Evidenta si gestionarea structurii de directoare si fisiere pe disc

1. Obiectiv

Aplicatia permite evidenta si gestionarea directoarelor si fisierelor de pe disc. Operatiile cu directoare si fisiere sunt efectuate prin comenzi specifice introduse in interfata de tip consola a aplicatiei.

2. Descriere generală

Aplicația rulează în mod consolă și așteaptă introducerea unei comenzi din cele disponibile cu/fără parametri asociați, conform sintaxei de definire a comenzii prezentată în subcapitolele următoare.
Alături de introducerea secvențială a comenzilor, sistemul acceptă și precizarea unui fișier batch (listă de comenzi) ca input pentru realizarea de operații aferente unuia din cele patru module.
Parametrii comenzilor introduse în linia de comandă a aplicației sunt gestionați intern prin structură de date de tip FIFO, fiind prelucrați secvențial. Comanda este procesata prin intermediul unei structuri de tip FIFO. După parsarea comenzii și stocarea acesteia în structura FIFO, aceasta se validează conform sintaxei de definire. Comanda validată se execută conform specificației de funcționalitate definită în subcapitolele următoare.
De asemenea, aplicația gestionează un fișier de log-uri cu privire la comenzile introduse în linia de comandă sau cele stocate în fișiere batch. Log-urile sunt salvate într-un fișier pe baza de timestamp. Fiecare log are în dreptul lui timestampul aferent momentului în care comanda a fost executată.

3. Funcționalitățile sistemului

3.1 Comenzi implementate

3.1.1 Comanda `list`

Comanda are următoare sintaxă definită:

`list [<Path_To>]`

unde:

- **`Path_To`** – locatia pentru care se identifica directoarele si fisierele;

Comanda implementează următoarele funcționalități:

- Adaugă în structura de date internă de tip Tabela de Dispersie, setul de date în concordanță locatia furnizata; dimensiunea Tabelei este determinata avand in vedere potentialul de incarcare a acesteia;
- Afișează în linia de comandă succesul sau eșecul efectuării operației;
- In cazul in care parametrul <Path_To> nu este precizat (parametrul este optional), este considerata locatia fisierului executabil aferent aplicatiei.

3.1.2 Comanda `filter`

Comanda are următoare sintaxă definită:

`filter [<Path_To>] [-d | -f] [<Size>]`

unde:

- **`Path_To`** – locatia pentru care se identifica directoarele si fisierele;
- **`-d`** – se iau in considerare doar directoarele;
- **`-f`** – se iau in considerare doar fisierele;
- **`Size`** – dimensiunea directoarelor si/sau fisierelor filtrate prin comanda; este exprimata in KB.

Comanda implementează următoarele funcționalități:

- Creeaza structura de date internă de tip arborescent, setul de date în concordanță parametrii comenzii; structura arborescenta este implementata astfel incat sa permita identificarea in timp optim a elementelor componente (directoare si/sau fisiere); pentru fiecare intrare (director si/sau fisier) se genereaza prin aplicatie un id unic; daca structura exista deja prin comanda filter anterioara, aceasta este dezalocata si se creeaza o structura noua conform cu parametrii precizati;
- Afișează în linia de comandă succesul sau eșecul efectuării operației;
- In cazul in care parametrul Path_To nu este precizat (parametrul este optional), este considerata locatia fisierului executabil aferent aplicatiei;
- In cazul in care parametrul -d sau -f nu este precizat (parametrul este optional), se considera atat directoarele, cat si fisierele;
- In cazul in care parametrul Size nu este precizat (parametrul este optional), se considera directoarele si/sau fisierele indiferent de dimensiune

3.1.3 Comanda `get`

Comanda are următoare sintaxă definită:

`get [-s] [-p] [<Size>] [<Date>] [<Extension>]`

unde:

- **`-s`** – fisierele extrase din structura de date interna sunt sortate alfabetic;
- **`-p`** – denumirea fisierelor este precizata de path (locatia) de pe disc;
- **`Size`** – se iau in considerare doar fisierele cu dimensiune mai mare decat cea precizata; este exprimata in KB;
- **`Date`** – se iau in considerare doar fisierele create dupa data precizata (inclusiv)
- **`Extension`** – se iau in considerare doar fisierele care au extensia precizata.

Comanda implementează următoarele funcționalități:

- Creeaza structura de date internă de tip structura liniara alocata dinamic prin considerarea si parsarea Tabelei de Dispersie creata si atualizata prin comenzi **`list`** anterioare (comanda 3.1.1);
- Afișează în linia de comandă succesul sau eșecul efectuării operației;
- In cazul in care parametrul -s nu este precizat (parametrul este optional), se considera fisierele in ordinea aferenta parsarii Tabelei de Dispersie;
- In cazul in care parametrul -p nu este precizat (parametrul este optional), se considera doar denumirile de fisiere (impreuna cu extensiile acerstora);
- In cazul in care parametrul Size nu este precizat (parametrul este optional), se considera toate fisierele indiferent de dimensiune;
- In cazul in care parametrul Extension nu este precizat (parametrul este optional), se considera toate fisierele indiferent de extensie.

3.1.4 Comanda `saveget`

Comanda are următoare sintaxă definită:

`saveget <Nume_fisier>`

unde:

- **`Nume_fisier`** – numele fisierului in care se salveaza continutul structurii liniare creata prin comanda 3.1.3; poate fi prefixat de path (locatie);

Comanda implementează următoarele funcționalități:

- Creeaza fisier pe disc in locatia precizata; daca locatia (path) lipseste, se considera calea fisierului executabil aferent aplicatiei ca locatie implicita;
- Afișează în linia de comandă succesul sau eșecul efectuării operației.

3.1.5 Comanda `exit`

Comanda are următoare sintaxă definită:

`exit`

Comanda determină oprirea execuției aplicației. Toate structurile de date interne aplicatiei sunt dezalocate.

3.2 Fluxuri si procese

3.2.1 Interpretarea unei comenzi

Comanda este executata de operator la nivel consola fiind apoi procesata de un modul specializat care permite parsarea comenzii si identificarea fiecarei comenzi specifice

4. Fişiere utilizate la nivel de aplicaţie

1. Fisierul text batch cu lista de comenzi aplicata pentru un caz de utilizare a aplicatiei (**`comenzi.txt`**);
2. Fişierul de log-uri (**`logs.txt`**).

5. Fișierul de log-uri

Fișierul de log-uri (**`logs.txt`**) este un fișier text care stochează pe fiecare linie următoarele date:

- Timestamp aferent momentului in care comanda a fost executata ;
- Comanda introdusă împreună cu parametrii aferenți;
- Rezultatul procesului (comentarii returnate de funcții ca urmare a execuției).

Fiecare comandă introdusă este salvată la sfârșitul fișierului de log-uri, astfel încât acesta să prezinte un istoric al operațiilor efectuate prin prezenta aplicație.

De asemenea, în cazul utilizării unui fișier batch, lista de comenzi conținută de acesta va fi salvată în fișierul de log-uri cu datele aferente lansării fișierului pentru toate comenzile conținute de acesta.

6. Observaţii generale

1. Toate fişierele program furnizate la intrare se presupun a fi corecte;
2. Toate fişierele temporare utilizate de către program vor fi create în directorul curent;
3. Programele vor fi compuse dintr-un singur fişier sursă C/C++. Pentru implementarea şi utilizarea structurilor de date, nu este permisă utilizarea altor biblioteci (ex. STL) în afara celor specificate în standardul ANSI.
4. Funcția HASH folosită pentru implementarea tabelei de dispersie este diferită de formele prezentate la curs; functia trebuie sa considere caracterul specific al elementelor din Tabela de Dispersie (directoare si fisiere).