Tema 1: **Editorul de text**
Liste. Stive. Cozi

Alexandra Maria Vieru
Facultatea de Automatică și Calculatoare

17 martie 2014

1 Scopul temei

Scopul acestei teme 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. Utilizatorul poate să introducă text și comenzi de la tastatură. Orice comandă are următorul format: `::<comanda> [param1] [param2]`

Se garantează faptul că în textul introdus de utilizator nu se va întâlni secvența "::". Toate comenzile vor fi urmate de o linie nouă. La execuția programului utilizatorul va fi capabil să scrie în fișierul al cărui nume este primit ca parametru în linia de comandă. Modificările făcute asupra textului trebuie să fie salvate în fișier doar în urma comenzii `save` sau la fiecare 5 comenzi diferite de `save` executate de utilizator. 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)
- `::s` - `save` (salvează documentul)
- `::q` - `quit` (închide editorul)
- `::b` - `backspace` (șterge caracterul de dinaintea cursorului)

- `::dl [nbr_line]` - `delete line` (șterge linia specificată prin parametru. În lipsa acestuia se va șterge linia curentă.)
- `::gl nbr_line` - `goto line` (poziționează cursorul la linia specificată prin parametru, la începutul acesteia)
- `::gc nbr_char [nbr_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 [nbr_chars]` - `delete` (șterge un număr de caractere de la 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.)

3 Cerințe

3.1 Cerința 1

Implementați comenzile următoare: `save`, `quit`, `backspace`, `goto line`, `goto character`, `delete line` și `delete`.

3.2 Cerința 2

Implementați comenzile `undo` și `redo` (undo the undos) folosind o stivă care să poată să fie redimensionată dinamic, în funcție de necesități. Comenzile `undo` și `redo` se vor putea aplica asupra tuturor comenzilor implementate pe parcursul temei (cu excepția `save` și, desigur, `quit`). Aceste două comenzi se pot executa ori de câte ori dorește 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.

4 Bonus

Implementați comanda `replace` conform specificației de mai sus. Comenzile `undo` și `redo` trebuie să poată să fie executate inclusiv asupra acestei comenzi.

5 Detalii de implementare

Se cere utilizarea structurilor de date studiate la curs și implementate la laborator: stive, cozi, structuri de date ciclice. Se pot folosi liste simplu înlănțuite, liste dublu înlănțuite.

Sugestii:

- folosirea unei structuri de date ciclice pentru ultimele n simboluri introduse;
- folosirea unei stive pentru operațiile revocate cu ajutorul comenzii `undo`;
- folosirea unei liste dublu înlănțuite accesată ca o stivă pentru reținerea comenzilor.

6 Exemple

Se consideră următoarea secvență de text și comenzi trimise editorului.

```
aaaaaaaa
bbbbbbbb
cccccccc
dddddddd
eeeeeeee
::gc 5 2
::gl 4
::dl
::u
::u
::dl
::u
::u
::r
xxxx
xxxx::s
:gl 6
yyyyyyyy
::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. Comanda `::gc 5 2` va muta cursorul pe poziția a cincea de pe linia a doua.
   ```
   aaaaaaaa
   bbbbbbbb
   cccccccc
   dddddddd
   eeeeeeee
   ```

3. Comanda `::gl 4` va muta cursorul pe prima poziție de pe linia a patra.
   ```
   aaaaaaaa
   bbbbbbbb
   cccccccc
   dddddddd
   eeeeeeee
   ```

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.
   ```
   aaaaaaaa
   bbbbbbbb
   cccccccc
   eeeeeeee
   ```

5. Comanda `::u` va anula comanda anterioară (`::dl`).
   ```
   aaaaaaaa
   bbbbbbbb
   cccccccc
   dddddddd
   eeeeeeee
   ```

6. Comanda `::u` va anula comanda anterioară (`::gl 4`), iar acum cursorul va fi repoziționat pe linia a doua, simbolul al cincilea:
   ```
   aaaaaaaa
   bbbbbbbb
   cccccccc
   dddddddd
   eeeeeeee
   ```

7. Comanda `::dl` va șterge linia curentă.
   ```
   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.
```
aaaaaaaa
bbbbbbbb
cccccccc
dddddddd
eeeeeeee
```

9. Comanda `redo` va reexecuta comanda `::gc 5 2`:
```
aaaaaaaa
bbbbbbbb
cccccccc
dddddddd
eeeeeeee
```

10. Secvența
```
xxxx
xxxx
```
va introduce textul începând cu poziția a cincea de pe linia a doua:
```
aaaaaaaa
bbbbxxxx
xxxxbbbb
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 ce îi urmează vor duce textul în următoarea stare:
```
aaaaaaaa
bbbbxxxx
xxxxbbbb
cccccccc
dddddddd
yyyyyyyy
eeeeeeee
```

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

7 Punctaj

Cerința 1	40 puncte
Cerința 2	50 puncte
Codying style, README, warninguri	10 puncte
Bonus	20 puncte

Tabela 1: Punctare

Se vor scădea puncte pentru implementările care folosesc vectori în loc de liste alocate dinamic sau dacă acestea au dimensiune fixă.

Temele vor fi punctate doar pentru testele care sunt trecute din vmchecker.

Nu lăsați warning-urile nerezolvate, altfel veți fi depunctați.

Bonusul nu se punctează integral dacă celelalte comenzi nu funcționează perfect.

8 Arhiva temei

Temele trebuie să fie încărcate pe vmchecker. Pe acest site trebuie folosite credențialele de pe curs.cs.pub.ro pentru logare. NU se acceptă teme trimise pe email sau altfel decât prin intermediul vmchecker-ului.

O rezolvare constă într-o arhivă 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 crearea binarului `myVim`.
- Fișierele sursă trebuie să fie în rădăcina arhivei. Mai exact, nu 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:
Nume_Toate_Prenumele_GrupaSeria_Tema1_SD.zip

Exemplu: Viziru Ștefan Andrei, grupa 312CC va trimite arhiva cu numele:
Viziru_Stefan_Andrei_312CC_Tema1_SD.zip.

NU se vor puncta temele care nu respectă specificațiile anterioare.

9 Precizări

- Tema trebuie predată cel târziu pe 6 aprilie 2014, ora 23:55. Deadline-ul temei este **hard**.
- Tema este **individuală**. Orice tentativă de copiere va fi sancționată.