**PROGRAMAREA CALCULATOARELOR SI LIMBAJE DE PROGRAMARE**

Top of Form

Bottom of Form

**Profesor Coordonator** **Student**

Gabriela Olteanu Nica Silviu

Cosmin Fudulu

Giorgian Neculoiu

**Cuprins**

* **Introducere**

**Motivare alegere temă**

**Obiective**

* **Tehnologii utilizate**
* **Studiu de caz – Tema aleasă**

**Implementare pas cu pas**

* **Concluzie**
* **Bibliografie**
* **Anexe**

## 1. Introducere

În lumea dezvoltării software, gestionarea fișierelor este una dintre cele mai fundamentale și frecvent întâlnite activități. Indiferent dacă vorbim despre salvarea unui document, logarea activităților unei aplicații sau preluarea unor setări dintr-un fișier de configurare, interacțiunea cu fișierele este o componentă esențială a majorității aplicațiilor moderne.

Limbajul C++, cunoscut pentru puterea, flexibilitatea și performanța sa, oferă programatorilor un set robust de instrumente pentru lucrul cu fișiere. Prin intermediul bibliotecilor sale standard precum <fstream>, C++ face posibilă gestionarea fișierelor textuale și binare într-un mod clar și eficient. Cu toate acestea, deși este o temă esențială, manipularea fișierelor este uneori tratată superficial în procesul de învățare, deși reprezintă o punte importantă între logica programării și realitatea practică a stocării datelor.

Scopul acestui proiect este de a aduce în prim-plan tocmai această latură pragmatică a programării. Prin construirea unei aplicații simple care permite utilizatorului să creeze, să scrie, să citească, să editeze și să șteargă un fișier, vom evidenția importanța înțelegerii modului în care datele sunt gestionate în afara memoriei volatile a programului. Aplicația va fi una interactivă, gândită să fie intuitivă și clară, astfel încât să poată fi folosită ca model sau punct de plecare pentru proiecte mai complexe.

De asemenea, utilizarea platformei OnlineGDB aduce un avantaj suplimentar: rularea codului direct din browser, fără a necesita configurări locale. Astfel, procesul de testare și verificare devine accesibil oricui, indiferent de dispozitiv sau sistem de operare.

În concluzie, acest proiect nu urmărește doar implementarea unei funcționalități tehnice, ci și cultivarea unei înțelegeri mai profunde asupra modului în care aplicațiile moderne stochează și gestionează date. Prin această lucrare, ne propunem să aducem mai aproape de realitate conceptele teoretice legate de fluxuri, fișiere și persistența informației în programare.

## **Motivare Alegere Temă**

Tema aleasă – realizarea unei aplicații în C++ care să permită gestionarea unui fișier (creare, scriere, citire, editare și ștergere) – nu a fost aleasă întâmplător. În spatele acestei decizii stă o motivație profund legată de importanța practică și didactică a subiectului.

În primul rând, lucrul cu fișiere este o componentă esențială a oricărui sistem informatic. Oricât de performant ar fi un algoritm sau oricât de intuitivă ar fi interfața unei aplicații, fără o modalitate eficientă de a salva date, de a le prelucra sau de a le recupera ulterior, funcționalitatea este limitată. Fișierele reprezintă forma de bază a persistenței datelor – ele permit programelor să „țină minte” informații și după încheierea execuției.

În al doilea rând, C++ este un limbaj care oferă control fin asupra memoriei și resurselor sistemului, dar care, în același timp, integrează concepte moderne precum programarea orientată pe obiect. Alegerea C++ pentru acest proiect a fost motivată de dorința de a învăța să folosim acest control pentru operațiuni concrete și utile, cum este manipularea fișierelor. Acest proces implică atât înțelegerea fluxurilor de intrare/ieșire, cât și gestionarea corectă a excepțiilor și a stărilor fișierelor.

De asemenea, un alt motiv important a fost accesibilitatea temei pentru începători, dar și potențialul său de extindere pentru proiecte mai avansate. Chiar dacă la prima vedere pare o temă simplă, lucrul cu fișiere este un exercițiu excelent pentru consolidarea noțiunilor fundamentale de programare – de la tipuri de date și funcții, până la structura unui program modular și interacțiunea cu utilizatorul.

În plus, alegerea acestei teme oferă posibilitatea unei testări ușoare și rapide, inclusiv pe platforme online precum OnlineGDB, fără a fi nevoie de configurări complicate. Astfel, oricine dorește să studieze sau să experimenteze poate rula codul în timp real, într-un mediu intuitiv și prietenos.

## Obiective

Scopul principal al acestui proiect este acela de a construi o aplicație funcțională în C++ care să permită gestionarea completă a unui fișier: de la creare și scriere, până la editare, citire și ștergere. Însă dincolo de rezultatul final, proiectul este conceput ca un exercițiu de învățare și consolidare a unor cunoștințe fundamentale în programare.

Un prim obiectiv este familiarizarea cu bibliotecile standard C++ destinate lucrului cu fișiere. Prin utilizarea <fstream> și <cstdio>, programatorul deprinde noțiuni esențiale despre manipularea fișierelor textuale, înțelegând modul în care funcționează fluxurile de intrare/ieșire (streams) în C++. Acest pas este crucial pentru orice dezvoltator care își dorește să construiască aplicații ce interacționează cu sistemul de fișiere sau care gestionează date persistente.

Un al doilea obiectiv constă în implementarea funcționalităților de bază legate de fișiere: crearea unui fișier nou, scrierea de conținut într-un fișier existent, citirea conținutului acestuia, rescrierea (editarea) conținutului și, în final, ștergerea fișierului. Aceste operații reflectă un flux complet de gestionare a fișierelor, așa cum este întâlnit în aplicațiile reale.

De asemenea, proiectul urmărește demonstrerea practică a unui ciclu complet de manipulare a fișierelor. Nu este vorba doar despre scrierea unui cod care „funcționează”, ci despre înțelegerea fiecărui pas, despre organizarea logică a funcțiilor și despre interacțiunea clară cu utilizatorul prin intermediul unui meniu intuitiv.

Un obiectiv adiacent, dar extrem de valoros, este consolidarea înțelegerii asupra conceptului de fluxuri (streams) și a manipulării de text în C++. În practică, aproape orice aplicație prelucrează sau generează text, iar lucrul cu fișiere este una dintre cele mai bune modalități de a învăța să o facem eficient și sigur.

**2. Tehnologii utilizate**

C++ este un limbaj de programare extrem de versatil, folosit pe scară largă atât în industrie, cât și în mediul academic. A fost creat de Bjarne Stroustrup, la începutul anilor ’80, din dorința de a aduce împreună eficiența limbajului C cu conceptele moderne ale programării orientate pe obiect. Practic, C++ a fost gândit ca un pas înainte față de C, un limbaj care să ofere dezvoltatorilor mai multă putere și flexibilitate atunci când creează aplicații complexe.

Unul dintre cele mai mari avantaje ale C++ este echilibrul pe care îl oferă între controlul detaliat asupra resurselor (precum memoria) și posibilitatea de a scrie cod clar, bine structurat. Datorită acestui echilibru, C++ este alegerea perfectă pentru aplicații care au nevoie de performanță maximă: jocuri video, simulări fizice, aplicații care rulează în timp real, sisteme de operare și multe altele. În același timp, suportul nativ pentru programarea orientată pe obiect îl face ideal pentru organizarea logică a codului, prin clase, moștenire și polimorfism.

Un alt aspect important este faptul că C++ este pe deplin compatibil cu limbajul C. Asta înseamnă că foarte mult cod scris în C poate fi folosit direct într-un program C++, ceea ce este extrem de util pentru dezvoltatorii care trebuie să lucreze cu proiecte vechi sau să integreze biblioteci scrise în C.

De-a lungul anilor, C++ a evoluat constant. Noile standarde – C++11, C++14, C++17, C++20 și recent C++23 – au adus îmbunătățiri semnificative, de la o sintaxă mai elegantă, la funcționalități moderne precum lambda functions, smart pointers sau concurența nativă.

Am mai folosit și OnlineGDB. Acesta este o platformă online de dezvoltare integrată (IDE) care oferă posibilitatea de a scrie, compila și rula cod C++ direct în browser, fără a fi nevoie să instalezi vreun software suplimentar pe calculatorul tău. Aceasta facilitează procesul de programare printr-o interfață prietenoasă și intuitivă, fiind ideală pentru testarea rapidă a codului, învățare sau demonstrații. Prin rularea interactivă, utilizatorii pot introduce date în timp real, pot vedea rezultatele imediat și pot depana programele fără a părăsi mediul online. Astfel, OnlineGDB simplifică accesul la un mediu complet de programare, fiind accesibil de pe orice dispozitiv conectat la internet.

Biblioteci C++ standard:

<iostream>  
Această bibliotecă oferă facilități pentru operațiuni de intrare și ieșire standard, adică pentru comunicarea programului cu utilizatorul prin consola text. Prin intermediul obiectelor predefinite precum std::cin (pentru citire de la tastatură) și std::cout (pentru afișare pe ecran), programatorii pot primi date de la utilizator și pot afișa mesaje sau rezultate.

<fstream>  
Este biblioteca esențială pentru lucrul cu fișiere în C++. Aceasta oferă clase specializate precum std::ofstream (pentru scriere în fișiere), std::ifstream (pentru citire din fișiere) și std::fstream (pentru operațiuni mixte de citire și scriere). Cu ajutorul acestei biblioteci, programul poate crea, deschide, modifica și salva conținut în fișiere externe, permițând astfel stocarea persistentă a datelor.

<cstdio>  
Această bibliotecă provine din limbajul C și este inclusă în C++ pentru a oferi funcții de manipulare la nivel scăzut a fișierelor și a altor resurse. În proiectul nostru, este utilizată în special pentru funcția remove(), care permite ștergerea fișierelor de pe disc. Astfel, <cstdio> extinde capabilitățile de gestionare a fișierelor dincolo de simpla citire și scriere.

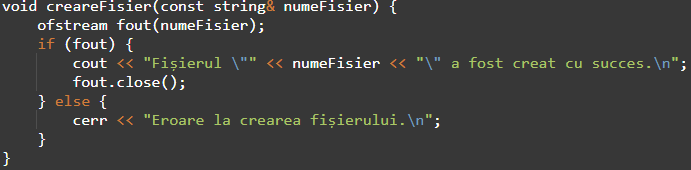
<string>  
Biblioteca <string> oferă suport pentru manipularea șirurilor de caractere în C++. Spre deosebire de tablourile de caractere tradiționale din C, clasa std::string este o abstracție puternică și flexibilă, care permite crearea, modificarea, concatenarea și compararea textelor într-un mod simplu și sigur. Aceasta facilitează lucrul cu text în programe, făcând codul mai clar și mai ușor de întreținut.

**3. Studiu de caz - Tema aleasa**

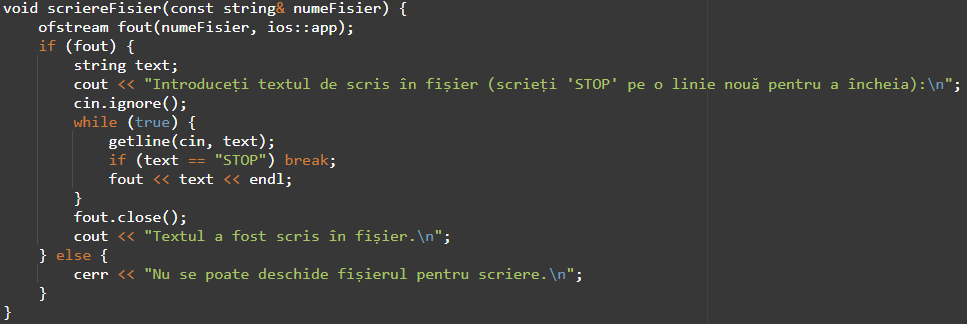
Tema aleasă, și anume crearea, scrierea, editarea și ștergerea unui fișier în C++, reprezintă o problemă foarte practică și relevantă în dezvoltarea software-ului. În viața de zi cu zi a oricărui programator, interacțiunea cu fișiere este inevitabilă — fie că este vorba despre salvarea unor date, logarea activității, configurarea aplicațiilor sau manipularea unor documente text. Acest studiu de caz pune în lumină exact aceste operațiuni fundamentale, cu ajutorul limbajului C++ și al bibliotecilor sale standard.

**Implementare pas cu pas:**

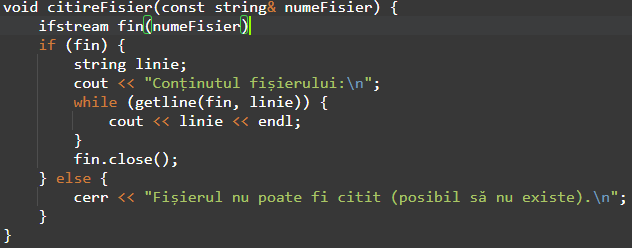
Pornind de la ideea simplă că un program trebuie să poată crea un fișier atunci când acesta nu există, aplicația exemplificată asigură această funcționalitate prin deschiderea unui flux de ieșire care, în mod implicit, creează fișierul dacă nu este găsit. Această operațiune pare banală, dar este esențială pentru orice aplicație ce lucrează cu date persistente. Astfel, utilizatorul poate salva informații importante într-un fișier text care să existe chiar și după închiderea programului.

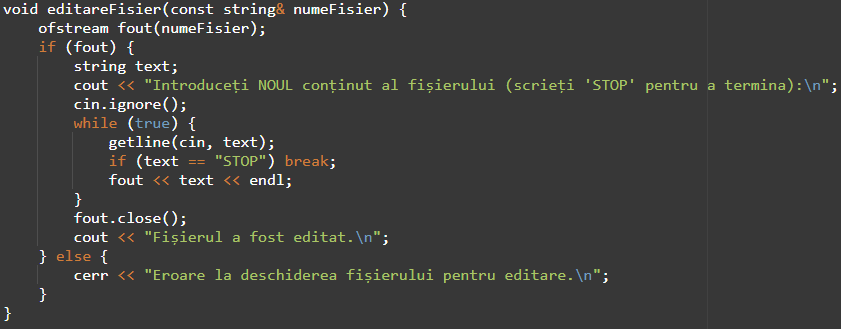


Mai departe, programul permite scrierea unor linii de text în fișier, adăugând conținut care poate fi ulterior citit sau modificat. Operația de scriere este gestionată cu atenție pentru a asigura că datele sunt stocate corect, iar fluxul de ieșire este închis imediat după terminarea scrierii, evitând pierderile de date sau blocajele.

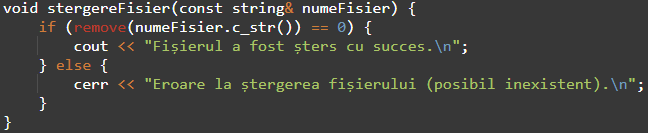


Citierea fișierului reprezintă o etapă vitală în care programul preia informațiile stocate anterior și le afișează utilizatorului. Acest lucru nu doar confirmă faptul că scrierea a fost realizată cu succes, dar oferă și o bază pentru orice proces ulterior care ar trebui să prelucreze datele respective. Metoda de citire linie cu linie folosește un mod intuitiv și eficient de a parcurge conținutul fișierului.

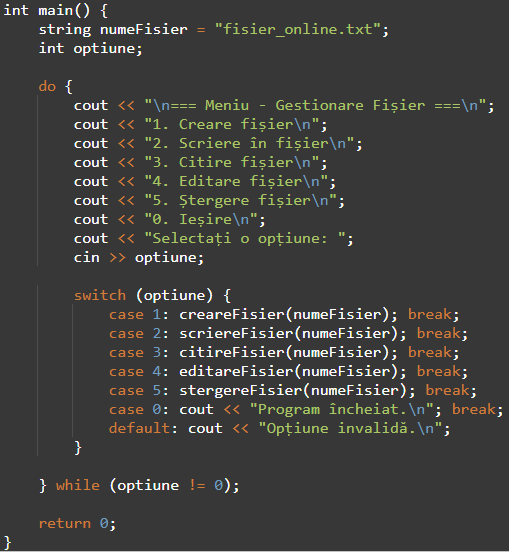


Editarea fișierului, adică suprascrierea conținutului existent cu text nou, arată flexibilitatea programului de a actualiza informațiile stocate fără a crea un fișier nou de fiecare dată. Aceasta este o caracteristică folosită frecvent în aplicațiile reale, unde datele trebuie să fie actualizate în mod constant, cum ar fi în fișierele de configurare sau rapoarte.

Ștergerea fișierului, ultima etapă, ilustrează gestionarea resurselor și curățenia în cadrul aplicației. Eliminarea unui fișier atunci când acesta nu mai este necesar previne ocuparea inutilă a spațiului de stocare și contribuie la o mai bună organizare a datelor.



În ultima bucla avem „mainul”, care, implementează un meniu interactiv pentru utilizator, care poate alege ce operație să facă asupra fișierului, iar programul răspunde prin apelarea funcției corespunzătoare pentru fiecare caz. Bucla repetitivă asigură o experiență fluidă, fără întreruperi inutile.



## 

## 

## **7. Concluzie**

Realizarea acestui proiect dedicat gestionării fișierelor în C++ a avut un impact semnificativ în consolidarea cunoștințelor teoretice și practice privind manipularea datelor persistente într-un limbaj de programare de nivel înalt. Prin abordarea unei aplicații interactive, care permite crearea, scrierea, citirea, editarea și ștergerea unui fișier text, s-a obținut o înțelegere clară asupra modului în care C++ interacționează cu sistemul de fișiere al unui computer.

Una dintre cele mai mari realizări ale proiectului constă în simplitatea și claritatea cu care funcțiile de bază ale lucrului cu fișiere au fost implementate și testate. Acest lucru a permis nu doar obținerea unui rezultat funcțional, ci și dezvoltarea unei intuiții despre modul în care datele pot fi stocate, regăsite și manipulate în aplicații reale. În plus, folosirea unui meniu interactiv a transformat aplicația într-o unealtă ușor de utilizat, intuitivă și prietenoasă pentru utilizator.

Pe parcursul dezvoltării, au fost evidențiate diferențele dintre metodele C++ clasice de lucru cu fișiere (prin fstream) și funcțiile C-style (precum remove() din <cstdio>), ceea ce a condus la o înțelegere mai largă și comparativă a uneltelor oferite de limbaj. Astfel, nu doar că am învățat cum să lucrăm cu aceste biblioteci, ci și când și de ce să le folosim în diverse scenarii.

Un alt aspect important este că proiectul a fost rulat și testat pe platforma OnlineGDB, care a permis o dezvoltare rapidă, fără a necesita configurări suplimentare pe sistemul local. Acest lucru a subliniat importanța instrumentelor moderne de lucru colaborativ și în cloud, devenite tot mai esențiale în programarea actuală.

**8. Bibliografie**

1) Bjarne Stroustrup - The C++ Programming Language, Addison-Wesley

2) <https://cplusplus.com/reference/fstream/>

3) <https://en.cppreference.com/w/cpp/io>

4) Documentația oficială GCC

5) Laboratoare GitHub

6) <https://www.youtube.com/watch?v=aHPuxLiX8IQ&t=521s>

7) https://www.youtube.com/watch?v=EaHFhms\_Shw&pp=ygUUZmlsZSBjcmVhdGlvbiBpbiBjKys%3D

8) https://www.youtube.com/watch?v=-TkoO8Z07hI&pp=ygUPYysrIGZ1bGwgY291cnNl

9) https://www.w3schools.com/cpp/

10) https://www.w3schools.com/cpp/cpp\_files.asp

11) https://www.geeksforgeeks.org/c-program-to-create-a-file/

12) https://stackoverflow.com/questions/478075/creating-files-in-c

13) https://www.ibm.com/docs/en/debug-for-zos/15.0.x?topic=started-creating-c-file

**9) Anexe**

### a) Anexa 1: Codul sursă complet:

Codul care stă la baza acestui proiect este conceput pentru a permite utilizatorului să interacționeze intuitiv cu un fișier text printr-un meniu afișat în consolă. Utilizatorul are posibilitatea să creeze un fișier, să adauge conținut în el, să vizualizeze ceea ce conține, să editeze textul existent sau, dacă este necesar, să îl șteargă complet de pe disc.

Acest comportament este împărțit în funcții distincte, fiecare fiind apelată în funcție de alegerea utilizatorului. Structura codului respectă principiile modularității, astfel încât fiecare operație asupra fișierului este tratată separat pentru claritate și întreținere facilă.

Codul complet este organizat astfel:

-main() — funcția principală care conține meniul interactiv.

-creareFisier() — creează fișierul dacă nu există deja.

-scriereFisier() — adaugă text în fișier, fără a șterge conținutul existent.

-citireFisier() — afișează în consolă conținutul actual al fișierului.

-editareFisier() — rescrie complet conținutul fișierului.

-stergereFisier() — șterge definitiv fișierul de pe disc.

Fiecare funcție este însoțită de verificări de bază pentru a gestiona cazurile în care operațiile pot eșua, cum ar fi imposibilitatea de a deschide un fișier.

### b) Anexa 2: Capturi de ecran:

Această anexă poate include capturi de ecran care documentează vizual comportamentul aplicației. Fiind o aplicație de tip consolă, capturile de ecran ar trebui să surprindă următoarele etape:

-Execuția programului în OnlineGDB, unde se afișează meniul cu opțiunile disponibile.

-Introducerea unei opțiuni de către utilizator, de exemplu, alegerea opțiunii 2 pentru scrierea în fișier.

-Vizualizarea rezultatului în consolă, cum ar fi afișarea conținutului fișierului după citire.

-Mesajul de confirmare la ștergerea fișierului, demonstrând că fișierul a fost eliminat cu succes.

Aceste capturi ajută la înțelegerea practică a funcționalității codului și pot fi adăugate într-un document Word sau PDF final pentru a susține partea teoretică.

### c)Anexa 3: Mesaje afișate în timpul execuției:

Pentru o mai bună transparență a execuției, programul returnează mesaje clare și concise în funcție de acțiunea întreprinsă. Exemple:

-La crearea fișierului: „Fișierul a fost creat cu succes.”

-După scriere: „Textul a fost adăugat în fișier.”

-În cazul unei citiri reușite: „Conținutul fișierului este următorul:”

-Dacă fișierul este editat: „Fișierul a fost rescris.”

-La ștergere: „Fișierul a fost șters cu succes.”

-În caz de eroare (de exemplu, fișierul nu există): „Eroare: fișierul nu a putut fi deschis.”

Aceste mesaje sunt esențiale pentru interacțiunea utilizatorului cu aplicația și contribuie la o experiență fluentă, clară și prietenoasă.