BPC-UDP Cvičení 12

# Příklad 1

Příklad je zaměřen na zpracování NTMBS (null-terminated multibyte string) řetězců znaků kódovaných v UTF-8 a práci s locale.

1. Vytvořte projekt, přidejte do něj soubor main.c.  
   Soubor uložte v kódování UTF-8: File > Save main.c as… > šipka na tlačítku Save > Save with Encoding… > povolit soubor přepsat > Encoding > Unicode (UTF-8 without signature) Codepage 65001 > OK.
2. Nastavte argumenty příkazového řádku (Command Arguments) na: 6,28 6.28 .
3. Do souboru main.c vložte hlavičkové soubory stdio.h, locale.h a uchar.h.
4. Ve funkci main():
   1. Pomocí funkce setlocale() vytiskněte nastavení LC\_ALL po spuštění programu.
   2. Pomocí funkce setlocale() nastavte LC\_ALL na hodnotu  
      cs\_CZ.UTF-8 a pro kontrolu vytiskněte nově nastavenou hodnotu.
   3. Definujte proměnnou text typu ukazatel na char a inicializujte ji tak, aby ukazovala na utf8 řetězec "České znaky i různé symboly ‰ ✉ ✆ ☺ ☕.". (Lze zadat i jako "České znaky i různé symboly \u2030 \u2709 \u2706 \u263A \u2615.")
   4. Vytiskněte číslo 3.14 (jako double).
   5. Pomocí funkce sscanf() převeďte argv[1] na číslo typu double a číslo vytiskněte na standardní výstup.
   6. Pomocí funkce setlocale() nastavte LC\_NUMERIC na hodnotu en\_US.UTF-8 a pro kontrolu vytiskněte nově nastavenou hodnotu.
   7. Vytiskněte číslo 3.14 (jako double).
   8. Pomocí funkce sscanf() převeďte argv[2] na číslo typu double a číslo vytiskněte na standardní výstup.
   9. Zjistěte, co se stane, pokud v bodě e použijete argv[2] a argv[1]? Chování programu zdůvodněte.

Poznámka: V konzole si nastavte font „NSimSun“. Po dokončení tohoto příkladu font vraťte na „Consolas“!!!

# Příklad 2

Upravte program z příkladu 3 cvičení 11 tak, aby jednotlivé operace s daty byly prováděny v samostatných funkcích.

1. Napište definici funkce data\_read(), která má úplný funkční prototyp:  
   int data\_read(FILE \*aFile, struct TData \*aPtrData);  
   Funkce přečte z datového streamu aFile vstupní data. Pokud při provádění funkce dojde k chybě, funkci ukončete s návratovou hodnotu -1. Jinak funkce vrací hodnotu 0.
   1. Pokud má ukazatel aFile nebo ukazatel aPtrData hodnotu NULL, ukončete funkci s návratovou hodnotou -1.
   2. Přečtěte ze streamu, na který ukazuje aFile, počet dat, dynamicky alokujte paměť pro uložení dat, přečtěte data a uložte je do dynamicky alokovaného pole.  
      Pokud při některé operaci dojde k chybě, funkci ukončete s návratovou hodnotou -1. Nezapomeňte předtím uvolnit dynamicky alokovanou paměť (pokud byla alokována).
   3. Počet dat a ukazatel na dynamicky alokované pole zapište do proměnné, na kterou ukazuje ukazatel aPtrData. Funkci ukončete s návratovou hodnotou 0.
2. Napište definici funkce data\_write(), která má úplný funkční prototyp:  
   int data\_write(FILE \*aFile, const struct TData aData);  
   Funkce zapíše do datového streamu aFile elementy pole, na které ukazuje složka iPtr parametru aData.  
   Pokud nedojde k chybě, funkce vrací hodnotu 0. Pokud má ukazatel aFile nebo iPtr hodnotu NULL nebo dojde při zápisu k chybě, funkce vrací hodnotu -1. Pokud nedojde k chybě funkce vrací hodnotu 0.
3. Napište definici funkce data\_minmax(), která má úplný funkční prototyp:  
   int data\_minmax(const struct TData aData, double \*aPtrMin, double \*aPtrMax);  
   Funkce nalezne nejmenší a největší číslo z pole, na které ukazuje složka iPtr parametru aData.
   1. Pokud má některý z ukazatelů iPtrMin, iPtrMax nebo iPtr (složka aData) hodnotu NULL, funkci ukončete s návratovou hodnotou -1.
   2. Stanovte nejmenší a největší hodnotu čísel v poli, na které ukazuje složka iPtr parametru aData.
   3. Největší a nejmenší hodnotu zapište do proměnných, na které ukazují ukazatele aPtrMin a aPtrMax. Funkci ukončete s návratovou hodnotou 0.
4. Napište definici funkce data\_minmax\_write(), která má úplný funkční prototyp:  
   int data\_minmax\_write(FILE \*aFile, double aMin, double aMax);  
   Funkce zapíše do streamu, na který ukazuje ukazatel aFile, hodnoty parametrů aMin a aMax s doprovodným textem "*min=*" "*max=*".  
   Pokud má parametr aFile hodnotu NULL nebo dojde při zápisu k chybě, funkci ukončete s návratovou hodnotou -1. Pokud nedojde k chybě, funkce vrací hodnotu 0.
5. Napište definici funkce data\_destroy(), která má úplný funkční prototyp:  
   void data\_destroy(struct TData \*aPtrData);
   1. Má-li parametr aPtrData hodnotu NULL, funkci ukončete.
   2. Uvolněte dynamicky alokovanou paměť, na kterou ukazuje složka iPtr parametru aPtrData.
   3. Do složky iPtr zapište hodnotu NULL, do složky iNum hodnotu 0.
6. Použijte výše popsané funkce ve funkci main(), tak, abyste zajistili stejnou funkčnost programu, jakou měl program z příkladu 3 cvičení 11.

# Příklad 3

Vytvořte nový projekt, přidejte do něj soubory main.c, TData.h, TData.c, check.h, check.c a vstup.txt. Soubory main.c a vstup.txt převezměte z předchozího příkladu.

1. Přemístěte definici struktury TData a úplné funkční prototypy funkcí data\_read(), data\_write(), data\_minmax(), data\_minmax\_write(), data\_destroy() do hlavičkového souboru TData.h.
2. Hlavičkový soubor ošetřete proti vícenásobnému vložení.
3. Přemístěte definice funkcí data\_read(), data\_write(), data\_minmax(), data\_minmax\_write(), data\_destroy() do souboru TData.c.
4. Program přeložte a ověřte jeho funkčnost.