© 2010 damig

IZP - Projekt č. 1 - Jednoduchá komprese textu

Než začnete programovat, přečtěte si pozorně celé zadání.
Poslední změna: 19. října 2010

Motivace

Cílem této úlohy je naučit se nebo si zopakovat

- vytváření a používání podprogramů,
- zpracování standardního vstupu po znacích,
- zpracování parametrů programu zadávaných z příkazové řádky,
- správně převádět hodnoty z textového řetězce na číselnou hodnotu (v tomto případě pomocí standardních knihovních funkcí),
- správné ošetřování chybových stavů programu,
- vytváření přehledného zdrojového kódu.

Mějte na paměti, že cílem není naučit se komprimovat data. Existují mnohem účinnější komprimační algoritmy. Máte předvést, že jste pochopili základy algoritmizace a konstrukce programů v jazyce C.

Zadání úlohy

Vytvořte program který bude dále popsaným postupem komprimovat nebo dekomprimovat text podle parametrů zadaných z příkazového řádku při spouštění. Jednotlivé požadované parametry jsou popsány dále.

Specifikace vstupních a výstupních dat

Nekomprimovaný vstupní text bude tvořen tisknutelnými a interpunkčními (mezery, konce řádků, atd.) znaky bez diakritiky kromě číslic. K jejich identifikaci můžete použít makra z `ctype.h`. Komprimovaný text bude navíc obsahovat i číslice, které zde budou mít svou specifickou roli. Vyskytne-li se na vstupu v danou chvíli nelegální (ve smyslu neočekávaný) znak, považujte tuto situaci za chybný vstup.

Vstupní text čtěte ze standardního vstupu (stdin), výsledek zapisujte na standardní výstup (stdout). Chybová hlášení zapisujte na standardní chybový výstup (stderr).

Program musí být schopen zpracovat vstupní text libovolné délky (tj. délky 0 až nekonečno) a nesmí mít v tomto smyslu zabudováno žádné omezení. Program musí ohlídat, aby číselné parametry příkazového řádku byly v rozsahu číselného typu `unsigned int`. Pokud budou mimo tento rozsah, ohlaste chybu.

Standardní vstup zpracovávejte po znacích. Pro čtení ze standardního vstupu smíte použít výhradně funkce getchar(), getc() nebo fgetc(). Pro zpracování dat ze standardního vstupu nelze používat žádné funkce ze standardní knihovny pro analýzu textových řetězců (scanf, sscanf a podobně). Pro tisk výsledku na standardní výstup smíte použít výhradně funkce putchar(), putc() nebo fputc() (to neplatí pro tisk nápovědy a chybových hlášení).

Komprimační a dekomprimační algoritmus

Komprimace i dekomprimace bude řízena celočíselným parametrem N, jehož hodnota bude větší nebo rovna 1. Při komprimaci bude program hledat ve vstupním textu opakující se bloky znaků délky N. Dále program při nalezení minimálně 2 a maximálně 9 shodných bloků délky N za sebou zapíše tuto posloupnost na výstup jako číslo udávající počet opakování následované tímto blokem písmen délky N.

Zde je příklad vstupního textu:

```
Neustale zvatlal: "Uiiiiiii, blablabla" a "blebleble" a "hophop hophop", kdyz skakal po jedne noze.
```

Po komprimaci s parametrem N=3, bude výsledek vypadat takto:

```
Neustale zvatlal: "U2iiii, 3bla" a "3ble" a "2hop 2hop", kdyz skakal po jedne noze.
```

Navíc při komprimaci program zpracovává vstup zleva doprava, přičemž hledá *maximální počet* opakujících se bloků, než zpracuje zbytek řetězce. Tedy vstup:

```
Uii, blablabla bleble.
```

bude po komprimaci N=3 mít výsledek:

```
Uii, 3bla 2ble.
```

a nikoliv:

```
Uii, b2labla 2ble.
```

Při dekomprimaci musí být zadán parametr N stejný, jako při komprimaci. Algoritmus pak bude opisovat vstupní znaky na výstup a když narazí na číslici, následujících N znaků zopakuje na výstupu tolikrát, kolik udává přečtená číslice.

Poznámky

Všimněte si, že pokud bude N=1, bude algoritmus komprimovat pouze posloupnosti stejných znaků. Počet stejných bloků, které algoritmus zpracovává je omezen na 9 pro snazší dekomprimaci (aby se tam vyskytovala pouze jednociferná čísla). Toto omezení můžete ignorovat až při implementaci rozšíření svého projektu.

Rozhraní programu

Program se bude spouštět s parametry s následujícím významem. Není nutné, aby program uměl zpracovat parametry v jiném než uvedeném pořadí nebo je uměl kombinovat.

Příkazový řádek	Očekávaná činnost	Poznámka
proj1 -h	vytiskne nápovědu	Povinně
proj1 -c N	Program bude komprimovat vstupní text po blocích délky N. Hodnota parametru N musí být větší než 0 a musí se vejít do rozsahu datového typu `unsigned int`.	Zvládne-li pracovat s N=1, cca 30% funkčnosti. Zvládne-li pracovat i s N>1, cca 70% funkčnosti.
proj1 -d N	Provede dekompresi vstupního textu. Předpokládá, že vstupní text byl komprimován se stejnou hodnotou N. Hodnota parametru N musí být větší než 0 a musí se vejít do rozsahu datového typu `unsigned int`.	Povinně, cca 30% funkčnosti
proj1 --extra	funkce podle vašich vlastních rozšíření	Nepovinně, za prémii 1 bod

Příklad: Po spuštění programu `proj1[.exe]` s parametry -c a 1

```
$ ./proj1 -c 1
```

program zpracovává text ze standardního vstupu. Například toto

```
aaaaa bb
ccccccccccc      dddd
```

Výsledek, který program vytiskne pak vypadá takto. Všimněte si, jak vypadá velmi dlouhá posloupnost bloků po komprimaci:

```
5a 2b
9c2c6 4d
```

Totéž s přesměrováním souborů (vstupní a výstupní soubor si stáhněte na svůj počítač a teprve pak si je prohlédněte, webový prohlížeč může při prohlížení změnit jejich formátování):

```
$ ./proj1 -c 1 < in.txt > out.txt
```

Po zadání prázdného vstupu, program vygeneruje prázdný výstup.

Maximální počet získaných bodů

Maximální počet bodů za tuto úlohu je **5 bodů**. Součástí řešení není dokumentace.

Prémie

Za úlohu lze získat jeden prémiový bod, pokud si zadání rozšíříte nějakým netriviálním způsobem, a pokud dodržíte výše zmíněné podmínky. V tomto případě je nutná dokumentace (stačí textový soubor - je to nutné zejména proto, aby si asistent rozšíření vůbec všiml). Některá netriviální rozšíření:

- zpracování většího počtu shodných bloků než 9,
- vylepšení komprimačního algoritmu (musíte tomu však rozumět, nelze pouze opsat nějaký známý komprimační algoritmus z webu),
- ...(poraďte se se cvičícím)

Co se zejména hodnotí

Tato úloha je zaměřena především na odstranění špatných a vytvoření správných návyků při tvorbě programů. V této úloze se zejména hodnotí:

1. Analýza problému a rozdělení úlohy na podproblémy. Program musí být složen z podprogramů. Ve funkci main by se neměl vyskytovat žádný výpočet, pouze volání vašich funkcí se správnými argumenty.
2. Správnost implementace.
3. Schopnost dodržet všechny požadavky.
4. Přehlednost zdrojového textu.

5. Vhodná volba identifikátorů.
6. Efektivita a znovupoužitelnost zvolených algoritmů.

Tipy pro řešení

Pokud s programováním nemáte velké zkušenosti, postupujte od jednodušších problémů ke složitějším. Dekomprimace je jednodušší problém, než komprimace. Komprimace s bloky délky 1 je snazší, než s delšími bloky (ale je zároveň dobré hned myslet i na to složitější řešení, abyste pak program nemuseli přepisovat, ale pouze rozšířit). Vytvářejte podprogramy a snažte se program udržet pořád alespoň částečně funkční. Za částečně funkční program určitě získáte více bodů než za program, který nebude fungovat vůbec.

Pro řešení může být užitečné (ovšem nikoli nezbytné) se seznámit s principem kruhového indexování pole pomocí operace modulo (tzv. kruhový buffer).

Testovací data můžete mít v souboru a do programu je přesměrovat. Bude to fungovat stejně, jako kdybyste je na standardní vstup psali ručně.

```
$ ./proj1 -c 3 < test.txt
```

V Linuxovém shellu Bash to jde i takto:

```
$ ./proj1 -c 2 <<< "popocatepetl"
```

Blíže viz info stránka programu bash.

Protože program používá stejný formát dat pro vstup i výstup, musí fungovat i přesměrování pomocí roury:

```
$ ./proj1 -c 2 <<< "popocatepetl" | ./proj1 -d 2
```

Pro řešení tohoto projektu můžete (ale nemusíte) použít tyto soubory:

- proj1-kostra.c = main + nápověda + náznak zpracování parametrů
- proj1-hola-kostra.c = main + nápověda
- readline.c - Funkce pro načtení libovolně dlouhého řádku do textového řetězce. Může se hodit, pokud místo čtení ze standardního vstupu raději zpracováváte textové řetězce. V tomto případě to je ovšem nouzové řešení s nepřiměřeně velkými paměťovými nároky! V polovině semestru však pro vás bude užitečné prozkoumat její kód. Při použití této funkce v tomto projektu je pravděpodobné, že nezískáte plný počet bodů, protože existují lepší řešení bez ní.

Autor: David Martinek. Poslední modifikace: 19. října 2010. Pokud v tomto dokumentu narazíte na chybu, dejte mi prosím vědět.