**DU - zadání v1.00**

Napište program, který bude obsahovat funkce, které:

1. vytvoří a zruší dynamicky alokované dvourozměrné pole,
2. načte hodnoty do dynamicky alokovaného pole ze souboru,
3. vytiskne 2D pole,
4. provede jednoduchý úkon s 2D polem (vypustí první řádek)

Program napište tak, aby bylo možné měnit datový typ, které pole obsahuje (datový typ pole bude dán pomocí *typedef DATA\_TYPE*  nebo *#define DATA\_TYPE xxx*, kde pro účely kontrol a ladění bude *xxx* typ *double*. Příkazy *define/typedef* se umísťují do hlavičkového souboru).

**DU – návod k řešení**  
**Vytvoření základních funkcí:**

Vytvořte projekt, ve kterém budete mít tři soubory. Zdrojový soubor s *main* (*du2main.c*), zdrojový soubor funkcí *du2f.c* a k němu příslušný hlavičkový soubor *du2f.h*. Hlavičkový soubor příslušně ošetřete a vložte do zdrojových souborů.

Do souboru *du2main.c* nakopírujte následující kód:

int main(int argc, char \*argv[])

{

int ret = 0;

ret = DU(argc, argv); // funkce pro vytváření domácího úkolu

ret = Test(argc,argv); // funkce pro vytváření příkladů testu na cvičení

ret = Kontrola(argc,argv,1); // kontrolní funkce pro vytvořené funkce

return ret;

}

Do funkce *main* již další kód nepište. Funkce domácího úkolu vypracujte v souboru *du2f.c (du2f.h)* a jejich testování proveďte ve funkci *DU()*, která bude v souboru *du2main.c*. Přesný obsah funkce DU je uveden níže.

V souboru *du2main.c* vytvořte i (zatím prázdnou) funkci *Test()*, ve které budete testovat funkce z testu na cvičeních.

**Přidání testovacích souborů:**

Do aktuálního adresáře nakopírujte soubor *check.h, kontrola.h a DU2\_lib.lib* a testovací soubory *\*.txt*.

Do (překladu) projektu je nutné vložit knihovnu *DU2\_lib.lib*. To je možné přímo vložením do sekce „*Resource Files*“ projektu v *solution exploreru*. Druhou možností (složitější, ale u složitějších projektů vhodnější) je do parametru projektu *properties/linker/input/additionalDependencies* přidat jméno knihovny, to je *DU2\_lib.lib*. Dále pokud by knihovny byly v jiných adresářích, je nutné tyto adresáře nastavit v parametru projektu *properties/configurationProperties/vc++directories/LibraryDirectories*.  
Připojená knihovna začne „fungovat“ (půjde bez chyb přilinkovat k programu) až v okamžiku, kdy budete mít nadefinovány všechny požadované funkce (prozatím prázdné). Nyní si tedy přečtěte zadání a vytvořte základ budoucích funkcí – hlavičky funkcí s prázdnými tělíčky. Návratovou hodnotu volte 100, která bude znamenat, že funkce ještě není funkční – neobsahuje smysluplný kód. Změňte ji na 0 (v pořádku) teprve až si budete myslet, že je funkce v pořádku). Hlavičky funkcí naprogramujete na základě prvních odstavců v následujících bodech zadání.

Testovací soubor se snaží „napadat“ vaše funkce (včetně DU) a zjišťuje, zda se správně zachováte. Program je koncipován tak, že se snaží upozornit na co nejvíce chyb v jednom běhu, proto (ač by bylo asi správnější a rozumnější) po první chybě nekončí, ale snaží se pokračovat. To může vést k tomu, že program „spadne“ na některém z testů, který by se správně už provést neměl (kontrola proto není navržena tak, aby zvládla všechny situace, nebo aby dokonce reagovala na vaše útoky na ni). Testovací část používá pro kontrolu soubory, proto může dojít k chybě při operaci se soubory, i když s nimi (zatím) nepracujete.

Při tvorbě programu tvořte funkce tak aby byl splněn požadavek, že v poli bude vždy tolik (naalokovaných) řádků, kolik říká parametr rozměru y (pokud vznikne prázdný řádek, potom bude na jeho místo přesunut následující a bude zmenšen parametr pro počet řádků. To znamená, že není nutné při práci testovat přítomnost řádků na nenulovost jejich ukazatele).

**Tvorba funkcí:**

Doporučený postup tvorby je nejprve vytvořit hlavičky funkcí s prázdnými těly, následně vytvořit testovací sekvence na správnost vstupních parametrů a nakonec vlastní kód funkce.

1. Funkce pro získání a uvolnění paměti pro 2D pole jsou základní funkce programu. Funkce pro získání paměti *allocate2D* bude mít jako parametry rozměry pole, vracet bude typ chyby a v parametrech ukazatel na vytvořené pole (v případě neúspěchu vrátí hodnotu NULL), návratové hodnoty jsou 100 pro nenapsanou/neodladěnou funkci, 0 pro úspěch, 1 pro špatná vstupní data, 2 pro chybu při alokaci. *int allocate2D(int x, int y, DATA\_TYPE\*\*\* pole)*.   
   Parametr y udává počet řádků (první index pole, tj. „první alokace“). Parametr x udává počet prvků v řádku (druhý index pole, tj. „další alokace“).  
   Alokace se provádí funkcí *malloc* (knihovna *stdlib.h* (*malloc.h)*).  
   Prvky pole inicializujte nulovými hodnotami.
2. Funkce pro uvolnění paměti *release2D* dostane jako parametr ukazatel na pole a rozměr udávající počet řádků. Vrátí 0, pokud se povedlo, 1 pokud došlo k chybě. Ukazatel na pole nastaví na NULL. *int release2D( int y, DATA\_TYPE\*\*\* pole)*.  
   Pozn. funkce *EraseFirst* může zapříčinit, že nebude přítomen ani jeden řádek a přitom bude naalokován „sloupec“.
3. Funkce *Load2D* – funkce pro načtení dvourozměrného pole. Vstupem je otevřený soubor (FILE \*) a ukazatel na dynamické dvourozměrné pole (x, y jsou parametry výstupní). Při úspěchu vrací hodnotu nula. *int Load2D(FILE\* vstup, DATA\_TYPE \*\*\*pole, int \*x, int \*y);*.  
   Formát načítaných znaků bude:  
   - na prvním řádku rozměr x a y pole,  
   - na dalších řádcích hodnoty pro jednotlivé řádky oddělené čárkou (čárka i za poslední hodnotou).  
   Funkce dostane "prázdné" pole a naplní všechny parametry s ním spojené (tj. data a rozměry). „Prázdné“ pole je takové, že z jeho hodnoty je poznat, že nemá naalokovánu žádnou hodnotu.  
   Při špatných vstupních parametrech vrátí hodnotu: pro chybný/neotevřený soubor 1, pro chybné hodnoty spojené s parametrem pole 2, pro chybné hodnoty odkazů na rozměry 3. chyba dat při načítání 4.  
   Soubor bude otevřen v textovém modu pro čtení.
4. Funkce *Print2D* – funkce pro tisk 2D pole. *int Print2D(FILE\* fw,DATA\_TYPE \*\*pole, int x, int y)*. „Prázdné“ pole se netiskne a vrací se kód 1, jinak 0.  
   Parametr fw určuje, kam se bude tisknout (při testování použijte pro tisk na konzolu proměnnou stdout). Formát tisku bude stejný jako ve funkci *Load2D* (za posledním řádkem se odřádkuje a nebude následovat již žádný znak).
5. Funkce *EraseFirst* – uvolní (odalokuje) první řádek pole, ostatní posune (odebere první řádek a následně na jeho místo přesune druhý, na místo druhého třetí …, na „poslední“ se dá NULL (=přealokace pole na kratší se neprovádí). V případě, že je první řádek úspěšně uvolněn, vrací hodnotu nula. (Jinak) Při chybných vstupních parametrech vrací hodnotu 1. Vyberte nejvhodnější prototyp funkce (Funkce bude mít pouze parametry (=co nejméně parametrů), které dostačují k požadované činnosti. Jak u rozměru, tak u „pole“ dat probíhá manipulace pouze s řádky, změny se musí projevit vně funkce (= musí se změnit y rozměr matice i vně funkce)):   
   *int EraseFirst(DATA\_TYPE \*\*pole, int x, int y)  
   int EraseFirst(DATA\_TYPE \*\*pole, int x)  
   int EraseFirst(DATA\_TYPE \*\*pole, int y)  
   int EraseFirst(DATA\_TYPE \*\*\*pole, int x, int y)  
   int EraseFirst(DATA\_TYPE \*\*\*pole, int x)  
   int EraseFirst(DATA\_TYPE \*\*\*pole, int y)  
   int EraseFirst(DATA\_TYPE \*\*pole, int \*x, int \*y)  
   int EraseFirst(DATA\_TYPE \*\*pole, int \*x)  
   int EraseFirst(DATA\_TYPE \*\*pole, int \*y)*

**Vzorové použití funkcí:**  
Demonstrujte napsané funkce v testovací funkci (DU), která načte data ze souboru,

který je uveden jako první parametr uvedený na příkazovém řádku (předán do funkce DU).

Vytvořte druhé pole stejného rozměru, do něhož překopírujete data.

Z tohoto pole vypusťte první dva řádky a výsledek vytiskněte do souboru, jehož název je udán jako další

parametr na příkazovém řádku.

Pokud nejsou uvedeny parametry (názvy souborů), vraťte chybový kód 1.

Pokud se nepodaří jeden z těchto souborů otevřít, vraťte chybový kód 2.

Pokud dojde k (jakékoli) chybě v jakékoli funkci, vraťte chybový kód 3.

Ve funkci nedělejte žádnou další manipulaci s pamětí (která nevyplývá ze zadání. Dělejte pouze a jen to co by ve funkci DU mělo být).