Temă 1 - Programarea Calculatoarelor 2014-2015
Distribuire pachete poştale

Termen de predare: 21.12.2014 ora 23:55
Punctaj: 150 p <=> 1.5p din nota finală

Specificaţii:
Să se implementeze un sistem de distribuire a pachetelor într-un oraş care să funcţioneze după următoarele specificaţii:

- Oraşul este împărţit în nrC cartiere. Un cartier este o structură ce conţine:
  - id - un întreg
  - nume - char *
- Un pachet este reprezentat printr-o structură care conţine următoarele câmpuri:
  - id - un întreg
  - adresă - un vector int[18] - doar valori 0 sau 1
  - idCartier - un întreg
  - strada - un întreg
  - număr - un întreg
  - prioritate - o valoare din mulţimea {1,2,3,4,5}, 5 = prioritate maximă, 1 = prioritate minimă
  - greutate - float
  - mesaj - un scurt mesaj ce însoţeşte pachetul (~ o propozitie) - char* . Mesajul poate să conţină doar: litere, cifre, spaţii şi semne de punctuaţie (.,!?:)
  - codificare mesaj - un întreg calculat după distribuirea pachetului - task I.

  Fiecare pachet are o adresă reprezentată printr-un vector de 18 de poziţii ce reţine doar valori 0 sau 1, cu următoarea semnificaţie: primele 5 poziţii reprezintă id-ul cartierului, următoarele 5 reprezintă strada, iar următoarele 8 reprezintă numărul.

Observaţie! Pentru câmpurile de tip char* - *nume* cartier şi *mesaj* - trebuie să alocaţi spaţiu exact cât este necesar, nu mai mult!

- Fiecare poştaş va avea un id ce coincide cu id-ul cartierului de care este responsabil, el trebuind să distribuie pachetele din acel cartier. Structura prin care e definit poştaşul conţine următoarele câmpuri:
  - id - un întreg din intervalul [0, 31], 32 fiind numărul maxim de poştaşi
  - nrPachete - un întreg din intervalul [0, 49], 50 fiind numărul maxim de pachete de distribuit.
  - vector cu pachetele pe care le are de distribuit

Există un poştaş şef care primeşte iniţial toate pachetele şi are responsabilitatea de a distribui poştaşilor (simpli) pachetele din cartierul fiecăruia.
După ce au primit de la şef pachetele pe care le au de distribuit, postaşii vor:

1. ordona pachetele de care sunt responsabili în funcţie de prioritate şi greutate. Întâi distribuie pachetele cu prioritatea mai mare. Dacă 2 pachete au aceeaşi prioritate, atunci poştaşul va distribui pachetul cu greutatea mai mare.
2. Fiecare destinatar îi dă poştasului un cod pe care îl calculează în funcţie de mesajul pe care îl primeşte împreună cu pachetul (câmpul mesaj) după următorul algoritm:
   i. inversează ordinea cuvintelor în propoziţie, eliminând semnele de punctuaţie.

   Exemplu: dacă mesajul iniţial era: “Ana, are cel mult 3 mere.”, după inversare mesajul trebuie să fie: “mere3multcelareAna”

ii. calculează un cod care este suma produselor dintre codul ASCII al fiecărui caracter şi poziţia pe care acesta se află în interiorul şirului, după ce s-au efectuat prelucrările anterioare. (Numerotarea poziţiilor caracterelor începe de la 0).
iii. codul final este restul împărţirii codului anterior la produsul dintre numărul casei şi numărul străzii pachetului respectiv.

Poştaşul şef va verifica dacă fiecare cod primit este corect, deoarece unii poştaşi obişnuiesc să modifice mesajele din anumite pachete. Verificarea codurilor o face folosindu-se de acelaşi algoritm pe care l-au folosit şi destinatarii pentru a genera codul pentru fiecare pachet.

<u>Task I</u> (90 p):

1. **(20 p)** Să se implementeze o funcţie care citeşte de la tastatură următoarele date:
   nrC - numărul de cartiere şi de poştaşi
   Pe următoarele nrC linii se primeşte numele fiecărui cartier;
   nrP - numărul total de pachete
   Pe următoarele 4 * nrP linii se va primi fiecare pachet (câte patru llinii / pachet):
   - adresa pachetului (18 valori 0 sau 1)
   - prioritatea
   - greutatea
   - mesajul - acesta are lungimea de maxim 100 de caractere.

   Atât id-urile cartierelor cât şi cele ale pachetelor se vor completa automat începând de la valoarea 0. Nu se vor citi de la tastatură!
2. **(15 p)** Să se implementeze o funcţie care, pentru un pachet dat, să extragă din adresa acestuia cartierul, strada şi respectiv numărul, completând câmpurile corespunzătoare din structura pachetului.
3. **(15 p)** Să se implementeze o funcţie care distribuie pachetele poştaşilor.
4. **(20 p)** Scrieţi o funcţie care, primind un vector de structuri de tip pachet, să ordoneze pachetele din vector în funcţie de prioritate (descrescător), iar dacă mai multe pachete au aceeaşi prioritate, să fie ordonate după greutate (tot descrescător), dacă în acest caz există pachete cu aceeaşi greutate şi aceeaşi prioritate, atunci pachetele respective să fie lăsate în ordinea naturală.
5. **(20 p)** Scrieţi funcţiile pentru a calcula codificarea mesajului:
   a. funcţie care să inverseze ordinea cuvintelor dintr-un text si să elimine semnele de punctuaţie.
   b. funcţie care primeşte ca parametru un pachet şi calculează codul mesajului conform algoritmului specificat mai sus.

<u>Task II</u> (40 p):

6. **(30 p)** Scrieţi o funcţie care primind id-ul poştasului să altereze doar acele coduri care au ultimele cifre egale cu id-ul postaşului.
   <u>Exemplu:</u> pentru poştaşul cu id-ul 6, acesta va altera codul 246, dar nu va altera codul 261 (fiindcă primul se termină în 6, iar al doilea, nu), iar dacă poştaşul are id-ul 13, atunci el va altera codul 4913, dar nu va altera codul 403, deoarece ultimele 2 cifre nu sunt 13 în cel de-al doilea caz.
   Se va folosi o funcţie auxiliară care să altereze (modifice) codul unui mesaj după următorul algoritm:
   a. Se calculează factorii primi ai id-ului poştaşului. Aceşti factori primi vor reprezenta poziţiile biţilor modificaţi. Dacă id-ul este 0, atunci vom considera doar valoarea 0 ca “divizor”. În cazul lui 1, singurul divizor va fi considerat 1.
   b. Fiecare bit aflat pe poziţiile descrise la pasul anterior va fi “inversat” (1 devine 0, 0 devine 1).

   Biţii se vor modifica o singură dată, chiar dacă factorul apare la o putere > 1. Dacă există factori mai mari decât 31, atunci acei factori se vor ignora pe parcursul algoritmului. Numerotarea biţilor începe de la 0 şi se face de la dreapta la stânga.
   Funcţia primeşte ca parametri codul şi id-ul unui poştaş.
7. **(10 p)** Scrieţi o funcţie pentru atribuirea unui scor fiecărui poştaş, scorul fiind egal cu numărul de pachete distribuite corect / numărul total de pachete distribuite. Un pachet se consideră distribuit corect dacă este corect codul corespunzător lui.

Observații: se pot defini funcții auxiliare de afişare (a unui vector de pachete/poştaşi) etc.
(15 p) Coding style
(5 p) README
Pentru a primi acest punctaj trebuie să dovediţi că aveţi un stil de codare, că sunteţi consecvenţi în folosirea acestuia şi că scrieţi un cod care se poate citi/inţelege uşor.
Detalii despre coding style găsiți aici.

Input:
Inputul va fi dat aşa cum se specifică la Task I.1. Formatul exact al inputului îl veți lua din fişierul *input.txt* din arhiva Tema1.zip!

Output:
Punctul 1
cartiere
pachete
Punctul 2
pachetele cu adrese completate
Punctul 3
pachetele fiecărui poştaş
Punctul 4
pachetele fiecărui poştaş
Punctul 5
pachetele fiecărui poştaş cu cod completat
Punctul 6
pachetele fiecărui poştaş cu cod completat/alterat
Punctul 7
scoruri poştaşi

Formatul exact al outputului îl veți lua din fişierul *output.txt* din arhiva Tema1.zip!
Atenție! Temele se vor corecta cu un checker automat şi trebuie să respectați întocmai formatul de ieşire!!!

Dacă nu implementați un punct atunci outputul corespunzător lui va fi 0!
Exemplu:
Punctul 1
cartiere si pachete
Punctul 2
0
Punctul 3
postasi cu pachete
Punctul 4
0
Punctul 5
0
Punctul 6
0
Punctul 7
0

Precizări:

- **Atenţie! Temele sunt individuale. Copierea temelor va fi sancţionată, persoanele cu porțiuni de cod identice vor primi 0p pe toată tema!**
- **După deadline-ul temei nu se mai pot încărca teme pe site şi ca urmare, temele respective nu vor fi punctate!**
- Nu se vor folosi variabile globale!
- Rezolvarea temei o veţi scrie în fişierul **tema1.c**. Dacă aveți nevoie de fişiere auxiliare, puteţi să adăugaţi.
- Atenție la sintaxa de **C**! Nu folosiți funcții C++ sau alte construcții specifice **C++**.
- Pentru a facilita testarea programului vostru puteți folosi citirea datelor din fişier. Pentru aceasta se pot folosi instrucțiunile:

  freopen("date.in", "r", stdin);
  freopen("date.out", "w", stdout);

  la începutul programului. Apoi veţi folosi în mod obişnuit funcțiile **scanf** şi **printf**. Astfel, datele de intrare se vor găsi în fişierul "date.in" (nu vor fi introduse de la tastatură), iar datele de ieşire se vor găsi în fişierul "date.out", deci nu vor apărea la consolă. Mai multe detalii găsiţi aici.
- Veți trimite o arhivă **zip** cu numele `grupa_nume_prenume.zip` (exemplu `313CC_Popescu_Maria_Elena.zip`) care va conține fişierele voastre, makefile, README. Fişierele trebuie să se găsească **direct în rădăcina arhivei, nu în alte foldere**.
- În fişierul README spuneti ce task-uri ati implementat, cât v-a luat implementarea aplicației şi explicaţi pe scurt cum aţi implementat problemele. O linie din fişier ar fi bine să aibă maxim 80 de caractere.
- Decărcaţi de pe site arhiva **Tema1.zip,** aceasta conţine un Makefile şi exemple de date de intrare şi de ieşire .
- Câteva indicaţii pentru compilarea temei cu ajutorul utilitarului **make**: fişierul Makefile pe care îl găsiţi ataşat temei vă asigură compilarea problemei. Pentru aceasta, în terminal, tastaţi comanda **make**. Veți observa că vă apare un fişier executabil, tema1 **în cazul in care problema se pot compila !** În cazul în care apar erori, compilarea nu este posibilă şi se va genera un output aferent în terminal. Compilarea se realizează cu flag-ul -Wall (warning all), care va genera output la consola în cazul în care aveți warninguri (**atenţie : warningurile nu opresc compilarea efectivă a problemelor !**). Nu se vor scădea puncte pentru aceste warninguri, dar este bine să le eliminaţi şi pe ele. Pentru a "curăța" folderul curent de executabilele generate, rulaţi din terminal *make clean*. (Dacă doriţi mai multe informaţii despre make, vă recomandăm cartea de USO, pagina 345 :D)
- Sursele vor fi compilate sub **Linux**, cu **gcc**. Pentru compilare vom folosi **Makefile**-ul aflat in arhivă. Dacă programele **nu compilează la comanda make, NU vor fi punctate**.