Šieste cvičenie

Základné požiadavky:

- JEDEN súbor obsahujúci celý zdrojový kód, v jazyku C (ANSI C podľa prednášok), s názvom a v štruktúre podľa zverejnených inštrukcií (MSTeams).
- Programy musia komunikovať. Ak program očakáva vstup, musí oznamovať aký vstup sa očakáva. Ak vypisuje výsledok, musí vypisovať zrozumiteľný oznam (napr. čo za hodnotu to vypisuje).
- Formátovanie zdrojového kódu by malo zodpovedať približne príkladom z prednášok. Odsadzovanie textov je základ. Príklad dobrého a zlého formátovania sú v prednáške číslo dva na konci.

Úloha prvá. Otočíme reťazec.

I keď existuje funkcie na otočenie reťazca, urobíte si vlastnú. Vytvorte funkciu ktoré otočí reťazec. Funkciu vyskúšajte na pevnom reťazci a použite v ďalších úlohách.

char *otoc(char * vstup)

funkcia reťazec otočí **priamo v poli**.

Príklad: **printf**(“%s\n”, otoc(“`abcdefgh`“));
Vytlačí: `hgfedcba`↵

Úloha druhá. Hráme sa s reťazcom.

Napíšte program, ktorý do **poľa** znakov (použiť môžete statický alebo dynamicky alokovaný reťazec, ak alokujete dynamicky, tak používajte aj **free**, maximálna veľkosť alokácie je **500 znakov**). Na čítanie z klávesnice použite funkciu gets() na načítanie riadku, aby reťazec mohol obsahovať aj biele znaky (medzery a podobne). S týmto reťazcom potom robíte také operácie, aby výsledný reťazec mal tieto vlastnosti:

a) všetky medzery budú zmazané
b) všetky číselné znaky budú nahradené **dvomi** hviezdičkami
c) výsledný reťazec bude otočený

Nie je povolené používať pomocné reťazce a polia. Všetky operácie je potrebné vykonať vo vnútri toho jediného alokovaného priestoru. Výsledok vypíšete jediným volaním funkcie puts() alebo printf(). Riešenia porušujúce tento princíp (používajú pomocné pole alebo vypisujú po znaku) nie sú plnohodnotné riešenia – také riešenie je oveľa ľahšie.

Príklad fungovania:
Zadajte reťazec znakov: `abc def9gh0`↵
Výstup: `**hg**fedcba`↵

Napriek tomu, že výsledný reťazec môže byť dlhší ako pôvodný, s dĺžkou sa netrápte, alokované pole je dosť dlhé (500 znakov) a zadanie od Vás nežiada to kontrolovať.

Úloha tretia. Reťazce a súbory.

Napíšte program, ktorý číta textový súbor **vstup.txt** po riadkoch. Tieto riadky upraví a zapisuje ich do textového súboru **vystup.txt** takto:

- v nepárnych riadkoch prvú polovicu znakov riadku prepíše hviezdičkou
- v párnych riadkoch druhú polovicu znakov riadku prepíše pomlčkou

Predpokladajte, že riadok súboru je dlhý maximálne 1000 znakov.
Ak počet znakov v riadku je nepárny, polovica znakov je myslená BEZ stredného znaku.

Príklad vstup.txt:

```
1234567890
abcdefghijklm
jozko
anca
```

Výstup vo vystup.txt:

```
*****67890
abcdefg------
**zko
an--
```

Iné príklady na precvičenie

1. Napíšte program, ktorý do poľa znakov načíta najviac 50 znakov zo štandardného vstupu. Potom **priamo v poli** pred každý výskyt znaku hviezdičky vloží do poľa znak '>' a za každý takýto výskyt vloží '<'. Takto upravené **pole** vypíše na obrazovku a odriadkuje. V prípade, že pole sa vkladaním znakov naplní, ďalšie znaky sa nevkladajú a na výstupe program pred výpis upraveného poľa vypíše správu `Pole je naplnene` a odriadkuje.
   Zadajte reťazec znakov (max. 20): `*abc*xyz*`↵
   Výstup: `>*<abc>*<xyz>*<`↵

   Zadajte reťazec znakov (max. 20): 123456789*123456*↵
   Výstup: `123456789>*<123456>*`↵

2. Napíšte program, ktorý priamo **z poľa** znakov vymaže všetky výskyty podreťazca. Obsah poľa najviac 50 znakov načítajte ako prvý riadkok vstupu zo štandardného vstupu. Druhý

riadok vstupu bude obsahovať min. 2 znaky a max. 5 znakov (kontrolujte). Výstupom programu bude obsah poľa po zmazaní všetkých výskytov podreťazca.
Zadajte reťazec znakov (max. 50): `qwertabcasdfabczxc`↵
Zadajte reťazec na vymazanie (od 2 do 5 znakov): `abc`↵
Výstup: `qwertasdfzxc`↵

3. Napíšte funkciu int `najdlhsie_opakovanie(char x[], int pocet, char *znak)`, ktorá vráti dĺžku najdlhšieho úseku v poli znakov obsahujúceho ten istý (opakujúci sa) znak. Argument `x` predstavuje pole najviac 30 znakov, argument `pocet` vyjadruje počet platných znakov v poli (`pocet` ≤ 30), v parametri znak funkcia vráti znak ktorý tvorí najdlhšie súvislé opakovanie. Funkciu použite v programe, ktorého vstup pozostáva z jedného riadku obsahujúceho najviac 30-znakové slovo. Slovo je ukončené znakom konca riadku. Slovo môže obsahovať ľubovoľné znaky (nielen písmená). Program má rozlišovať medzi veľkými a malými písmenami.

   Zadajte reťazec znakov (max. 30): `aabbbcCcCcddeeeeff`↵
   Výstup: Najviac opakujúce písmeno je “e” a opakuje sa 4 krat↵

4. Vytvorte program na hádanie slova náhodne vybraného zo súboru `hadanka.txt`. Súbor bude obsahovať v prvom riadku počet slov. Potom bude nasledovať daný počet slov, každé v jednom riadku vždy nasledované znakom konca riadku. Slovo reprezentujte ako pole znakov, pričom koniec slova označte znakom '\0'. Po načítaní slova program umožní používateľovi hádať zvolené slovo a to dvomi rôznymi spôsobmi, z ktorých si používateľ v každom kroku jeden vyberie. Tieto spôsoby sú:

   - hádanie po písmenkách: používateľ zadá znak. Zobrazí sa slovo tak, že všetky doteraz uhádnuté písmená sa zobrazia (všetky výskyty týchto písmen) a namiesto neuhádnutých písmen sa zobrazí podčiarkovník.
   - hádanie celého slova: má zmysel použiť, keď si už používateľ myslí, že slovo uhádol. Vtedy program načíta od používateľa celé slovo.

   Program po každom kroku kontroluje, či používateľ slovo uhádol. V programe nepoužívajte funkcie pre prácu s reťazcami! Používajte na indexy. Pre náhodné čísla použite `srand((unsigned)time(NULL))` – nastavuje náhodný generátor (funkcia `time()` je definovaná v `time.h` a funkcia `srand()` v `stdlib.h`). Funkciu `srand()` je vhodné zavolať na začiatku programu. Potom na priradenie náhodného čísla od 0 do `n`-1 použite príkaz `r = (int) (N * (rand() / (RAND_MAX + 1.0)));` (funkcia `rand()` je definovaná v `stdlib.h`). Vstup a výstup programu formátujte podľa vlastného uváženia.
   Ukážka súboru `hadanka.txt`:
   `5`↵
   `ahoj`↵
   `koleso`↵
   `slovo`↵
   `hodiny`↵
   `potom`↵

Ukážka hádania, kde náhodne vybrané slovo je koleso:

```
(výstup) hadanie po pismenkach (p) alebo po slovach(s)?↲
(vstup) p↲
(výstup) zadajte pismeno: ↲
(vstup) o↲
(výstup) _o___o↲
(výstup) hadanie po pismenkach (p) alebo po slovach(s)?↲
(vstup) p↲
(výstup) zadajte pismeno: ↲
(vstup) e↲
(výstup) _o_e_o↲
(výstup) hadanie po pismenkach (p) alebo po slovach(s)?↲
(vstup) s↲
(výstup) zadajte slovo: ↲
(vstup) koleso↲
(výstup) Bingo!↲
```

5. Napíšte program, ktorý slová zo súborov `prvy.txt` a `druhy.txt` zapíše do súboru `treti.txt` striedavo tak, že každé nepárne slovo v súbore `treti.txt` bude zo súboru `prvy.txt` a každé párne zo súboru `druhy.txt` v poradí, ako boli v pôvodných súboroch. Každé (aj posledné) slovo v súbore `treti.txt` bude nasledované medzerou. Naviac, pred každým slovom bude značka vyjadrujúca, z ktorého súboru slovo pochádza. Ak zo súboru `prvy.txt`, značkou je znak `+`, ak zo súboru `druhy.txt`, značkou je znak `-`. Ak niektorý zo súborov obsahuje viac slov ako druhý, potom tieto budú zapísané za sebou na konci súboru `treti.txt`. Predpokladajte, že slová obsahujú len písmená a oddelené môžu byť len jednou medzerou alebo jedným znakom konca riadku.
   Ukážka súboru `prvy.txt`:

```
Ahojte↲
nasi studenti↲
ktori maju radi programovanie
```

   Ukážka súboru `druhy.txt`:

```
vsetci mili
```

   Ukážka súboru `treti.txt`:

```
+Ahojte -vsetci +nasi -mili +studenti +ktori +maju +radi
+programovanie
```

6. Napíšte program, ktorý z poľa znakov vymaže všetky výskyty podreťazca ako trojice znakov tak, že sa v poli na konci programu (a na výstupe) nebude nachádzať ani jeden výskyt takejto trojice. Ošetrite teda aj prípad, keď sa vymazaním podreťazca v poli objaví ďalší výskyt tohto podreťazca. Napr. z poľa s obsahom `xababccy` získame vymazaním podreťazca `abc` reťazec `xabcy`. Tak ale získavame ďalší výskyt tohto podreťazca, ktorý je treba odstrániť a vymazať ho, čím získame výsledný reťazec `xy`. Obsah poľa najviac 50 znakov načítajte ako prvý riadkok vstupu zo štandardného vstupu. Druhý riadok vstupu bude obsahovať 3 znaky.

Výstupom programu bude obsah poľa po zmazaní všetkých výskytov 3-znakového podreťazca.
Ukážkový vstup:

```
xababccy↵
abc↵
```

Ukážkový výstup:

```
xy↵
```