Tema 1: Editor text
Liste, stive, cozi

Responsabil: David Iancu

Data publicării: 22 martie 2021
Deadline: 11 aprilie 2021

1. Introducere

Scopul temei este de a implementa un editor de text minimal, fără interfață grafică, având o parte din funcționalitățile editorului vim. De asemenea, se dorește implementarea tuturor structurilor de date folosite (liste/stive/cozi) pentru rezolvare.

2. Descriere

Interfața editorului va fi una text. În cadrul programului, utilizatorul poate să introducă text și comenzi în fișierul ”editor.in”. Trecerea din modul inserare text în modul inserare comenzi și invers se face prin ::i.
Se garantează faptul că în textul introdus de utilizator nu sa va întâlni secvența ”::”.
Toate comenzile vor fi urmate de o linie nouă.
La execuția programului, utilizatorul va primi comenzile în fișierul ”editor.in” iar fișierul final se va numi ”editor.out”. Modificările făcute asupra textului trebuie să fie salvate în fișier doar în urma comenzii save. Editorul va avea un cursor care în mod obișnuit va fi poziționat după ultimul caracter scris, dar el poate fi poziționat cu ajutorul unor comenzi la anumite poziții din text.

Atât liniile, cât și caracterele de pe fiecare linie, se numerotează de la 1. Lista comenzilor:
• u – undo. Anulează rezultatul ultimei operații efectuate.
• r – redo. Anulează rezultatul ultimei operații undo. Comenzile undo și redo se vor putea aplica asupra tuturor comenzilor implementate pe parcursul temei (cu excepția save și quit). După o operație de save, nu se mai poate aplica operația de undo. Aceste două comenzi se pot executa ori de câte ori doresște utilizatorul sau până când nu mai există nicio comandă căreia să i se poată face undo/redo. Comanda redo se poate executa doar atâta timp cât ultima comandă executată a fost un undo. Dacă este executată comanda undo după introducerea unui text, atunci va fi șters tot textul de la ultima comandă până la caracterul curent.
• s – save. Salvează documentul.
• q – quit. Închide editorul. În momentul în care se închide, se salvează doar până la comanda de save.
• b – backspace. Șterge caracterul de dinaintea cursorului.
• dl [line] - delete line. Șterge linia specificată prin parametru. În lipsa acestuia se va șterge linia curentă.
• gl line - goto line. Poziționează cursorul la linia specificată prin parametru, la începutul acesteia.
• gc char [line] - goto character. Poziționează cursorul la caracterul specificat prin parametrul 1 de pe linia indicată de parametrul 2. Al doilea parametru este opțional, în acest caz se va considera linia curentă.

• d [chars] – delete. Șterge un număr de caractere începând cu poziția curentă a cursorului. numărul de caractere este specificat prin primul parametru, sau se consideră a fi 1 dacă acesta lipsește.
• re old_word new_word – replace. Înlocuiește prima apariție de după cursor a cuvântului old word cu textul new word. Se garantează că niciunul dintre argumente nu va conține spații albe.
• ra old_word new_word - replace all. Înlocuiește prima apariție de după cursor a cuvântului old word cu textul new word în felul următor: se inlocuiește prima apariție, se caută a doua apariție din textul original, se înlocuiește, și asa mai departe. Se garantează că niciunul dintre argumente nu va conține spații albe. De exemplu, pentru textul ”aaaaaaaaa” si comanda ra aaa aa, rezultatul trebuie să fie ”aaaaaa”.
• dw word – delete word. Șterge prima apariție de după cursor a cuvântului old word.
• da word – delete all. Șterge toate aparițiile cuvântului word, în felul următor: se șterge prima apariție, apoi se caută a doua apariție, se șterge, și asa mai departe.

3. Cerințe

Să se implementeze toate operațiile de mai susavând în vedere că textul trebuie ținut sub forma unei liste în care fiecare element din listă este un caracter (implementările cu vectori de caractere nu se punctează).

4. Date de intrare/ ieșire

Se citesc date din fisierul ”editor.in” și se afișează textul obținut în fișierul ”editor.out”. Se citesc date până se introduce comanda ”q”.

5. Detalii de implementare

Se vor folosi liste dublu înlănțuite pentru a se memora textul și o stivă pentru a memora comenzile în cazul operațiilor de undo/ redo.

6. Exemplu

editor.in:

```
aaaaaaaa
bbbbbbbb
cccccccc
dddddddd
eeeeeeee
::i
gc 5 2
gl 4
dl
u
u
dl
u
u
r
::i
xxxx
```

xxxx
::i
s
gl 6
::i
yyyyyyyy
::i
s
::i
qqqqqqqq
::i
u
u
u
gl 1
dw y
da yyy
s
re y z
ra ddd dd
q

Conform descrierii de mai sus a comenzilor, trebuie să se petreacă următoarele:

1. Secvența
aaaaaaaa
bbbbbbbb
cccccccc
dddddddd
eeeeeeee va adăuga cele 5 linii de text și va muta cursorul pe prima poziție de pe linia 6.

2. Se trece în modul inseare comenzi (::i). Comanda gc 5 2 va muta cursoul pe poziția a cincea de pe linia a doua.

3. Comanda gl 4 va muta cursorul pe prima poziție de pe linia a patra.

4. Comanda dl, deoarece nu primește un argument cu numărul liniei ce va fi ștearsă, elimină linia curentă (4) și mută cursorul la începutul liniei care îi ia locul. Textul arată acum:

aaaaaaaa
bbbbbbbb
cccccccc
eeeeeeee

5. Comanda u va anula comanda anterioară (dl) și obținem din nou:

aaaaaaaa
bbbbbbbb
cccccccc
dddddddd
eeeeeeee

6. Comanda u va anula comanda anterioară (gl 4), iar acum cursorul va fi poziționat pe linia a doua, simbolul al cincilea.

7. comanda dl va șterge linia curentă și obținem

aaaaaaaa
cccccccc
dddddddd
eeeeeeee

8. Cele două comenzi undo vor anula atât comanda dl, cât și gc 5 2, repoziționând cursorul pe linia a șasea.

9. Comanda redo va reexecuta comanda gc 5 2.

10. Se trece în modul inserare text (::i). Secvența
xxxx
xxxx
va introduce textul dupa poziția a cincea de pe linia a doua. Textul arată acum

aaaaaaaa
bbbbbxxxx
xxxx
bbb
cccccccc
dddddddd
eeeeeeee

11. Comanda s salvează fișierul pe disc cu numele primit ca argument în linia de comandă.

12. Comanda gl 6 și textul care îi urmează vor duce textul în următoarea stare:

aaaaaaaa
bbbbbxxxx
xxxx
bbb
cccccccc
yyyyyyyy
dddddddd
eeeeeeee

13. Se salveaza, apoi se insereaza un text nou (qqqqqqqq). Se execută mai multe operații de undo, dar având în vedere că tocmai au fost salvate modificările, nu se anulează cele anterioare, textul rămânând același ca la pasul precedent.

14. Se merge cu cursorul la prima linie, apoi se șterge o apariție pentru y, urmată de ștergerea tuturor aparițiilor pentru ”yyy”. Rezultatul este următorul:

aaaaaaaa
bbbbbxxxx

xxxx
bbb
cccccccc
y
ddddddd
eeeeeeee

15. Se salvează, apoi se înlocuiește prima apariție a lui y cu z și toate aparițiile pentru ”ddd” cu ”dd”. Rezultatul este următorul:

aaaaaaaa
bbbbbxxxx
xxxx
bbb
cccccccc
z
dddddd
eeeeeeee

16. Comanda q va termina programul. Atenție, fișierul trebuie să nu salveze modificările ulterioare comenzii save.

7. Punctaj

Implementarea corectă a operațiilor – 90p
Coding style, README – 10p

Folosirea de vectori în loc de liste și stive duce la anularea punctajului pe temă!
Tema va fi verificată antiplagiat.

8. Arhiva

Temele trebuie să fie încărcate pe vmchecker. NU se acceptă teme trimise pe email sau altfel decât prin intermediul vmchecker-ului. O rezolvare constă într-o arhiva de tip zip care conține toate fișierele sursă alături de un Makefile ce va fi folosit pentru compilare și un fișier README cu detaliile implementării.
Trebuie respectate următoarele reguli:
• Makefile-ul trebuie să aibă obligatoriu regulile pentru build și clean
• Regula build trebuie să aibă ca efect compilarea surselor și creare binarului ”editor”.
• Fișierele sursă trebuie să fie în rădăcina arhivei. Mai exacţnu faceți un director în care se află toate sursele și arhivați directorul, ci selectați toate sursele și faceți arhiva din ele. Arhiva trebuie să aibă tipul zip și trebuie să fie numită astfel: NumePrenumeGrupaSeriaTema1SD.zip
• Testele sunt făcute astfel încât să fie verificate toate cerinţele temei. În cazul în care sunt ”hardcodate”, fără ca implementarea să fie cea corectă, punctajul nu va fi acordat pe testele respective.

9. Precizări
• Deadline-ul este hard: duminică, 11 aprilie 2021.

• Poziția cursorului este fix după caracterul respectiv. De exemplu, dacă avem textul "abcd" și poziția cursorului este 3, acesta este fix dupa caracterul c. La backspace se șterge c, la delete se șterge d.
• Textul trebuie reținut fără să se folosească vector de caractere (de exemplu, ca listă de liste). Comenzile pot fi reținute și folosint vector de caractere (pentru undo/redo).
• Operația de undo trebuie anulată făcând operațiile invers, nu aveți voie să țineți mai multe liste cu textul, din motive de eficiență a memoriei (ce se întâmplă dacă aveți 100 de operații de undo/ redo, multiplicați de 100 de ori fișierul?).
• Dacă se inserează în mijlocul unui rând, nu se adaugă linie nouă (\n). Dacă se inserează la final de text, se adaugă linie nouă. Vă puteți de seama exact de modul în care trebuie să funcționeze din teste. Cât timp testele trec, orice implementare care respectă restricțiile legate de liste/ stive este punctată.