**Přípravný (domácí) úkol pro druhý test BPC2A (2016)**

**Motivace:**

Příklad je věnován procvičování alokací a práci s ukazateli (obsahuje proto části kódu, které lze implementovat lépe a efektivněji). Pro práci se správcem pamětí (získání=alokace; vrácení = dealokace) používejte pouze funkce: malloc a free.

**Stručný popis úkolu:**

Napište program, který pomocí funkce načítá řádky ze vstupního souboru a po zpracování uloží výsledky do výstupního souboru. Zpracování dat spočívá ve zdvojení mezer a v převodu znaků na čísla (jejich ASCII hodnoty).

**Popis zpracování úkolu:**

1. Napište funkci PocetMezer, která vrátí celkový počet znaků mezera v řetězci předaném jako parametr. Prototyp funkce je: unsigned PocetMezer(char \*aRadek). Pokud parametr aRadek má hodnotu NULL, funkce vrací hodnotu 0.
2. Napište funkci pro načtení řádku ASCII znaků ze souboru (výsledkem bude C řetězec). Řádek jsou znaky ukončené znakem konec řádku. Prototyp funkce int NactiRadek(char \*\*aRadek, FILE\* aFile);

Pokud budou předané ukazatele (aRadek nebo aFile) ukazovat na neplatnou adresu (tj. budou obsahovat NULL) načtení se neprovede a funkce vrátí příslušný chybový kód (viz seznam chybových kódů níže).

Pokud bude ukazatel (aRadek) platný, ale místo, na které ukazuje, neobsahuje NULL (tj. již obsahuje alokovanou paměť) načtení se neprovede a vrátí se příslušný chybový kód (viz seznam chybových kódů níže).

Funkce bude realizovat načtení řádku následujícím způsobem:

- funkce naalokuje pole pro 10 znaků,

- Zahájí se čtení jednotlivých znaků řádku. Pokud bude načítaný řádek delší než je velikost naalokovaného pole (tj. nebude možné do pole uložit načtený znak, včetně rezervace místa pro uložení koncového znaku), naalokuje se nové pole dvojnásobné délky, zkopírují se znaky z původního pole do nově alokovaného, a původní pole se odalokuje. Další znaky řádku se načítají (přídávají za již načtené) do nově alokovaného pole. Pokud opět dojde k překročení velikosti pole, proces alokace nového pole a odalokace původního se opakuje.

- Po načtení celého řádku dojde k jeho expandování pomocí následujícího pravidla: každá mezera ve vstupním řetězci bude ve výstupním řetězci zdvojena.

Expandovaný řádek bude uložen do nově naalokované paměti s délkou odpovídající přesně délce tohoto C řetězce (počet znaků řádku + počet mezer (pomocí funkce PocetMezer) + ukončovací znak). Odalokujte paměť pro neexpandovaný řádek. Expandovaný řádek vraťte pomocí parametru aRadek a skutečnou návratovou hodnotou funkce bude délka expandovaného řádku (bez ukončovacího znaku).

|  |  |  |
| --- | --- | --- |
| vstup | “Ahoj“ | “Ahoj,␣jak␣se␣mas?“ |
| výstup | “Ahoj“ | “Ahoj,␣␣jak␣␣se␣␣mas?“ |

- ošetřete zmíněné chybové stavy  
- při chybě odalokujte veškerou funkci naalokovanou paměť a vraťte chybový návratový kód  
- pokud nedojde k chybě funkce, odalokuje veškerou funkci naalokovanou paměť kromě paměti odpovídající navrácenému řádku.

- návratová hodnota funkce bude:  
nezáporné číslo - délka načteného řetězce (bez ukončovacího znaku)  
(testy na chyby provádějte v níže uvedeném pořadí   
-1 (kterýkoli) předaný ukazatel nemá platnou adresu (tj. obsahuje hodnotu NULL)  
-2 na adrese odkazované ukazatelem pro uložení výsledku jsou již data (je tam platná adresa)  
-3 chyba při alokaci paměti (nedostatek paměti)  
-4 chyba při čtení znaků ze souboru

==============

Napište funkci int PrevedPole(char \*aRadek, unsigned int \*\*aVystup), která naalokuje novou paměť pro pole unsigned int stejné délky jako vstupní řetězec (alokované pole vloží na adresu aVystup). Do pole dosažitelného přes aVystup vloží ASCII hodnoty znaků ze vstupního pole aRadek.

V případě, že parametr aRadek obsahuje prázdný řetězec (tj. pouze znak konce řádku) funkce PrevedPole nebude alokovat žádnou paměť pro výstupní pole a vrátí návratovou hodnotu 0.

Návratovou hodnotou funkce bude:

nezáporné číslo - délka naalokovaného pole aVystup  
-1 (kterýkoli) předaný ukazatel nemá platnou adresu (tj. obsahuje hodnotu NULL)  
-2 na adrese odkazované ukazatelem aVystup určeném pro uložení výsledku jsou již alokována data (aVystup ≠ NULL)  
-3 chyba při alokaci paměti (nedostatek paměti)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Vstup (znaky) | A | b | ␣ | ␣ | 1 | 2 | \0 |
| Výstup (hodnoty) | 65 | 98 | 32 | 32 | 49 | 50 |

==============

Napište funkci test(int argc, char \*argv[]), (která bude mít stejné parametry jako funkce main). Funkci test budou předány stejné hodnoty parametrů jako má funkce main. Příkazový řádek (předaný prostřednictvím parametrů funkce main) bude obsahovat názvy vstupního a výstupního souboru.

Funkce test nejprve provede následující kontroly:

- zkontroluje počet parametrů předaných z příkazové řádky (tj. název vstupního a výstupního souboru) - při chybě návratový kód: NK = -5  
- otevřete soubor pro čtení - při chybě vraťte NK = -6  
- otevřete soubor pro zápis - při chybě vraťte NK = -7

Následně bude funkce test pro všechny řádky vstupního souboru provádět tuto činnost:

- načtěte řádek pomocí funkce NactiRadek. Pokud funkce vrátí chybu, vrátí i funkce test NK=shodný s vrácenou hodnotou funkce NactiRadek

- pomocí funkce (PrevedPole); převeďte načtený řádek na pole unsigned int reprezentující ASCII hodnoty znaků. Pokud dojde ve funkci k chybě, vraťte kód NK=návratový kód volané funkce (PrevedPole)

- převedené pole typu unsigned int uložte do výstupního souboru ve formátu čísel následovaných čárkami (na konci tisku řádku jednou odřádkujte)

Pro příklad výše uloží: 65,98,32,32,49,50,’\n’

Zajistěte, aby se program choval korektně vůči (alokované) paměti a (otevřeným) souborům (tj. odalokujte všechnu dynamicky získanou paměť a uzavřete všechny soubory - pro kontrolu správnosti použijte knihovnu [check](http://www.uamt.feec.vutbr.cz/~petyovsky/projects/checker/checker.zip)).