Exerciții explicate

1. Scrieți o funcție care să se verifice dacă două șiruri de caractere sunt anagrame sau nu. Două șiruri sunt anagrame dacă unul se poate obține din celălalt printr-o permutare a caracterelor sale. De exemplu, șirurile *emerit* și *treime* sunt anagrame, dar șirurile *emerit* și *treimi* nu sunt!

Rezolvare:

Problema poate fi rezolvată în mai multe moduri, dar, indiferent de metoda aleasă, se va testa mai întâi faptul că ambele șiruri au aceeași lungime:

a) Se observă faptul că două șiruri sunt anagrame dacă sunt formate din aceleași caractere și fiecare caracter apare cu aceiași frecvență în ambele șiruri. Astfel, se construiesc 2 vectori de frecvențe, câte unul pentru fiecare șir, și apoi se verifică dacă aceștia sunt egali. În funcție de

tipul caracterelor din cele două șiruri, cei 2 vectori pot avea lungimi de 128/256 de elemente (ASCII/ASCII extins) sau o altă valoare convenabilă (de exemplu, dacă știm faptul că ambele șiruri sunt formate doar din litere mici, cei 2 vectori de frecvențe pot avea doar 26 de elemente).

int anagrame\_1(char \*s, char \*t)

{

unsigned int i;

unsigned int fs[256] = {0}, ft[256] = {0};

if(strlen(s)!= strlen(t))

return 0;

for(i = 0; i < strlen(s); i++)

{

fs[s[i]]++;

ft[t[i]]++;

}

return memcmp(fs, ft, 256\*sizeof(unsigned int)) == 0;

}

O altă variantă, asemănătoare, constă în utilizarea unui singur vector de frecvențe, iar elementele sale vor fi incrementate pentru caracterele din primul șir și apoi decrementate pentru caracterele din cel de-al doilea șir. Cum vom stabili, în acest caz, dacă șirurile sunt anagrame sau nu?

b) Se ordonează lexicografic caracterele din fiecare șir și apoi se verifică egalitatea celor două noi șiruri, folosind funcția strcmp. Atenție, folosind această metodă, cele două șiruri inițiale vor fi modificate!

int cmpCaractere(const void \*a, const void\* b)

{

return \*(char \*)a - \*(char \*)b;

}

int anagrame\_2(char \*s, char \*t)

{

if(strlen(s) != strlen(t))

return 0;

qsort(s, strlen(s), sizeof(char), cmpCaractere);

qsort(t, strlen(t), sizeof(char), cmpCaractere);

return strcmp(s, t) == 0;

}

c) Se caută, pe rând, fiecare caracter din primul șir în cel de-al doilea. În cazul în care caracterul nu este găsit înseamnă că șirurile nu sunt anagrame, altfel se șterge caracterul din cel de-al doilea șir (de ce?) și se trece la următorul caracter din primul șir. Atenție, folosind această metodă, cel de-al doilea șir va fi modificat!

int anagrame\_3(char \*s, char \*t)

{

int i;

char \*p;

if(strlen(s) != strlen(t))

return 0;

for(i = 0; i < strlen(s); i++)

{

p = strchr(t, s[i]);

if(p == NULL)

return 0;

strcpy(p, p+1);

}

return 1;

}

2. Fișierul text cuvinte.in conține pe prima linie două numere naturale a și b (a ≤ b), iar pe următoarele linii un text în care cuvintele sunt despărțite prin spații și semnele de punctuație uzuale. Realizați un program care să scrie în fișierul text cuvinte.out toate cuvintele din fișierul cuvinte. in ale căror lungimi sunt cuprinse între a și b sau mesajul "Imposibil" dacă în fișierul de intrare nu există nici un cuvânt cu proprietatea cerută.

Rezolvare:

În primul rând, vom citi de pe prima linie a fișierului text cele două numere naturale a și b, folosind funcția fscanf. După citirea celor două numere, în buffer-ul asociat fișierului va rămâne un caracter '\n', pe care îl vom elimina printr-o citire în gol a unui caracter, folosind funcția fgetc. Atenție, dacă nu am elimina caracterul '\n' din buffer, prima linie care va fi citită din fișier, folosind funcția fgets, ar fi una formată doar din acel caracter '\n', ci nu prima linie din fișier! În cazul problemei date acest aspect nu influențează rezultatul, dar în cazul altor probleme se poate obține un rezultat incorect (de exemplu, dacă am dori să calculăm și câte linii conține textul din fișier, am obține o valoare mai mare cu 1 decât valoarea corectă).

În continuare, vom citi linie cu linie conținutul fișierului text folosind funcția fgets, o vom împărți în cuvinte folosind funcția strtok și vom scrie în fișierul de ieșire doar cuvintele ale căror lungimi se încadrează între a și b.

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int main()

{

FILE \*fin, \*fout;

int g, lmin, lmax;

char linie[1001], \*cuv;

char delim[] = " .,:;?!\n";

fin = fopen("cuvinte.in", "r");

fout = fopen("cuvinte.out", "w");

fscanf(fin, "%d %d", &lmin, &lmax);

fgetc(fin);

g = 0;

while(fgets(linie, 1001, fin) != NULL)

{

cuv = strtok(linie, delim);

while(cuv != NULL)

{

if(strlen(cuv) >= lmin && strlen(cuv) <= lmax)

{

fprintf(fout, "%s\n", cuv);

g = 1;

}

cuv = strtok(NULL, delim);

}

}

if(g == 0) fprintf(fout, "Imposibil!");

fclose(fin);

fclose(fout);

return 0;

}

Variabila g este folosită pentru a putea verifica faptul dacă am găsit cel puțin un cuvânt cu proprietatea cerută sau nu.