**Proiect la PCLP**

Top of Form

Bottom of Form

**Todo-List**

**Profesor Coordonator Student**

Olteanu Gabriela Bucur Razvan

Fudulu Cosmin

**Cuprins**

[Proiect la PCLP 1](#_Toc1959636814)

[1. Introducere 4](#_Toc784713813)

[Motivarea Alegerii Temei 4](#_Toc1652908479)

[Obiective propuse 5](#_Toc1775792287)

[Evoluția Pieței 6](#_Toc105046195)

[1. Origini și evoluție timpurie 6](#_Toc1919505112)

[2. Perioada aplicațiilor mobile și cloud (2008 – prezent) 7](#_Toc440507333)

[3. Tendințe actuale în dezvoltarea aplicațiilor To-Do 7](#_Toc516295753)

[4. Valoarea economică și utilizare globală 8](#_Toc1586775404)

[5. De ce sunt importante aplicațiile To-Do List în prezent 8](#_Toc590369726)

[6. Locul proiectului actual în acest context 8](#_Toc801568449)

[2. Tehnologii Folosite 9](#_Toc945869878)

[2.1 Limbajul de programare C++ 9](#_Toc1283748932)

[2.2 Biblioteca standard C++ (STL) 9](#_Toc1498160322)

[2.3 API Windows (WinAPI) 10](#_Toc1203213659)

[2.4 Lucrul cu fișiere 10](#_Toc1630183369)

[2.5 Mediu de dezvoltare 10](#_Toc119470457)

[2.6 Observații privind simplitatea și extensibilitatea 11](#_Toc1167776727)

[3. Studiu de caz 11](#_Toc805107029)

[3.1 Tema aleasă 11](#_Toc230510412)

[3.2 Variante de implementare 11](#_Toc762023193)

[3.3 Prezentarea detaliată a aplicației cu interfață grafică 12](#_Toc1074310574)

[3.4 Fluxul de lucru – pas cu pas 17](#_Toc1882277590)

[3.5 Beneficii și limitări 17](#_Toc1606599667)

[4.Concluzie 17](#_Toc470446000)

[Bibliografie 18](#_Toc1916818101)

# **1. Introducere**

## **Motivarea Alegerii Temei**

Tema aleasă pentru acest proiect, To-Do List, răspunde unei nevoi cotidiene și universale: gestionarea eficientă a sarcinilor. Indiferent de vârstă sau ocupație, cu toții avem activități și responsabilități pe care trebuie să le organizăm, iar o aplicație de tip listă de sarcini oferă un mijloc simplu și eficient pentru a face acest lucru.

Am ales această temă deoarece îmbină utilitatea practică cu provocările de programare: de la manipularea șirurilor și colecțiilor de obiecte, până la interacțiunea cu utilizatorul prin interfață grafică. În plus, dezvoltarea unei aplicații To-Do List este un exemplu clasic, dar solid, prin care pot fi exersate concepte fundamentale din programarea orientată pe obiect, gestionarea memoriei, operarea fișierelor și dezvoltarea interfețelor interactive.

Pentru a evidenția progresul tehnic și modularitatea codului, proiectul a fost implementat în două variante:

**-Varianta pentru terminal (linie de comandă)** – o versiune simplificată, bazată pe clase C++ și STL (list, string), care permite adăugarea de sarcini, marcarea lor ca finalizate și afișarea lor într-un format simplu, dar clar.

**-Varianta cu interfață grafică (GUI)** – realizată cu ajutorul Windows API (WinAPI), aduce o experiență vizuală mai apropiată de aplicațiile moderne, cu butoane, listă de selecție, câmp de introducere text și funcții de sortare, ștergere și salvare automată a sarcinilor într-un fișier text persistent.

Această abordare duală oferă atât o experiență completă în lucrul cu C++, cât și o vedere de ansamblu asupra evoluției aplicației de la un nivel de bază la unul orientat spre interacțiune grafică și experiență a utilizatorului.

## **Obiective propuse**

Scopul principal al acestui proiect este dezvoltarea unei aplicații care să permită utilizatorului gestionarea simplă a sarcinilor personale printr-o interfață prietenoasă și funcționalități esențiale. Pentru a atinge acest scop, au fost definite următoarele obiective:

#### **1. Dezvoltarea a două implementări funcționale:**

**-Varianta terminal**: permite introducerea, afișarea și marcarea sarcinilor ca finalizate, folosind clase și STL (lista dublu înlănțuită std::list), precum și o abordare simplă de interacțiune cu utilizatorul.

**-Varianta GUI**: permite interacțiunea prin intermediul unei ferestre, oferind funcții de adăugare, ștergere, sortare și salvare automată în fișier (tasks.txt), folosind funcționalități din Windows API (controale EDIT, LISTBOX, BUTTON, gestionarea mesajelor prin WindowProcedure).

#### **2. Organizarea modulară a codului:**

-Separarea logicii aplicației în funcții clare: AddTask, DeleteTask, SortTasks, LoadTasks, SaveTasks.

-Folosirea unei clase TodoItem în versiunea de consolă pentru modelarea unei sarcini ca obiect, cu atribute precum ID, descriere și stare (completed).

#### **3. Persistența datelor:**

-Implementarea mecanismului de salvare automată a sarcinilor într-un fișier text în varianta GUI, pentru a păstra progresul utilizatorului între sesiuni.

#### **4. Interfață intuitivă:**

-În versiunea terminal, meniul oferă opțiuni clare: adăugare sarcină, marcarea ca „finalizat”, ieșire.

-În versiunea grafică, controalele sunt distribuite logic: câmp de text pentru sarcină, butoane de acțiune și listă vizibilă cu sarcinile.

#### **5. Testarea și validarea aplicației:**

-Aplicația a fost testată pentru comportamente frecvente și potențiale erori: inserarea de date invalide, selectarea elementelor inexistente, sortarea sarcinilor, salvarea multiplă etc.

-Codul a fost scris cu atenție la claritate, simplitate și robustețe.

#### **6. Posibilități de extindere ulterioară:**

-Introducerea de priorități pentru sarcini;

-Salvarea într-un format structurat (JSON/XML);

-Implementarea unei interfețe grafice moderne folosind Qt sau un framework cross-platform;

-Suport pentru notificări sau deadline-uri.

Evoluția Pieței

Aplicațiile de tip *To-Do List* au evoluat din simple liste scrise pe hârtie în soluții digitale complexe, integrate în ecosisteme de productivitate, management de proiect și organizare personală. Popularitatea acestor aplicații reflectă o tendință tot mai accentuată spre organizare și eficiență, atât în plan profesional, cât și personal.

### **1. Origini și evoluție timpurie**

Primele forme digitale ale listelor de sarcini au apărut odată cu dezvoltarea primelor organizatoare electronice (PDA-uri) și a software-ului de birou în anii 1980-1990. Programe precum **Microsoft Outlook** sau **Lotus Organizer** permiteau deja gestionarea rudimentară a sarcinilor și notițelor.

Cu apariția sistemelor de operare moderne și a internetului, aplicațiile standalone pentru gestionarea sarcinilor au început să se dezvolte accelerat, devenind aplicații cross-platform.

### **2. Perioada aplicațiilor mobile și cloud (2008 – prezent)**

Odată cu lansarea smartphone-urilor și a magazinelor de aplicații (App Store, Google Play), aplicațiile de tip To-Do List au cunoscut o creștere masivă în popularitate. Aplicații precum:

**Todoist** (lansată în 2007),

**Microsoft To Do** (succesorul Wunderlist),

**Any.do**,

**Trello** (axat pe sarcini vizuale în echipă),

**Google Tasks** și **Notion**

au devenit extrem de populare, fiecare venind cu funcționalități precum:

-sincronizare automată între dispozitive,

-notificări și memento-uri,

-organizare pe proiecte și etichete,

-colaborare în echipă,

-integrare cu aplicații terțe (calendar, email, Slack, etc.).

### **3. Tendințe actuale în dezvoltarea aplicațiilor To-Do**

Piața aplicațiilor To-Do List a evoluat dintr-un segment de nișă într-un domeniu larg, competitiv și diversificat. Câteva direcții recente de evoluție includ:

**-Integrarea AI**: algoritmi care sugerează sarcini, detectează priorități, oferă analize despre obiceiuri de lucru.

**-Gamificare**: aplicații precum **Habitica** motivează utilizatorii prin elemente de joc (puncte, niveluri, recompense).

**-Focus pe sănătatea mentală și echilibrul viață-muncă**: sarcinile nu mai sunt doar despre eficiență, ci și despre binele personal.

**-Interfețe vocale** și control prin asistenți virtuali (Google Assistant, Siri, Alexa).

**-Design minimalist și UX centrat pe utilizator**, pentru reducerea stresului și supraîncărcării informaționale.

### **4. Valoarea economică și utilizare globală**

Conform rapoartelor de piață, segmentul aplicațiilor de productivitate, în care se încadrează și cele de tip To-Do List, este estimat la miliarde de dolari anual. Aplicațiile cele mai cunoscute au milioane de utilizatori activi zilnic.

**Microsoft To Do**, de exemplu, este integrat direct în pachetul Microsoft 365, fiind folosit de companii din întreaga lume, iar **Todoist** a raportat peste 30 de milioane de utilizatori înregistrați în 2024.

### **5. De ce sunt importante aplicațiile To-Do List în prezent**

Într-o lume tot mai aglomerată și digitalizată, gestionarea eficientă a timpului și a sarcinilor a devenit esențială. Aplicațiile de tip To-Do List oferă:

-Claritate mentală,

-Reducerea anxietății legate de uitare,

-Un sentiment de control și realizare,

-Planificare zilnică, săptămânală sau pe termen lung,

-Suport pentru multitasking și lucrul colaborativ.

### **6. Locul proiectului actual în acest context**

Proiectul de față, deși la o scară mai mică și realizat cu scop educativ, se înscrie în acest trend general. Prin cele două variante implementate – una cu interfață grafică și alta în terminal – el reflectă bazele funcționale ale acestor aplicații și demonstrează cum un software simplu poate oferi o experiență practică de organizare și automatizare.

# 2. Tehnologii Folosite

Pentru realizarea acestui proiect am ales limbajul de programare **C++**, alături de două medii distincte de afișare: una bazată pe **interacțiunea în consolă (terminal)** și alta bazată pe **interfață grafică (GUI)**, utilizând **WinAPI** – biblioteca nativă pentru crearea aplicațiilor cu interfață în sistemul de operare Windows.

### **2.1 Limbajul de programare C++**

C++ este un limbaj de programare compilat, performant, orientat pe obiecte, utilizat pe scară largă în dezvoltarea de aplicații desktop, sisteme embedded, jocuri și software industrial. Alegerea acestui limbaj a fost motivată de mai multe considerente:

-Familiarizarea cu paradigma orientată pe obiect, prin clase, metode și instanțe;

-Accesul direct la sistemul de operare, resursele hardware și bibliotecile native;

-Control ridicat asupra memoriei și performanței;

-Suport pentru biblioteci externe (cum este WinAPI);

-Popularitate în educația informatică și dezvoltarea algoritmilor de bază.

Prin utilizarea C++, am putut crea o structură clară a aplicației, organizând fiecare funcționalitate (adăugare, ștergere, marcare ca finalizată, sortare) în metode specifice, și definind o clasă TodoItem pentru reprezentarea fiecărei sarcini.

### **2.2 Biblioteca standard C++ (STL)**

Aplicația cu interfață în consolă a utilizat componente din STL – *Standard Template Library*, mai exact:

**<iostream>**: pentru afișare și citire în/din consolă;

**<string>**: pentru lucrul cu șiruri de caractere;

**<list>**: pentru stocarea dinamică a sarcinilor (cu posibilitate de adăugare și ștergere rapidă);

**<ctime>** și **<cstdlib>**: pentru generarea unor ID-uri pseudo-aleatoare ale sarcinilor.

Această abordare demonstrează flexibilitatea STL în organizarea datelor și facilitarea manipulării colecțiilor de obiecte.

### **2.3 API Windows (WinAPI)**

Pentru varianta cu interfață grafică, am utilizat **WinAPI** – interfața de programare nativă a sistemului Windows. Aceasta oferă control direct asupra ferestrelor, controalelor UI, evenimentelor și mesajelor din sistemul de operare.

Aplicația GUI include:

**Crearea de ferestre și controale** (CreateWindow, CreateWindowEx);

**Gestionarea evenimentelor prin mesaje Windows** (WindowProcedure, WM\_COMMAND, WM\_CREATE, etc.);

**Interacțiunea cu controale UI** precum EDIT, BUTTON, LISTBOX;

**Manipularea persistentă a datelor prin fișiere text (fstream)**.

Această variantă oferă o experiență de utilizare mai prietenoasă și interactivă, folosind doar funcționalitățile native Windows, fără biblioteci externe suplimentare.

### **2.4 Lucrul cu fișiere**

Persistența datelor (lista de sarcini) este realizată prin fișiere text simple (tasks.txt), utilizând bibliotecile:

**<fstream>** – pentru scriere și citire în fișiere;

**std::ofstream / std::ifstream** – pentru scrierea sarcinilor salvate și reîncărcarea acestora la pornirea aplicației GUI.

Această metodă asigură o stocare minimă, dar funcțională, a datelor utilizatorului între sesiuni.

### **2.5 Mediu de dezvoltare**

Proiectul a fost dezvoltat într-un mediu de programare standard pentru C++:

**IDE**: Code::Blocks / Visual Studio (în funcție de platformă);

**Sistem de operare**: Windows 10/11;

**Compilator**: MinGW / MSVC (pentru suport WinAPI și compilare C++).

### **2.6 Observații privind simplitatea și extensibilitatea**

Deși tehnologiile utilizate sunt de bază și nu includ framework-uri moderne sau baze de date, structura aplicației permite extinderea cu ușurință în viitor. Posibile direcții includ:

Migrarea spre o interfață multiplatformă (folosind Qt sau wxWidgets);

Persistență mai complexă (JSON, SQLite);

Salvarea stării aplicației în cloud sau partajare prin conturi de utilizator;

Implementarea unei versiuni mobile (Android/iOS) pe baza conceptului dezvoltat.

# **3. Studiu de caz**

### **3.1 Tema aleasă**

Tema proiectului este realizarea unei aplicații de tip **To-Do List**, care are ca scop gestionarea eficientă a sarcinilor personale sau profesionale. Utilizatorul poate introduce sarcini noi, le poate marca drept finalizate, le poate șterge sau sorta, iar aplicația oferă un mod simplu de a menține evidența lucrurilor de făcut.

Această temă este una relevantă în contextul actual, fiind frecvent întâlnită sub forma unor aplicații mobile sau desktop utilizate zilnic de milioane de oameni. Proiectul vizează în primul rând înțelegerea și implementarea conceptelor de bază ale programării orientate pe obiect și ale interacțiunii cu interfața grafică în limbajul C++.

### **3.2 Variante de implementare**

Aplicația a fost dezvoltată în două variante:

**-Varianta 1 – Consolă (CLI)**: Implementarea de bază, care rulează în terminal și permite adăugarea, afișarea și marcarea sarcinilor ca fiind finalizate. Folosește STL (lista dublu înlănțuită) pentru stocarea sarcinilor, oferind o interfață simplă de tip text. Această variantă a fost utilă pentru prototipare și testarea logicii aplicației.

**-Varianta 2 – Interfață Grafică (GUI)**: Implementarea principală a proiectului, care folosește WinAPI pentru a crea o fereastră grafică, unde utilizatorul poate interacționa prin controale vizuale (textbox, butoane, listbox). Aceasta este varianta prezentată în detaliu în continuare.

### **3.3 Prezentarea detaliată a aplicației cu interfață grafică**

#### **a) Funcționalități oferite**

Aplicația permite următoarele operații de bază:

**Adăugarea unei sarcini** prin intermediul unui câmp de text și a unui buton „Add”;

**Ștergerea unei sarcini selectate** din listă;

**Sortarea sarcinilor** în ordine alfabetică;

**Salvarea automată** a sarcinilor în fișier la închiderea aplicației;

**Încărcarea automată** a sarcinilor din fișier la deschiderea aplicației.

#### **b) Interfața utilizatorului (UI)**

Interfața este simplă și intuitivă, compusă din:

**Un câmp de introducere a textului** (textbox);

**Trei butoane**:

„Add” – pentru adăugarea unei sarcini,

„Delete” – pentru ștergerea sarcinii selectate,

„Sort” – pentru sortarea întregii liste;

**Un listbox** – afișează toate sarcinile curente;

Etichetă statică: „Task:” – indică funcția câmpului de text.

#### **c) Structura logică a aplicației (cu fragmente de cod)**

Structura aplicației este organizată clar, pe componente funcționale, după cum urmează:

#### **1. Crearea controalelor în fereastră – butoane și listă**

Toate elementele grafice sunt create în cadrul mesajului WM\_CREATE, în funcția WindowProcedure. Mai jos este fragmentul de cod care creează cele trei butoane (Add, Delete, Sort), precum și textbox-ul și listbox-ul:

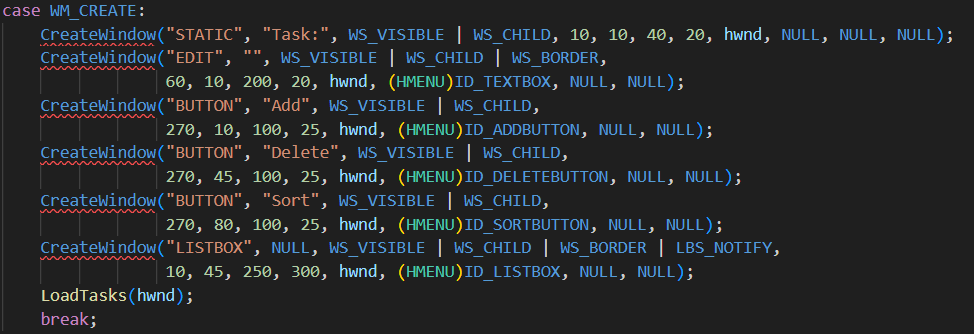


Fig.3.1. Cod pentru crearea butoanelor.

#### **2. Tratarea acțiunilor butoanelor (Adăugare, Ștergere, Sortare)**

Operațiile declanșate de butoane sunt tratate în mesajul WM\_COMMAND:

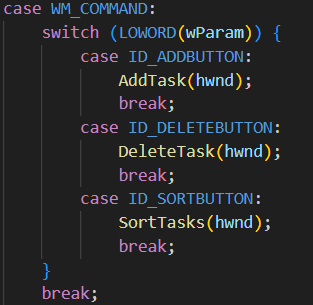


Fig.3.2. Cod pentru declansarea butoanelor.

#### **3. Funcția AddTask – Adăugarea unei sarcini în listă**

Această funcție preia textul introdus în textbox și îl adaugă în listbox:

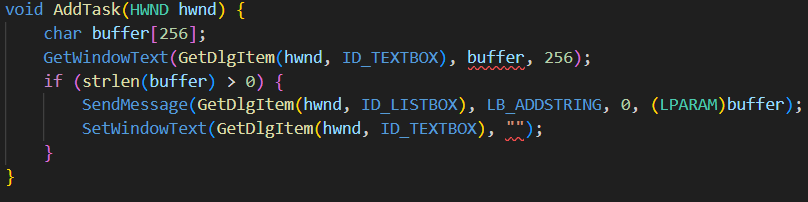
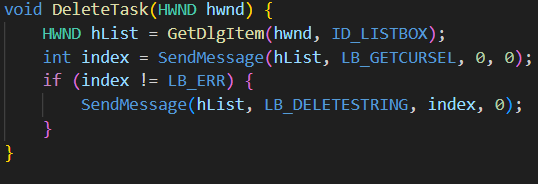


Fig.3.3. Functia AddTask.

**4. Funcția DeleteTask – Ștergerea sarcinii selectate**

Această funcție șterge sarcina selectată de utilizator din listă:

  
 Fig.3.4. Functia DeleteTask.

#### **5. Funcția SortTasks – Sortarea sarcinilor în listă**

Această funcție citește toate sarcinile, le sortează și le reafișează:

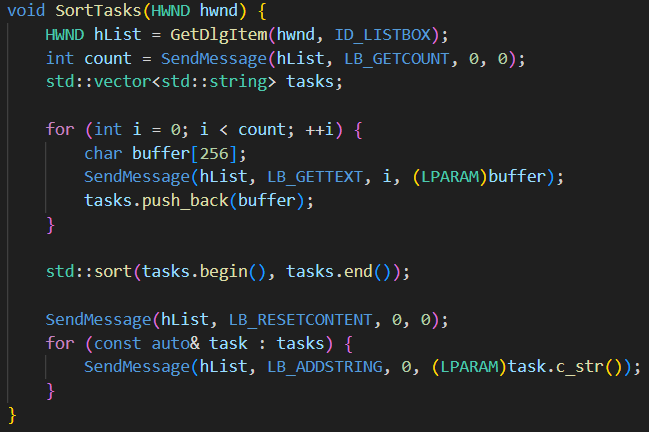
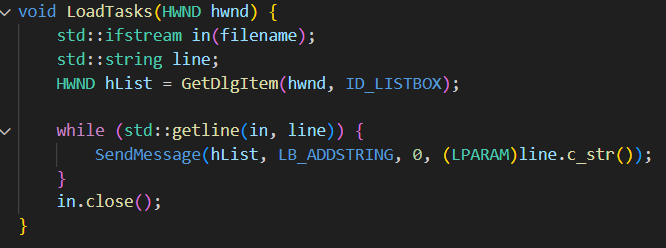


Fig.3.5. Functia Sort Task.

#### **6. Persistența datelor – Salvare și încărcare**

Fișierul tasks.txt este folosit pentru salvarea și încărcarea datelor, cu funcțiile:

 Fig.3.6. Incarcarea datelor.

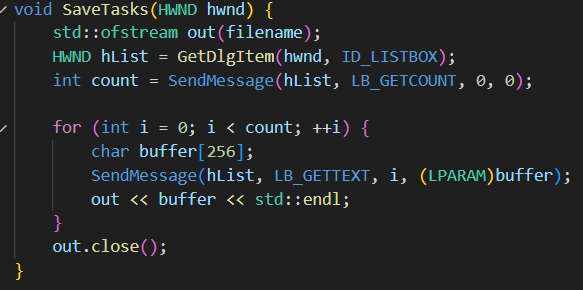


Fig.3.7. Salvarea datelor.

### **3.4 Fluxul de lucru – pas cu pas**

1. **Lansarea aplicației**
   1. Se deschide o fereastră nouă.
   2. Sunt încărcate automat sarcinile din fișierul tasks.txt.
2. **Adăugarea unei sarcini**
   1. Utilizatorul scrie un text în câmpul „Task:”.
   2. Apasă butonul „Add”.
   3. Sarcina este adăugată în listbox și vizibilă imediat.
3. **Ștergerea unei sarcini**
   1. Utilizatorul selectează o sarcină din listă.
   2. Apasă butonul „Delete”.
   3. Elementul este eliminat din listbox.
4. **Sortarea sarcinilor**
   1. Apăsarea butonului „Sort” sortează toate sarcinile afișate în ordine alfabetică.
5. **Închiderea aplicației**
   1. La închiderea ferestrei, aplicația salvează toate sarcinile rămase în tasks.txt.

### **3.5 Beneficii și limitări**

#### **Beneficii:**

-Interfață grafică ușor de folosit;

-Persitență a datelor fără baze de date;

-Cod organizat, modular;

-Aplicație mică, portabilă, fără dependențe externe.

#### **Limitări:**

-Nu există funcționalități avansate precum căutare, categorii, alerte sau deadline-uri;

-Interfața nu este redimensionabilă sau personalizabilă;

-Salvarea se face în fișier text, fără criptare sau validare a datelor.

# **4.Concluzie**

Proiectul To-Do List a fost o oportunitate reală de a aplica cunoștințele de programare în C++ atât la nivel de console interface, cât și la nivel de GUI cu WinAPI. Tema aleasă este practică și aplicația a fost capabilă să implementeze adăugare, ștergere, salvare și informații listate ale sarcinilor.

Finalizând acest proiect, am achiziționat cunoștințe relevante de programming orientat pe obiect, manipularea fișierelor, colecții și delatarea cu graficele. Acest proiect constituie o bună fundație pentru software-urile viitoare, precum și pentru corroborarea design-ului atractiv cu funcționalitatea precisă.

# **Bibliografie**

1. <https://www.youtube.com/watch?v=D17rZyd9r18&ab_channel=ErikYuzwa>
2. <https://www.youtube.com/watch?v=MPLfOoRUKMU&t=2875s&ab_channel=OttoBotCode>
3. <https://www.youtube.com/watch?v=urIpZnCTeKw&ab_channel=OttoBotCode>