Tema de casă 2 - Honeycomb

Responsabili: Florin Pop Dorinel Filip Marian Aldescu Adina Budrigă

Termen de predare: 26.11.2017, ora 23:55

Obiective

În urma realizării acestei teme, studentul va fi capabil:

- să utilizeze matrici și vectori
- să respecte formate stricte de intrare/ieșire
- lucreze modularizat, prin implementarea unor funcții ajutătoare

Menționăm că pentru testare (pe vmchecker) se folosește o mașină virtuală pe 32 de biți. Arhiva de test folosește un astfel de binar. În caz că sistemul vostru de operare de pe mașina fizică este pe 64 de biți, sugerăm să faceți testarea finală și pe o mașină (virtuală sau nu) de 32 de biți.

Introducere

Nea Ion, om de la țară, care se ocupă cu apicultura încă de când era copil, își propune să își mărească producția de miere, asa că apelează la nepotul său Dorel, student la ACS.

Bătrânul a observat că unele albine sunt mai relaxate decât altele, în funcție de forma fagurelui, factor esențial în creșterea cantității de miere produse. Dorel i-a spus bunicului că plecând de la niște date generate de el, poate construi forme de fagure extravagante pentru satisfacerea nevoilor albinelor. Așa că bătrânul i-a povestit lui Dorel detalii esențiale despre fagurii de care are nevoie, după cum urmează:

- fagurii sunt organizați pe coloane, putând avea lungimi diferite;
- fiecare coloana este compusă din celule de formă hexagonală, și poate fi ridicată sau coborâtă, pentru a se potrivi perfect;
- în anumite celule se pot află mătci (regine);
- un fagure poate conține maxim 50 de coloane, iar pe fiecare coloană fiind maxim 30 de celule;
- pe o coloană pot convietui maxim 3 mătci.

Enunţul temei

Pentru generarea imaginii unui fagure, Dorel va porni de la un fișier în care pe linii se află câte o descriere a configurației unui singur fagure.

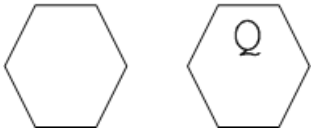
Fisierul va conține mai multe linii, fiecare având următoarea forma: **array_of_int** x **char** x **array_of_pairs** = fagure (modul de construire al unui fagure de miere pornind de la date primitive). Cu alte cuvinte, un fagure este un șir de simboluri (cifre, literă, cifre).

Iată cum va fi reprezentat un fagure cu ajutorul acestui format:

- primele n numere din șir (cele n numere până la citirea literei) reprezintă numărul de celule de pe fiecare coloană (⇒ n coloane);

- urmează litera 'C' sau 'R' care sugerează dacă prima coloana a fagurelui este coborâtă sau ridicată.
- urmeaza un set de perechi de câte 2 cifre care reprezintă coloana și linia unei celule în care se află o matcă.

O celulă a fagurelui este reprezentată cu ajutorul caracterelor: '\' (slash), '/' (backslash), '_' (underscore), ' ' (space) și 'Q', ca în figura de mai jos:



Cerinţa temei

Sarcina voastră este de a-l ajuta pe Dorel să citească un format de intrare care conține configurația unor faguri și să generați, la ieșire, imaginea fagurilor (folosind caractere ASCII).

Exemplu

Input

```
4 6 5 4 7 6 R 1 1 1 4 2 3 2 5 3 4 4 2 5 1 5 4 6 2 6 5
4 5 4 5 4 C
```

Output

```
 _  _  _
/Q\_/ \_/Q\_
\_/ \_/ \_/ \
/ \_/ \_/ \_/
\_/ \_/Q\_/Q\
/ \_/ \_/ \_/
\_/Q\_/ \_/ \
/Q\_/Q\_/Q\_/
\_/ \_/ \_/ \
  \_/ \_/ \_/
  /Q\_/ \_/Q\
  \_/   / \_/
  / \   \_/ \
  \_/   / \_/
      \_/

  _   _
 _/ \_/ \_
/ \_/ \_/ \
\_/ \_/ \_/
/ \_/ \_/ \
\_/ \_/ \_/
/ \_/ \_/ \
\_/ \_/ \_/
/ \_/ \_/ \
\_/ \_/ \_/
  \_/ \_/
```

Precizări legate de implementarea temei

- Implementarea se va face in limbajul C.
- Tema va fi compilată și testată DOAR într-un mediu LINUX.
- Se garantează corectitudinea datelor de intrare.

- Deși programul vostru va trebui să citească direct de la tastatura și să afișeze pe ecran (folosind, de exemplu, scanf și printf), puteți sa citiți datele și să le scrieți în fișiere, folosind redirectările din consolă, fără sa modificați programul. Pentru mediul Windows, dacă fișierul de intrare este in.txt, și cel de ieșire este out.txt, iar programul vostru se numește tema2.exe, veți tipări la consolă:

```
tema2.exe < in.txt > out.txt
```

- Pentru Linux, comanda este similară:

```
./tema2 < in1 > out1
```

Pentru a citi, linie cu linie, input-ul dat de utilizator puteți face apeluri de forma fgets(buffer_linie, dimensiune_buffer, stdin) până când funcția întoarce valoarea NULL. Dacă bufferul oferit este suficient de mare, după fiecare apel veți găsi în el următoarea linie din input.

Se garantează că liniile input-ului nu vor depăși 32K(32678 octeți).

Caracterele din interiorul unui șir de caractere pot fi referite în mod similar elementelor unui tablou unidimensional, cu mențiunea că, în locul specificării explicite a dimensiunii șirului de caractere, pe ultima poziție din șir este pus caracterul \0.

- Fișierele temei trebuie OBLIGATORIU împachetate într-o arhiva de tip '.zip', cu numele 'Grupa_NumePrenume_Tema2.zip'. Această arhivă va conține:
  - Codul sursă al programului vostru.
  - Un fişier Makefile care să conţină regulile build şi clean. Regula build va compila programul într-un executabil cu numele **tema2**. Regula clean va şterge executabilul şi eventual toate binarele intermediare (fişiere obiect) generate de voi.
  - Un fişier README care să conţină explicaţii privitoare la modul de rezolvare;
  - Arhiva temei NU va conține fișiere binare;
- Arhiva va fi trimisă atât pe vmchecker cât și pe moodle.
- O temă care nu compilează nu va fi punctată.

Testare

- Exemplu enunț: honeycombs.zip
- Arhiva de testare folosită pe vmchecker va fi următoarea: honeycombs-checker.zip
- În arhiva checker.zip aveți un script pentru testarea temei (./check.sh)
- Checkerul și directoarele cu inputuri și outputuri vor fi dezarhivate în același director în care se află implementată tema, precum și fișierul Makefile

Copierea parţială sau totală a unei rezolvări din altă sursă va atrage după sine anularea punctajelor pentru toate temele de casă, atât pentru cel care a copiat, cât şi pentru sursa acestuia.