Dokumentace k projektu pro předměty IUS & IZP

Maticové operace

Projekt č. 3

Autor: Jan Kaláb (xkalab00@stud.fit.vutbr.cz)

Obsah

Úvod

Úkolem bylo vytvořit program, který bude mezi sebou sčítat a násobit dvě matice zapsané v souborech. Dále měl program umět dvě další volitelné operace, můj program umí ověřit monotónnost matice a zjistit zda je matice submaticí.

V první kapitole se dočtete základní teorii o maticích. Druhá kapitola pojednává o načítání matic ze souboru. Další kapitoly obsahují informace o implementovaných operacích s maticemi. Závěrem dokument zmiňuje, použité zdroje a metriky kódu.

Matice

Česká Wikipedie o maticích říká:

Matice je v matematice schématické uspořádání matematických objektů – *prvků matice* (též *elementů matice*) – do m řádků a n sloupců. Hovoříme o matici typu $m \times n$.

Vodorovnou n-tici prvků $(a_{i1}, a_{i2}, \ldots, a_{in})$, kde $i = 1, 2, \ldots, m$, označujeme jako i-tý *řádek matice*. Svislou m-tici čísel $(a_{1j}, a_{2j}, \ldots, a_{mj})$, kde $j = 1, 2, \ldots, n$, označujeme jako j-tý *sloupec matice*.

Část matematiky, která využívá matice, je označována jako *maticový počet*.

Matice se často využívají pro vyjádření obecné rotace vektorů, transformace vektorů od jedné báze k bázi jiné, k výpočtu soustav lineárních rovnic, či k vyjádření operátorů v kvantové mechanice.

Prvky matice jsou označeny indexy udávajícími **řádek** a **sloupec**, v nichž se prvek nalézá. Např. a_{53} leží v pátém řádku a třetím sloupci.

Obdélníková matice velikosti 4 × 3, prvek matice a_{23} je 7:

$$\begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 7 \\ 4 & 9 & 2 \\ 6 & 1 & 5 \end{pmatrix}$$

Načítání matic

V zadání je napsáno, že matice budou uloženy v souborech ve formátu: počet řádků matice, počet sloupců matice, a následně prvky matice. Údaje budou odděleny bílými znaky. Takže například výše uvedená matice bude:

```
4 3
1 2 3
1 2 7
4 9 2
6 1 5
```

Při prvním pohledu na matici musí být každému programátorovi jasné, že se jedná o typické dvourozměrné pole. K tomuto poli navíc patří i údaje o rozměrech matice, takže nejlepší bude udělat si pro takovou matici strukturu. A pro snadnou manipulaci s maticemi ve funkcích programu z ní navíc udělat vlastní datový typ.

```
typedef struct tmatice {
  int radky, sloupce;
  int** matice;
} TMatice;
```

Když pak chceme nějakou TMatici vytvořit, přečteme nejdřív ze souboru její rozměry (první dvě čísla). Následně alokujeme ono dvourozměrné pole, a to tak, že nejdříve alokujeme řádky matice, a do nich pak postupně alokujeme prostor pro sloupce, takže v paměti jsou pak prvky matice uloženy za sebou. Jelikož budeme s maticemi provádět nejrůznější aritmetické operace, je vhodnější pro alokaci použít funkci `calloc`, které alokovanou paměť navíc vynuluje. Tudíž v případě, že má matice v souboru méně prvků, než by měla mít, jsou nezadané prvky doplněny nulami. Teprve po alokaci samotné matice do ní zapíšeme její ze souboru přečtené prvky.

V této vstupní části programu může vzniknout nejvíce chyb, především ze strany uživatele. Například může uvést nesprávný název souboru, může do souboru napsat neplatné údaje, například písmena, zápomé rozměry matice, a podobně. Program s takovými stavy počítá, a v případě že nastanou se samozřejmě postará o uvolnění paměti a vrátí nadřazené funkci matici o nulových rozměrech, což si nadřazená funkce odchytne a zabrání dalšímu provádění chybných výpočtů.

Sčítání matic

Sčítání matic je jedna ze základních operací s maticemi, a je také velmi jednoduchá.

Pro sčítání matic je potřeba, aby všechny sčítané matice měly stejné rozměry. Pokud je tato podmínka splněna, pak se vždy sečtou prvky na stejné pozici, a také se na stejnou pozici umístí do výsledné matice.

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} + \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix} = \begin{pmatrix} 6 & 8 \\ 10 & 12 \end{pmatrix}$$

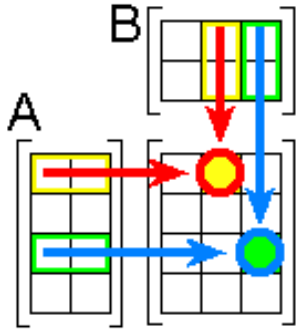
Násobení matic

Násobení matic už je trošku složitější, než běžné násobení čísel. V první řadě násobení matic není komutativní operace.

Řekněme tedy, že chceme provést $A_{mn} \times B_{op} = C_{mp}$. Povšimněte si, že výsledná matice bude mít stejný počet řádků jako matice A a stejný počet sloupců jako matice B. Podmínkou také je, aby $A_n = B_o$. Samotné násobení matic se pak provádí takto:

$$(A \times B)_{ij} = a_{i1} \cdot b_{1j} + a_{i2} \cdot b_{2j} + \ldots + a_{in} \cdot b_{nj}.$$

Lépe je to vidět na tomto obrázku:



Stejným způsobem je to provedeno i v programu. Nejdříve se ověří zda jde násobení vůbec provést. Pokud ano, tak se alokuje výsledná matice správných rozměrů, a do ní se vzájemně násobí a sčítají prvky zadaných matic.

Monotónnost matic

V zadání je uvedeno: „Matici nazveme monotónní, pokud jsou všechny její řádky a sloupce monotónní – to znamená neklesající nebo nerostoucí posloupnosti. Navíc musí platit, že všechny řádky jsou monotónní stejným směrem, stejně tak všechny sloupce musí být monotónní stejným směrem. Matice se všemi prvky stejnými je také monotónní maticí.“ Čili první matice monotónní je, druhá nikoliv:

$$\begin{pmatrix} 5 & 4 & 3 & 2 & 1 \\ 4 & 4 & 3 & 2 & 1 \\ 3 & 3 & 3 & 2 & 1 \\ 2 & 2 & 2 & 1 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 1 & 3 & 4 & 5 \\ 0 & 1 & 2 & 2 & 3 \\ 2 & 2 & 2 & 2 & 2 \\ 3 & 2 & 2 & 1 & 0 \end{pmatrix}$$

Řešení je celkem snadné. Zkontrolují se postupně sousedící dvojice prvků, a pokud se nemění směr posloupnosti, je řádek prohlášen za monotónní a zkontroluje se další. Po zkontrolování řádků se stejná činnost opakuje i se sloupci. Pokud je směr posloupnosti porušen byť i v jediné dvojici, pak je matice prohlášena za nemonotónní.

Submatice

Submatice je v podstatě výřez ze zadané matice. A úkolem je zjistit, zda-li je druhá zadaná matice submaticí první. Tedy například u této dvojice matic tomu tak je:

$$\begin{pmatrix} 1 & 7 & 3 & 4 \\ 4 & 9 & 5 & 7 \\ 1 & 2 & 6 & 7 \\ 5 & 6 & 9 & 7 \\ 4 & 4 & 7 & 1 \end{pmatrix} \begin{pmatrix} 9 & 5 & 7 \\ 2 & 6 & 7 \end{pmatrix}$$

Řešení není nijak náročné, i když trošičku chaotické. Jde o několik zanořených podmínek a cyklů. První dvojice cyklů postupně čte první matici, a hledá v ní první prvek druhé matice (na příkladu je to číslice 9). Pokud jej najde, vnoří se do druhého cyklu, který projíždí druhou matici a kontroluje, zda se shoduje s blokem první matice. Pokud najde rozdíl, vyskočí z cyklu, a dál prohledává první matici na výskyt prvku druhé matice. Pokud se však dostane až na poslední prvek druhé matice, prohlásí ji za submatici, a vrátí nadřazené funkci úspěch. Pokud se dostane první cyklus na konec první matice bez úspěchu, vrátí funkce neúspěch.

Drobnými úpravami lze docílit mírné optimalizace: V prvé řadě je dobré otestovat, je-li druhá matice menší než první, jinak nemůže být nikdy její submaticí. Dále je také zbytečné testovat ty prvky první matice, za které už by se druhá matice nevešla. Takže na příkladu se v prvním cyklu testuje jen 8 prvků.

Použité zdroje

- http://en.wikipedia.org/wiki/Matrix_multiplication
- http://cs.wikipedia.org/wiki/Matice

Metriky kódu

Zdrojový kód má 6181 bajtů na 245 řádcích. Po přeložení (`gcc -std=c99 -Wall -W -pedantic proj3.c -o proj3 -02`) na sytému Linux má binární soubor 9473 bajtů, z toho 304 bajtů statických dat.