřazeném slovníku). Druhá funkce *„roman_to_arabic“* používá také cyklus *for* pro postupné zjištění hodnot římských symbolů. Vstup je omezený na hodnotu mezi 1 - 4000.

Program

Program začíná dvěma funkcemi. První funkce *„arabic_to_roman“* si vytváří slovník *roman_values*. Ve slovníku se nachází arabské hodnoty a k nim jsou přiřazeny římské symboly. Tento slovník je seřazen sestupně (od 1000 - M do 1 - I), toto seřazení je zásadní pro správné fungování celé této funkce. Z tohoto slovníku si budeme postupně brát římské symboly a zkoušet, zda se vejdou do zadaného čísla. Následně je vytvořená proměnná *result* jako string, do této proměnné se budou ukládat římské symboly a bude výstupem této funkce.

```python
# Funkce pro převod arabských čísel na římské číslice
def arabic_to_roman(number):
    # Slovník ve sestupném pořadí (hodnota: římský symbol)
    roman_values = {
        1000: "M",
        900: "CM",
        500: "D",
        400: "CD",
        100: "C",
        90: "XC",
        50: "L",
        40: "XL",
        10: "X",
        9: "IX",
        5: "V",
        4: "IV",
        1: "I"
    }
    # Proměnná pro uložení výsledného římského čísla (string)
    result = ""
```

Obr.1: Screenshot 1. část 1. funkce

Pokračujeme cyklem *for*, který postupně prochází slovník. V každé iteraci si vezme arabskou hodnotu římského symbolu. Touto hodnotou celočíselně vydělí zadané číslo a vloží výsledek do proměnné *count*. Potom se do proměnné *result* vloží tolik konkrétního římského symbolu kolikrát se jeho hodnota vešla do zadaného čísla (pokud ani jednou, tak hodnota *count* je 0, takže žádný symbol není vložen). Následně se zadané číslo upraví tak, že se odečte hodnota konkrétního římského symbolu tolikrát, kolikrát byl přidán do *result*. Po dokončení cyklu *for* nám funkce vrátí proměnnou *result*, ve které se nachází římské číslo.

```python
    # Cyklus for procházející hodnoty v slovníku (který je seřazený sestupně)
    for value in roman_values:
        # Počet kolikrát se hodnota vejde do zadaného čísla
        count = number // value # "//" pro celočíselné dělení
        # Přidání odpovídajícího počtu římských symbolů do výsledné proměnné
        result += roman_values[value] * count
        # Odečtení hodnoty (přičteného římského symbolu) od zadaného čísla
        number -= value * count
    # Vrátí výsledné římské číslo jako string
    return result
```

Obr.2: Screenshot 2. část 1. funkce

Druhá funkce *„roman_to_arabic“* si také vytváří slovník *roman_values*. Ale v tomto slovníku se nenachází dvou charakterové římské symboly a nemusí být sestupně seřazen. Je vytvořena proměnná *total* jako integer sloužící pro ukládání hodnot římských symbolů a jako výstup této funkce. Dále je získána délka (počet charakterů v stringu) římského čísla jako proměnná *n*.

```
    # Slovník pro převod římských číslic na arabské hodnoty
    roman_values = {
        'I': 1,
        'V': 5,
        'X': 10,
        'L': 50,
        'C': 100,
        'D': 500,
        'M': 1000
    }
    # Proměnná pro uložení výsledného arabského čísla (integer)
    total = 0
    # Délka zadaného římského čísla
    n = len(roman)
```

Obr.3: Screenshot 1. část 2. funkce

Následně použijeme cyklus *for*, kterým projdeme každý charakter v zadaném stringu (římské číslici). Římská symboly jsou v římském čísle seřazené od největšího po nejmenší (zleva doprava), pokud nejde o odečítací případ (jako je 4, 9, ...). V každé iteraci si přiřadíme hodnotu římského symbolu do proměnné *current_value* a poté se ptáme, v případě že aktuální symbol není poslední symbol římské číslice, zda následující symbol nemá větší hodnotu než aktuální. V případě že hodnota aktuální symbolu je menší než hodnota následujícího symbolu, hodnotu aktuálního symbolu odečteme od proměnné *total*. Pokud hodnota následujícího symbolu je menší, tak hodnotu aktuálního symbolu přičteme k proměnné *total* a pokračujeme na další iteraci. Po dokončení cyklu *for* nám funkce vrátí proměnnou *total*, ve které se nachází arabská hodnota římské číslice.

```
    # Cyklus for procházející každý charakter v zadaném římském čísle
    for i in range(n):
        # Arabská hodnota pro aktuální římský symbol
        current_value = roman_values[roman[i]]
        # Kontrola zda aktuální symbol nemá menší hodnotu než následující
        if i < n - 1 and current_value < roman_values[roman[i + 1]]:
            total -= current_value
        else:
            # Jinak přičteme hodnotu římského symbolu k celkovému součtu
            total += current_value
    # Vrátí výsledné arabské číslo jako integer
    return total
```

Obr.4: Screenshot 2. část 2. funkce

Hlavní část programu používá nekonečný cyklus (*while True*). Tímto způsobem uživatel může zadávat své vstupy hned po sobě bez zbytečné práce navíc. Nachází se zde vstup, který umožňuje vložit číslo pro převod nebo „*konec*“ pro ukončení programu. Kontroluje se zde jestli uživatel zadal „konec“, zda vložil celé číslo v rozmezí 1 až 4000 a zda vůbec vložil platný vstup. Pokud uživatel zadá platný vstup, jsou zavolány funkce a jsou vráceny výsledky funkcí.

```python
# Hlavní část programu
print("Program pro převod arabských čísel na římské.")
while True:
    vstup = input("Zadejte arabské kladné celé číslo v rozsahu 1 až 4000 "
                  "pro převod na římské číslo nebo 'konec' pro ukončení programu (pište bez mezer): \n"
                  ).strip()
    # Kontrola zda uživatel zadal "konec" pro ukončení programu
    if vstup.lower() == 'konec':
        print("Konec programu.")
        break
    # Kontrola zda je vstupní hodnota číslo (kladné celé číslo)
    if vstup.isdigit():
        arabic_number = int(vstup)
        if not 1 <= arabic_number <= 4000: # Kontrola zda je zadané číslo v požadovaném rozsahu
            print("Zadejte kladné celé číslo v rozsahu 1 až 4000.")
            continue
        roman_number = arabic_to_roman(arabic_number)
        print(f"Římské číslo pro {arabic_number} je: {roman_number}")
        print(f"Převod zpět na arabské číslo: {roman_to_arabic(roman_number)}")
    else:
        # Pokud vstup není číslo, zobrazí chybovou zprávu
        print("Neplatný vstup. Zadejte kladné celé číslo v rozsahu 1 až 4000 nebo 'konec'.")
```

Obr.5: Screenshot hlavní části programu

Vstupní/výstupní data

Uživatel vloží bez mezer na jedné řádce arabské číslo, které chce převést do římských číslic. Číslo musí být v povoleném rozmezí 1 - 4000. Program mu vrátí hodnotu zadaného čísla v římských číslicí a pro ověření ji ještě převede zpět.

Závěr

Program by měl správně fungovat a vytvářet římské číslice podle pravidel pro tvorbu římských číslic. Jedna výjimka byla udělána u čísla 4000, které se zapisuje v tomto programu jako *MMMM*, což je podle pravidel špatně (i přesto to někdy lidé, takhle zapisují), ale z estetického důvodu jsem rozšířil rozmezí z 3999 na 4000. Celý program lze jednoduše rozšířit na vyšší římská čísla, přidáním římských symbolů s čárkou nad nimi do slovníků (např. $\overline{V} = 5000$, jeden ze způsobů zapisování vyšších čísel). Nebo se může rozšířit o možný převod římských číslic na arabské číslice. V takovém případě už máme funkci na převod a je jen potřeba vytvořit kontroly, které zaručí, že římské číslo je zapsáno správně podle pravidel.