****

**MINISTERUL EDUCAȚIEI, CULTURII ȘI CERCETĂRII**

**AL REPUBLICII MOLDOVA**

**Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare, Informatică şi Microelectronică**

**Departamentul Informatică şi Ingineria Sistemelor**

**Calancea Catalin**

**MI-222**

**Raport**

**pentru lucrarea de laborator Nr.3**

***la cursul de “Programarea Orientată pe Obiect”***

Verificat:

lector universitar

Buldumac Oleg

**Chișinău – 20****23**

**CUPRINS**

[1.INTRODUCERE 3](#_Toc149568596)

[2.SARCINA 4](#_Toc149568597)

[3.CONCLUZIA 5](#_Toc149568598)

[4.WEBOGRAFIE 6](#_Toc149568599)

[5.ANEXĂ 7](#_Toc149568600)

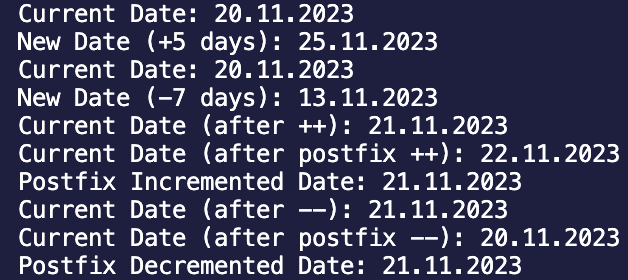
# 1.INTRODUCERE

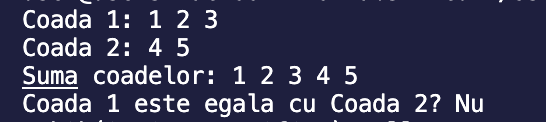
# Lucrarea data se concentrează pe un aspect esențial al programării orientate pe obiect: supraîncărcarea operatorilor. Acest concept oferă dezvoltatorilor posibilitatea de a redefine și adapta comportamentul operatorilor în funcție de nevoile specifice ale claselor pe care le definesc. Scopul principal al acestei lucrări de laborator este de a explora necesitatea și modul în care se realizează supraîncărcarea operatorilor, precum și de a înțelege sintaxa și tipurile de operatori implicate.Deși utilizarea operatorilor aduce beneficii evidente, cum ar fi reducerea lungimii codului, aduce cu sine și un dezavantaj. Claritatea codului poate fi compromisă atunci când este dificil să se distingă între utilizarea unui operator predefinit și a celui supraîncărcat. Cu toate acestea, supraîncărcarea operatorilor devine esențială în rezolvarea unor probleme specifice, precum operatorul de atribuire, unde alternativele nu sunt întotdeauna ideale.

# 2.SARCINA

а) Să se creeze clasa Date – data, care conţine cîmpurile: ziua, luna, anul. Să se definească operatorii "+" şi "-" ca metode ale clasei, iar operatorii "++" şi "--" în ambele variante(prefixă şi postfixă) ca funcţii prietene. Operatorul "+" trebuie să permită realizarea operaţiilor numai cu variabilele de tip predefinit int (x=y+5). De prevăzut prelucrarea corectă a anilor bisecţi.

b) Să se creeze clasa List – coadă. Să se definească operatorii "+" – de adunare a listelor, "=" – de atribuire ca metode ale clasei. Să se definească operatorii de comparare "==", "!=", "<", ">" ca funcţii prietene. Să se supraîncarce operatorii "<<" şi ">>" pentru ieşiri/intrări de obiecte, precum şi pentru inserarea/eliminarea elementelor în/din coadă. Clasa trebuie să fie absolut funcţională, adică să conţină toţi constructorii necesari şi destructorul. Pentru simplificarea lucrului să se utilizeze clasa sau structura ListItem pentru reprezentarea elementelor listei spre care referă List.

a) 

b) 

# 3.CONCLUZIA

În finalul acestei lucrări de laborator, putem trage câteva concluzii importante cu privire la supraîncărcarea operatorilor și impactul acesteia în dezvoltarea software:

1. Flexibilitate și Adaptabilitate: Supraîncărcarea operatorilor adaugă un nivel suplimentar de flexibilitate în programare, permițând dezvoltatorilor să adapteze comportamentul operatorilor la nevoile specifice ale claselor și structurilor de date.

2. Complexitatea versus Claritatea: Deși supraîncărcarea operatorilor poate contribui la complexitatea codului, acesta devine o alegere esențială atunci când soluțiile convenționale, cum ar fi operatorii predefiniți, nu sunt suficiente. Este important să se găsească un echilibru între eficiență și claritate pentru a menține codul ușor de înțeles și de întreținut.

3. Reducerea Codului: Utilizarea corectă a supraîncărcării operatorilor poate duce la o reducere semnificativă a lungimii codului, facilitând scrierea și citirea acestuia. Aceasta poate îmbunătăți semnificativ productivitatea dezvoltatorilor și reduce riscul de introducere a erorilor.

4. Necesitatea Înțelegerii Profunde: Pentru a evita ambiguitățile și erorile de interpretare, este esențială o înțelegere profundă a sintaxei de supraîncărcare a operatorilor și a modului în care aceasta interacționează cu tipurile de date.

În concluzie, supraîncărcarea operatorilor reprezintă o unealtă puternică în arsenalul programatorului, oferind o abordare elegantă și personalizată în manipularea datelor și a obiectelor în cadrul limbajelor de programare orientate pe obiect. Cu atenție și înțelegere, acest concept poate contribui semnificativ la dezvoltarea unei baze de cod robuste și eficiente.

# 4.BIBLIOGRAFIE

1. Stroustrup, B. (2013). "The C++ Programming Language," 4th Edition. Addison-Wesley Professional. - Această carte oferă o acoperire exhaustivă a limbajului de programare C++, inclusiv constructori și destructori.

2. Lippman, S. B., Lajoie, J., & Moo, B. E. (2012). "C++ Primer," 5th Edition. Addison-Wesley. - Cartea "C++ Primer" este o resursă excelentă pentru a înțelege conceptele fundamentale ale C++, inclusiv constructorii și destructorii.

3. Liberty, J., & Cadenhead, R. (2015). "Sams Teach Yourself C++ in One Hour a Day," 7th Edition. Sams Publishing. - Această carte oferă instrucțiuni pas cu pas pentru învățarea C++ și acoperă constructorii și destructorii într-un stil accesibil.

4. Horstmann, C. S. (2016). "Big C++," 2nd Edition. Wiley. - Acest manual acoperă programarea C++ în contextul dezvoltării de software și oferă informații despre constructori și destructori.

5. Deitel, P., Deitel, H., & Deitel, A. M. (2017). "C++ How to Program," 10th Edition. Pearson. - Această carte este un ghid detaliat pentru programarea C++ și include capitole dedicate constructorilor și destructorilor.

6. Strachan, J. (2013). "Beginning C# Object-Oriented Programming." Apress. - Dacă doriți să învățați despre constructori și destructori în limbajul C#, această carte este o sursă valoroasă.

7. Troelsen, A. (2012). "Pro C# 5.0 and the .NET 4.5 Framework." Apress. - Această carte oferă o prezentare detaliată a programării C# și acoperă utilizarea constructorilor și destructorilor în acest limbaj.

# 5.ANEXĂ

Codul sursa :

Sarcina a ) :

<https://github.com/KetSchnaider/Anul2/blob/main/POO/Lab3/lab3a.cpp>

Sarcina b) :

<https://github.com/KetSchnaider/Anul2/blob/main/POO/Lab3/Lab3b.cpp>