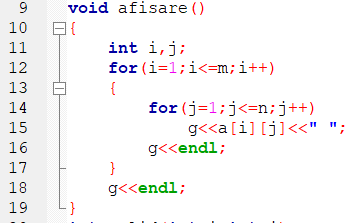
PROBLEMA 13

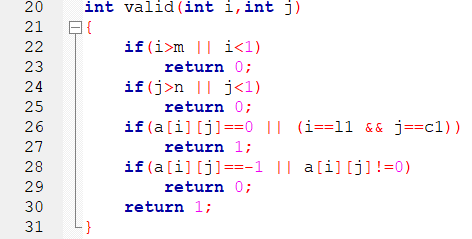
Variabilele problemei sunt:

* m,n – numărul de linii,respectiv coloane
* l0,c0-coordonatele lui Romeo în matrice
* l1,c1-coordonatele Julietei în matrice
* a- matricea
* susjos- vectorul de direcții pentru a parcurge liniile
* drst- vectorul de direcții pentru a parcurge coloanele

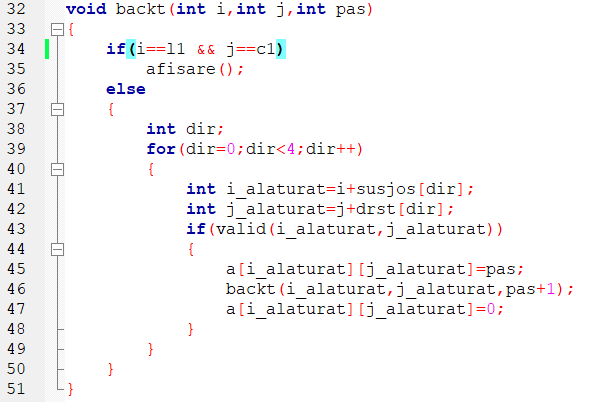
Programul este alcătuit din patru subprograme:



* funcția „afișare” este creată pentru a afișa matricea pe linii și coloane diferite.

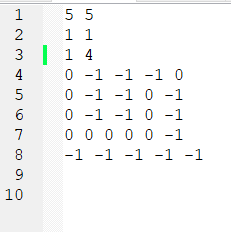
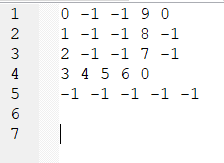


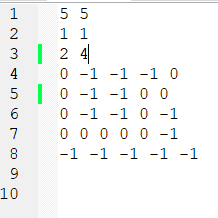
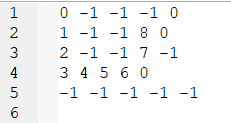
* funția „valid” este creată pentru a verifica în ce condiții se află poziția curentă
  + liniile 22-25 verificăm dacă elementul se află în afara matricei.
  + liniile 26-27 verificăm dacă elementul poate fi parcurs sau dacă am ajuns la destinație.
  + Liniile 28-29 verificăm dacă elementul curent este zid sau dacă este deja parcurs.



* Funcția „backt” este creată pentru partea de backtracking pentru :
  + Linile 34-35 verifică dacă am ajuns la coordonatele Julietei, dacă da afișează
  + Linia 39 for-ul este creat pentru a parcurge vectorii de direcții
  + i\_alaturat și j\_alaturat au fost creați pentru a verifica vecinii poziției actuale
  + Linia 43 verifică condițiile de la funția „valid”
  + Linia 45 condițiile au fost adevărate deci inițializăm elementul vecin cu pas pentru a știi că l-am parcurs deja.
  + Linia 46 apelăam recursiv funția „backt” pentru a verifica și restul cazurilor.
  + Linia 47 la întoarcerea din recursivitate elementele parcurse vor fi reinițializate cu 0 pentru a putea verifica și alte cazuri.

Exemple de date de intrare și ieșire:





Problema are o complexitate de O([nr de direcții=4]m\*n)=O(4m\*n).