Zadání zápočtové úlohy z předmětu KMI/JC

Úlohu odevzdejte nejpozději 9. 12. 2024 (do 23:59) přes MS Teams, případně zašlete na e-mail **marketa.trneckova@gmail.com** s předmětem „KMI/JC zapoctova uloha“. Úlohu po jejím schválení osobně předvedete vyučující (na hodině/domluvené konzultaci). Součástí prezentace bude modifikace kódu nebo doprogramování funkcionality dle vylosovaného zadání.

Naprogramujte modul (bez funkce `main()`) pro práci s „obrázky“ reprezentovanými jako dvourozměrná pole, jejichž prvky jsou celá čísla (typu `short`). Modul se bude skládat ze zdrojového kódu (soubor `obrazek.c`) a z hlavičkového souboru (soubor `obrazek.h`).

V modulu musí být definován strukturovaný datový typ `obrazek`. Struktura bude uchovávat informaci o velikosti obrázku: výška `int h` a šířka `int w`. Jednotlivé prvky obrázku (hodnoty pixelů) budou celá čísla a budou uložena v dvourozměrném poli `data` typu `short**`.

Dále modul obsahuje následující funkce:

- `obrazek inicializace(int h, int w)` – funkce vytvoří obrázek o velikosti $h \times w$, včetně alokování pole `data`.
- `obrazek cerny(int h, int w)` – funkce vytvoří obrázek o velikosti $h \times w$, kde všechny pixely budou mít hodnotu 0.
- `void odstran(obrazek obr)` – funkce dealokuje paměť alokovanou pro obrázek `obr`.
- `void zobraz(obrazek obr)` – funkce pro vykreslení obrázku do konzole. Všechny pixely s hodnotou 0 se vykreslí jako `' '` (znak mezera), pixely s hodnotou 1 se vykreslí jako `'.'` (tečka), pixely s hodnotou 2 se vykreslí jako `':'` (dvojtečka), pixely s hodnotou 3 se vykreslí jako `'+'` (plus) a pixely s hodnotou 4 se vykreslí jako `'#'` (hash).
- `obrazek otoc90(obrazek obr)` – funkce vrací obrázek, který vznikl otočením obrázku `obr` o 90 stupňů proti směru hodinových ručiček.
- `obrazek morfing(obrazek obr1, obrazek obr2)` – funkce vrací obrázek, který vznikl prolnutím obrázků `obr1` a `obr2`. Viz teorie na konci zadání.
- `short min(obrazek obr)` – funkce vrací nejmenší jasovou hodnotu (intenzitu) pixelu v obrázku `obr`.
- `short max(obrazek obr)` – funkce vrací největší intenzitu pixelu v obrázku `obr`.
- `obrazek jasova_operace(obrazek obr, operace o, ...)` – funkce vrací obrázek, který vznikl aplikací operace `o` na obrázek `obr`. Definujte výčtový typ `operace`, který může nabývat hodnot `NEGATIV`, `ZMENA_JASU` a `ZMENA_KONTRASTU`. Na základě operace bude přijímat 0, 1 nebo 2 nepovinné argumenty, představující parametry operace.

`NEGATIV`: nepřijímá žádný nepovinný argument.

`ZMENA_JASU`: přijímá jeden nepovinný argument typu `int`.

`ZMENA_KONTRASTU`: přijímá jeden nepovinný argument typu `double` představující konstantu pro změnu kontrastu a jeden nepovinný argument typu `int` upravující jas výsledného obrázku.

Jak operace pracují viz konec zadání.

- `obrazek nacti_ze_souboru(const char *soubor)` – funkce vrací obrázek, který je uložen v souboru `soubor`. Každý řádek souboru reprezentuje řádek obrázku a jednotlivé pixely jsou odděleny mezerou.
- `void uloz_do_souboru(obrazek obr, const char *soubor)` – funkce uloží obrázek `obr` do souboru `soubor` (ve stejném formátu, jako se obrázek vykresluje do konzole).

Dále modul obsahuje pomocné funkce pro práci s obrázky (abychom s nimi nemuseli pracovat přímo):

- `int vyska(obrazek obr)` – funkce vrací výšku obrázku (`h`).
- `int sirka(obrazek obr)` – funkce vrací šířku obrázku (`w`).
- `short prvek(obrazek obr, int i, int j)` – funkce vrací intenzitu pixelu obrázku `obr` na souřadnicích `i` a `j`.
- `void nastav_prvek(obrazek obr, int i, int j, short hodnota)` – funkce, která nastaví pixel na souřadnicích `i` a `j` obrázku `obr` na hodnotu `hodnota`.
- Globální proměnnou `chyba`, která bude uchovávat, zda v poslední operaci nedošlo k chybě. Tato proměnná bude nastavovaná v každé funkci.
- Definujte symbolické konstanty pro různé chyby:
  - `BEZ_CHYBY` – pokud ve funkci nedošlo k žádné chybě.
  - `CHYBA_ALOKACE` – chyba při alokaci paměti.
  - `CHYBA_OTEVRENI` – chyba při otevírání souboru.
  - `CHYBA_ZAVRENI` – chyba při zavírání souboru.
  - `CHYBA_TYPU` – chyba při provádění operací s nesprávně definovaným obrázkem / obrázky (např. přiřazení špatné hodnoty).
  - `CHYBA_JINA` – ostatní chyby.

Modul může obsahovat i další pomocné funkce.

Upozornění:

- V případě jakýchkoliv nejasností ohledně zadání, neváhejte mě kontaktovat. Ptejte se mě, ne spolužáků!
- Je nutno dodržet zadání! Včetně všech názvů funkcí, maker, struktury i položek struktury a výčtového typu.
- Opisování netoleruji! A to ani od umělé inteligence.

`https://www.zurnal.upol.cz/nc/zprava/clanek/univerzita-pripravila-doporuceni-k-vyuzivani-generativnich-modelu-ai/`

Je možné použít pouze knihovny: `stdlib.h`, `stdio.h`, `stdarg.h`

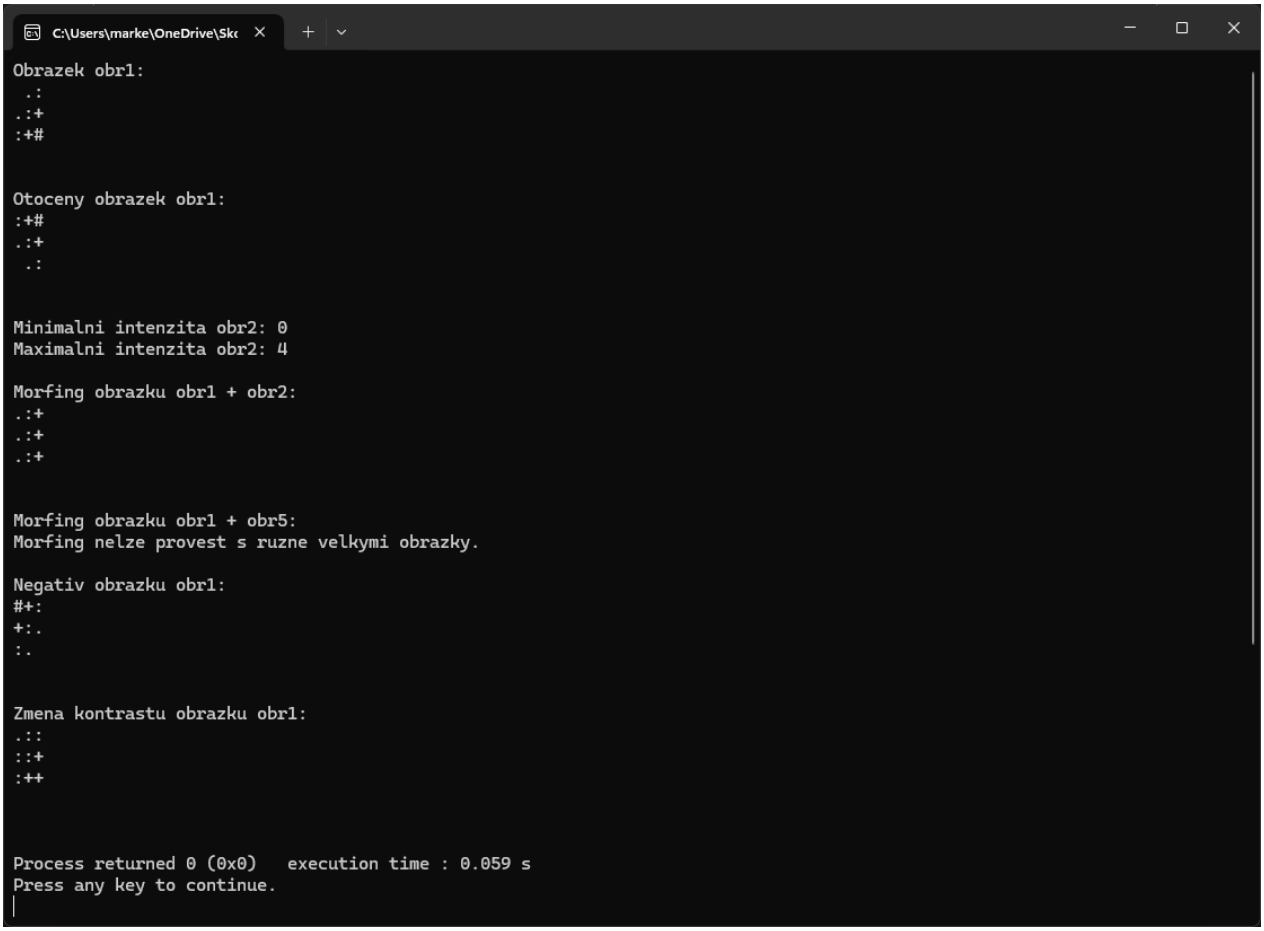
Příklad použití

Pro následující zdrojový kód

```
#include <stdio.h>
#include "obrazek.h"

int main(){
    obrazek obr1, obr2, obr3, obr4, obr5, obr6;

    obr1 = cerny(3,3);
    nastav_prvek(obr1, 0, 0, 0);
    nastav_prvek(obr1, 0, 1, 1);
    nastav_prvek(obr1, 0, 2, 2);
    nastav_prvek(obr1, 1, 0, 1);
    nastav_prvek(obr1, 1, 1, 2);
    nastav_prvek(obr1, 1, 2, 3);
    nastav_prvek(obr1, 2, 0, 2);
    nastav_prvek(obr1, 2, 1, 3);
    nastav_prvek(obr1, 2, 2, 4);

    printf("Obrazek obr1: \n");
    zobraz(obr1);
    printf("\n\n");

    obr2 = otoc90(obr1);
    printf("Otoceny obrazek obr1: \n");
    zobraz(obr2);
    printf("\n\n");

    printf("Minimalni intenzita obr2: %i \n", min(obr2));
    printf("Maximalni intenzita obr2: %i \n\n", max(obr2));

    obr3 = morfing(obr1, obr2);
    printf("Morfing obrazku obr1 + obr2: \n");
    switch(chyba){
        case CHYBA_TYPU:
            printf("Morfing nelze provest s ruzne velkymi obrazky.");
            break;
        case CHYBA_ALOKACE:
            printf("Nastala chyba alokace pameti.");
            break;
        case BEZ_CHYBY:
            zobraz(obr3);
            break;
    }
    printf("\n\n");

    obr4 = cerny(10,5);
    obr5 = morfing(obr1, obr5);
    printf("Morfing obrazku obr1 + obr5: \n");
    switch(chyba){
        case CHYBA_TYPU:
            printf("Morfing nelze provest s ruzne velkymi obrazky.");
            break;
        case CHYBA_ALOKACE:
            printf("Nastala chyba alokace pameti.");
            break;
        case BEZ_CHYBY:
            zobraz(obr5);
            break;
    }
    printf("\n\n");

    obr5 = jasova_operace(obr1, NEGATIV);
    printf("Negativ obrazku obr1: \n");
    zobraz(obr5);
    printf("\n\n");
```

```
    obr6 = jasova_operace(obr1, ZMENA_KONTRASTU, 0.5, 1);
    printf("Zmena kontrastu obrazku obr1: \n");
    zobraz(obr6);
    printf("\n\n");

    return 0;
}
```

dostaneme tento výstup



V kódu kvůli jeho délce chybí detekce chyb při vykonávání většiny operací.

Teorie

Obrázek můžeme chápat jako matici jasových hodnot (také jim říkáme *intenzity*). Matice je obdélníkové, nebo čtvercové schéma v našem případě celých čísel (my budeme uvažovat pouze jasové hodnoty 0, 1, 2, 3 a 4). Jednotlivé prvky matice nazýváme *pixely*. Obrázek s výškou h a šířkou w chápeme jako matici, která má h řádků a w sloupců. $a_{i,j}$ označuje prvek matice na i-tém řádku a j-tém sloupci.

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \dots & a_{m,n} \end{pmatrix}$$

Jasová hodnota 0 odpovídá černé barvě, hodnota 4 odpovídá bílé. Následující matice

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

odpovídá obrázku

Otočení obrázku o 90 stupňů proti směru hodinových ručiček vypadá následovně.

původní otočený

Máme dva obrázky A a B o velikosti $h \times w$. *Morfingem* (prolnutím) těchto obrázků nazýváme obrázek C o velikosti $h \times w$, pro který platí $c_{i,j} = (a_{i,j} + b_{i,j})/2$ pro všechna $i \in \{1, \dots, h\}$ a $j \in \{1, \dots, w\}$.

Negativem obrázku A o velikosti $h \times w$ nazýváme obrázek B o velikosti $h \times w$, pro který platí $b_{i,j} = 4 - a_{i,j}$ pro všechna $i \in \{1, \dots, m\}$ a $j \in \{1, \dots, n\}$.

Změna jasu obrázku A o velikosti $h \times w$ vrátí obrázek B o velikosti $h \times w$, pro který platí $b_{i,j} = a_{i,j} + k$ pro všechna $i \in \{1, \dots, m\}$ a $j \in \{1, \dots, n\}$, kde k představuje konstantu, o kterou měníme jasovou hodnotu (ta může být kladná, pak se jas zvyšuje – obrázek se zesvětlí, nebo záporná, pak se jak snižuje). Výsledné hodnoty menší než 0 se nastavují na 0, hodnoty větší než 4 se nastaví na 4.

Změna kontrastu obrázku A o velikosti $h \times w$ vrátí obrázek B o velikosti $h \times w$, pro který platí $b_{i,j} = k_1 \cdot a_{i,j} + k_2$ pro všechna $i \in \{1, \dots, m\}$ a $j \in \{1, \dots, n\}$, kde k_1 představuje kladnou konstantu, o kterou měníme kontrast (kontrastem rozumíme rozdíl mezi nejsvětlejším a nejtmavším pixelem). Pokud je tato konstanta větší než 1, pak kontrast zvyšujeme, pro hodnoty menší než 1 kontrast snižujeme. k_2 pak upravuje jasovou hodnotu výsledného obrázku, viz výše. Výsledné hodnoty zaokrouhlíme a opět hodnoty menší než 0 nastavíme na 0 a hodnoty větší než 4 nastavíme na 4.