UNIVERSITETI POLITEKNIK I TIRANËS
FAKULTETI I TEKNOLOGJISË SË INFORMACIONIT
DEPARTAMENTI I ELEKTRONIKËS DHE TELEKOMUNIKACIONIT

Algoritmikë

Punë Laboratori nr. 2

Punoi : Klajdi Gjoka ; Enrik Doçi

Ing. Telekomunikacioni II – A

4 Maj 2016

Lista Rrethore

Për implementimin e listave u zgjodh përdorimi i listes rrethore, ku element i fundit shenjon tek koka e listës. Kjo zgjedhje u bë për të lehtësuar punë meqënëse përdorimi i listës rrethore ishte më i përshtatshëm gjatë kodimit, duke ulur ndjeshëm gjatësinë e kodit dhe vështirësinë e tij (eliminoi rastet e veçanta dhe kushtet që duheshin shtuar për të kryer shtimin dhe heqjen e elementeve në pozicione kritike si në fillim të listës ose në fund tësaj).

Për të mundësuar emërtimi dhe përshkrimi i listave, gjithashtu edhe shmangia e problemeve me fshirjen e elementit të parë të listës (problem ky që doli nga përdorimi i listave rrethore, por i papërfillshëm përpara përmirësimeve), element ii parë i listës u zgjodh të mos ruante informacion mbi artikujt, por mbi listën, ose të rrinte bosh.

Funksione shtesë

1. ReadFileListeArticles(ListeArticles list) – Ky funksion kur thërritet hap skedarin emri i të cilit ruhet në përshkrimin e kokës së listës që merr si argument (qoftë lista e artikujve të disponueshëm ose ajo e artikujve të mbaruar) dhe për rekord (rresht në skedar) e ruan në memorie pasi alokon vend. Më pas thërret funksionin AddArticle , për të shtuar në listën që merr si argument elementin e lexuar.

2. WriteFileListeArticles(ListeArticles list) – Ky funksion bën të kundërtën e funksionit më lart . Ai shkruan në skedarin emri i të cilit ruhet në përshkrimin e kokës së listës që merr si argument të gjithë elementet e

listës. Skedari hapet për shkrim “w” , pra kur thërritet ky funksion skedari fillimisht fshihet, pastaj fillon rishkruajtja e tij.

3. AddRequest(ListeRequests list , Request * a) – Sikurse funksioni AddArticle, ky funksion merr si argument një listë të tipit ListeRequests dhe një pointer që shenjon tek vendi ku ruhet elementi Request të cilin duam ta fusim në listë, dhe e vendos në fillim të listës (renditja e kërkesave nuk është e nevojshme), pra është një funksion me kompleksitet constant.

4. ReadFileRequest(ListeRequests list) - Ky funksion lexon nga skedari “request.txt” të dhënat për kërkesat e reja që do bëhen mbi listën e stokut dhe i ruan tek liste_requests.

5. DeleteRequest(ListeRequests list) – Nëse pas përdorimit të një liste kërkesash përdoruesi do të kishte nevojë për të rikryer veprime të ngjashme, fillimisht lista duhet boshatisur. Për të kryer boshatisjen e listës mjafton të thërritet ky funksion me argument listën e tipit ListeRequests .

6. int menu( ) – Funksioni menu sa herë që thërritet pastron programin nga çdo tekst , printon listen e funksioneve që mund të thërrasë përdoruesi, dhe vlerën që do të skanojë e kthen (përkatësisht variablit int answer në main( )

Optimizime te tjera : free(Article *) përdoret për të liruar vendin në memorie që zë cdo element që do të hiqet nga një listë, gjë që është mjaft e nevojshme për këtë program i cili ka si qëllim ruajtjen e shumë të dhënave .

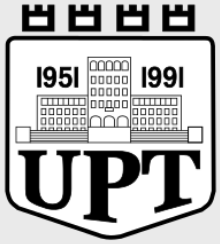
Probleme të hasura dhe zgjidhja e tyre

1. Emri specifik i skedarëve për lista të ndryshme – Meqë u zgjodh që skedarët me të dhëna të emërtoheshin në varësi të të dhënave që do ruanin duhet të krijoheshin dy funksione të njëjta si për leximin edhe për shkrimin e skedarëve. Për të shkurtuar dhe lehtësuar kodin në vetëm një funksion, emri i skedarit u mor si emri i listës (ruajtur në elementin e parë të listës) shtuar me prapashtesën .txt që të aksesohet nga përdoruesi edhe ne programe te tjera .
2. Lëvizja në listën rrethore – Edhe pse mund të mendohet se kushti i ciklit për lëvizje në një listë rrethore është thjeshtë që current!=head(ku current inicializohet si current=head) , kjo do të silte që mos të futeshim fare në cikël. Për këtë arsye cikli nis me current = head->nextArticle

Llogaritja e kompleksitetit

Të gjithë funksionet që kryejnë për vetëm një elemnt shtim/kërkim/fshirje (ku fillimisht duhet bërë kërkimi i elementit që do fshihet) kanë kompleksitetin e rastit më të keq O(n) , pasi lista duhet kontrrolluar deri në fund, kurse kompleksitetin në rastin më të mirë konstant 1 , kur elemti që duhet aksesuar ndodhet në fillim të listës , Por kjo vetëm në rastin kur duam të aksesojmë vetem 1 element , funksionet e tjerë që bëjnë kërkime për të gjithë elementët me një tipar kanë kompleksitetin O(n) .

Funksione si AddRequest , funksioni i të cilit është vetëm shtimi i elemntit në krye të listës ka kompleksitetin 1 , një nga përfitimet e përdorimit të listave , kundrejt O(n) që do të ishte kompleksiteti nëse do përdornim vektorë .



Punoi : Enrik Doçi ; Klajdi Gjoka

Ing. Telekomunikacion II-A

4 Maj 2016