Deviate cvičenie

Základné požiadavky:

- JEDEN súbor obsahujúci celý zdrojový kód, v jazyku C (ANSI C podľa prednášok), s názvom a v štruktúre podľa zverejnených inštrukcií (MSTeams)
- Programy musia komunikovať. Ak program očakáva vstup, musí oznamovať aký vstup sa očakáva. Ak vypisuje výsledok, musí vypisovať zrozumiteľný oznam (napr. čo za hodnotu to vypisuje).
- Formátovanie zdrojového kódu by malo zodpovedať približne príkladom z prednášok. Odsadzovanie textov je základ. Príklad dobrého a zlého formátovania sú v prednáške číslo dva na konci.

V zadaní použite funkciu z prednášky na tvorbu univerzálneho 2D poľa:

```
int **create(int riadky, int stlpce)
{
  int **p, i;
  p = (int **) malloc(riadky * sizeof(int *));
  for(i = 0; i < riadky; i++)
    p[i] = (int *) malloc(stlpce * sizeof(int));
  return p;
}
```

V zadaní je zakázané používať globálne premenné. Matica a aj jej rozmery budú definované vo funkcii MAIN a budú použité vo volaniach funkcií. Prenos matice, alebo jej rozmerov cez globálne premenné je nesplnením zadania.

Úloha prvá. Načítame maticu ľubovoľnej veľkosti.

Funkcia zo súboru načíta maticu. V súbore je všetko potrebné, rozmer aj obsah.
Povinný prototyp funkcie:

```
int **AlokujNapln(int *riadky, int *stlpce)
```

a) Načíta súbor s názvom vstup.txt, ak sa súbor nepodarí otvoriť, funkcia vráti NULL
b) Súbor má na prvom riadku 2 celé čísla, ktoré označujú počet RIADKOV (výšku) a počet STĹPCOV (šírku) tabuľky – načítajte tieto rozmery a použite ďalej
c) Dynamicky alokujte priestor pre tabuľku pomocou funkcie **create**
d) Súbor na ďalších riadkoch obsahuje obsah tabuľky
e) Predpokladajte, že v súbore je dosť čísel. Kto to vie urobiť, **nepovinne** môže urobiť, aby keď v súbore nie je dosť čísel (šírka x výška), funkcia vráti **NULL (**pomôcka: čítate riadky a stĺpce v cykloch po jednom čísle a keď fscanf nevráti 1, je zle – keďže pole je alokované – funkcia by v tejto vetve mala volať Uvolni() pred return NULL – bonus)
f) Funkcia v prípade úspechu vracia smerník na začiatok dvojrozmerného poľa a v parametroch **riadky** a **stlpce** aj načítané rozmery tabuľky.

Úloha druhá. Uvoľníme dynamicky alokovanú maticu ľubovoľnej veľkosti.

```
void Uvolni(int **pole, int riadky)
```

a) Uvoľní pamäť, ktorá bola pre maticu alokovaná
b) Pomôcka: uvoľňuje sa v opačnom poradí ako sa alokuje; uvedomte si, že Vám stačí len počet riadkov – uvoľňujú sa jednotlivé riadky a na koniec pole pointrov na riadky

Úloha tretia. Vypíšeme dynamicky alokovanú maticu ľubovoľnej veľkosti.

```
void Vypis(int **pole, int riadky, int stlpce)
```

formátovanie ako v cvičení číslo 8 (do riadkov a stĺpcov, 5 pozícií na číslo)

Úloha štvrtá. Nájdeme maximum v matici ľubovoľnej veľkosti.

Napíšte funkciu NajdiMax, ktorá nájde v tabuľke maximum a vráti vo výstupe jeho hodnotu a vráti aj súradnice na ktorých sa maximum našlo (číslované od nula – v parametroch odkazom **riadok, stlpec**). Ak viac ako jedna pozícia obsahuje najväčšie číslo, je nám jedno pozíciu ktorého vrátite.

```
int NajdiMax(int **pole, int riadky, int stlpce, int *riadok, *stlpec
```

Požadovaná forma main()

```
int main() {
    int **pole;
    int riadky, stlpce, max, max_riadok, max_stlpec;

    pole = AlokujNapln(&riadky, &stlpce);
    if (pole == NULL) {
      printf(„Pole sa nepodarilo naplnit“);
      return 1;
    }
    Vypis(pole, riadky, stlpce);
    max = NajdiMax(pole, riadky, stlpce, &max_riadok, &max_stlpec);
    printf(„Maximum pola [%d] sa naslo na suradnici [%d, %d]“,
      max, max_riadok, max_stlpec);
    Uvolni(pole, riadky);
    return 0;
}
```