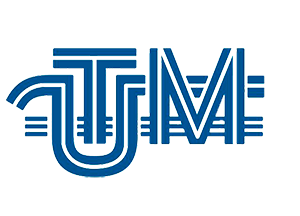
**MINISTERUL EDUCAŢIEI, CULTURII ȘI CERCETĂRII AL REPUBLICII MOLDOVA   
UNIVERSITATEA TEHNICĂ A MOLDOVEI   
Facultatea Calculatoare, Informatică şi Microelectronică**

**Zolotoi Veaceslav**

**Proiect de cercetare: ATESTAREA I**

**Analiza descriptivă și comparată a domeniilor de programare orientată pe obiecte și programare logică**

Verificat:  
**Burlacu Natalia,** *doctor, conf, univ.*Departamentul Informatică și IS,  
Facultatea FCIM, UTM

**Chișinău - 2022**

**CUPRINS:**

CUPRINS……………………………………………………………………………………… 2  
INTRODUCERE ……………………………………………………………………………… 3

1. ANALIZA PARADIGMEI PROGRAMĂRII LOGICE ………………………………….. 4
   1. INFORMAȚIE GENERALĂ ………………………………………………………….4
   2. PRINCIPALELE CARACTERISTICI A PROGRAMARII LOGICE ……………….. 5
   3. STRUCTURA UNUI PROGRAM BAZAT PE PARADIGMA LOGICĂ …………….5
2. ANALIZA PARADIGMEI PROGRAMĂRII ORIENTATE PE OBIECTE ……….………7  
   2.1 INFORMAȚIE GENERALĂ …………………………………………………………..7

2.2 PRINCIPALELE PARADIGME A POO ………………………………………………8

2.3 PRINCIPALELE PLUSURI A POO ……………………………………………………9

2.4 STRUCTURA UNUI POO PROGRAM ………………………………………………10

CONCLUZII ……………………………………………………………………………………12

BIBLIOGRAFII …………………..…………………………………………………………….13

**INTRODUCERE**

Programarea este una dintre cele mai importante aspecte în lumea tehnologiei și este folosită în toate aspectele vieții noastre moderne. De la telefoane mobile la mașini, de la site-uri web la aplicații de afaceri.

Programarea este utilizată pentru a crea software care face viața noastră mai ușoară și mai eficientă. Programarea poate ajuta la crearea de soluții personalizate și adaptabile care pot îmbunătăți eficiența, productivitatea și rentabilitatea companiilor. De asemenea, programarea poate fi utilizată pentru a crea noi produse și servicii care pot satisface nevoile și cerințele în schimbare ale clienților.

Acum programarea începe a fi împărțită nu doar pe limbaje sau domenii de utilizare, ci și cu ajutorul paradigmelor de programare. Despre paradigmele de programare se poate spune că sunt abordări diferite în dezvoltarea de software, care sunt concepute pentru a oferi dezvoltatorilor un set de reguli, concepte și practici în programare. Fiecare paradigma se concentrează pe o anumită abordare în dezvoltarea de software, cu avantaje și dezavantaje specifice.

În această lucrare va fi comparată paradigma programării orientate pe obiecte și paradigma programării logice, care acum sunt unele din cele mai populare și folosite paradigme.

1. **ANALIZA PARADIGMEI PROGRAMĂRII LOGICE**

**1.1. INFORMAȚIE GENERALĂ**

Programarea logică este o paradigma de programare care se concentrează pe definirea relațiilor logice între diferite variabile și reguli de interferență care leagă aceste variabile. Această paradigmă utilizează logica matematică și filosofică pentru a defini reguli și constrângeri care trebuie respectate de către program.

În programarea logică, codul este scris folosind un limbaj de programare logică, cum ar fi Prolog sau Haskell, care se bazează pe teoria logică și calculele de predicat. În acest tip de programare, programatorii nu trebuie să specifice pașii individuali pe care trebuie să îi urmeze programul, ci să specifice condițiile necesare pentru ca programul să ajungă la o anumită concluzie.

Programarea logică poate fi utilizată în diferite domenii, cum ar fi inteligența artificială, analiza de date și verificarea formală. De exemplu, programarea logică poate fi folosită pentru a dezvolta sisteme expert care pot rezolva probleme complexe prin aplicarea unor reguli și cunoștințe stocate în bazele de date. În analiza de date, programarea logică poate fi utilizată pentru a defini reguli de interferență care pot ajuta la identificarea modelelor în datele analizate.

În general, programarea logică se concentrează pe relații logice și constrângeri, în loc de instrucțiuni și flux de control, ceea ce face ca această paradigma să fie potrivită pentru problemele complexe care nu pot fi rezolvate cu ușurință prin programarea procedurală sau orientată pe obiecte.



Figura 1.1 Reprezentarea schematică a programării logice

**1.2 PRINCIPALELE CARACTERISTICI A PROGRAMARII LOGICE**

Programarea logică se bazează pe o serie de caracteristici cheie care o fac o paradigmă de programare distinctă. Aceste caracteristici includ:

1. Utilizarea logicii matematice: Programarea logică se bazează pe logica matematică și calculele de predicat pentru a defini regulile și constrângerile programului.
2. Definirea relațiilor logice: Programarea logică se concentrează pe definirea relațiilor logice între diferite variabile și reguli de interferență care leagă aceste variabile.
3. Utilizarea unui limbaj de programare logică: Programarea logică utilizează un limbaj de programare logică, cum ar fi Prolog sau Haskell, care se bazează pe teoria logică și calculele de predicat.
4. Lipsa instrucțiunilor și fluxurilor de control: În programarea logică, programatorii nu trebuie să specifice pașii individuali pe care trebuie să îi urmeze programul, ci să specifice condițiile necesare pentru ca programul să ajungă la o anumită concluzie.
5. Orientarea spre probleme complexe: Programarea logică este potrivită pentru problemele complexe care nu pot fi rezolvate cu ușurință prin programarea procedurală sau orientată pe obiecte.
6. Utilizarea pentru inteligență artificială și analiză de date: Programarea logică poate fi utilizată în diferite domenii, cum ar fi inteligența artificială, analiza de date și verificarea formală.

În general, programarea logică oferă o modalitate diferită de abordare a programării, prin utilizarea logicii matematice și a relațiilor logice pentru a defini regulile și constrângerile programului. Această paradigmă poate fi folosită pentru a rezolva probleme complexe și poate fi adaptată la diferite domenii, cum ar fi inteligența artificială și analiza de date.  
  
**1.3 STRUCTURA UNUI PROGRAM BAZAT PE PARADIGMA LOGICĂ**

Un program scris cu ajutorul paradigmei logice are o structură diferită față de cele scrise cu ajutorul paradigmei OOP. Structura unui program scris cu ajutorul paradigmei logice poate fi împărțită în următoarele părți:

1. Bază de cunoștințe: Partea programului care conține faptele și regulile care descriu problemele și informațiile disponibile. Baza de cunoștințe este scrisă utilizând sintaxa limbajului de programare logică și conține informații precum tipuri de date, relații între acestea și reguli de interferență.
2. Motorul de interferență: Acesta este motorul care citește baza de cunoștințe și utilizează regulile de interferență pentru a deduce informații noi și a rezolva problemele. Motorul de interferență poate fi gândit ca o componentă care folosește logica programului pentru a lua decizii.
3. Interfața cu utilizatorul: Aceasta este partea programului care permite utilizatorilor să interacționeze cu baza de cunoștințe și motorul de interferență. Interfața cu utilizatorul poate fi o interfață grafică sau o interfață de linie de comandă.
4. Alte componente: Programul scris cu ajutorul paradigmei logice poate include și alte componente, precum module pentru procesarea datelor de intrare și ieșire, biblioteci de funcții sau componente pentru conectarea la alte sisteme sau baze de date.

În general, un program scris cu ajutorul paradigmei logice are o structură modulară și este format din componente interconectate care permit utilizatorilor să interacționeze cu motorul de inferență și să primească rezultate într-un mod intuitiv. Programul este scris utilizând un limbaj de programare logică și este proiectat pentru a utiliza logica și regulile pentru a rezolva problemele și a deduce noi informații.



Figura 1.2 Diferența structurii unui program scris cu ajutorul paradigmei logice și paradigmei funcționale

**2. ANALIZA PARADIGMEI PROGRAMĂRII ORIENTATE PE OBIECTE**

**2.1 INFORMAȚIE GENERALĂ**

Programarea orientată pe obiecte (POO) este o paradigmă de programare care se concentrează pe organizarea programelor în obiecte sau entități care interacționează între ele. În POO, un program este format din obiecte care pot fi considerate ca fiind unități autonome, fiecare cu propriile caracteristici și comportamente.

O caracteristică cheie a POO este incapsularea, care se referă la capacitatea de a proteja datele și comportamentul obiectului într-un pachet sigur, accesibil numai prin metodele publice ale obiectului. Aceasta face ca obiectele să fie independente și ușor de utilizat, ceea ce permite dezvoltatorilor să își concentreze eforturile asupra obiectelor individuale, fără a fi nevoie să se preocupe de detalii de implementare.

În POO, obiectele pot fi create și manipulate prin intermediul mesajelor, care sunt trimise între obiecte și care declanșează comportamente specifice. De asemenea, POO permite definirea relațiilor între obiecte, cum ar fi moștenirea, polimorfismul și agregarea, ceea ce permite dezvoltatorilor să creeze programe complexe și structurate.

Limbajele de programare orientate pe obiecte, cum ar fi Java, C++, Python, JavaScript și Ruby, sunt utilizate frecvent în dezvoltarea de software, în special pentru dezvoltarea aplicațiilor web, mobile și desktop. POO a devenit o paradigmă de programare populară datorită abilității sale de a permite reutilizarea codului, organizarea programelor în mod logic și ușurința de utilizare a obiectelor.

**2.2 PRINCIPALELE PARADIGME A POO**

Paradigmele de programare orientate pe obiecte (POO) sunt modele de gândire și dezvoltare a programelor care se concentrează pe crearea de obiecte sau entități independente care interacționează între ele pentru a rezolva o problemă sau a îndeplini o sarcină specifică. Această paradigmă de programare este larg utilizată în programarea modernă și este utilizată de dezvoltatorii de software pentru a crea aplicații complexe, fiabile și ușor de întreținut.

Moștenirea este una dintre cele mai importante paradigme de programare orientată pe obiecte. Aceasta se referă la capacitatea unui obiect de a fi creat ca o extensie a altui obiect deja existent, moștenind astfel proprietățile și comportamentele sale. Aceasta permite dezvoltatorilor să creeze noi obiecte care au caracteristici similare cu cele ale obiectului moștenit, fără a fi nevoie să rescrie întreaga funcționalitate. În schimb, ei pot adăuga doar modificările și adăugirile necesare.

Polimorfismul este o altă paradigmă importantă de programare orientată pe obiecte. Aceasta se referă la capacitatea unei clase de a fi utilizată pentru a crea obiecte de tipuri diferite, iar aceste obiecte pot fi tratate în mod uniform prin intermediul unei interfețe comune. Acest lucru permite dezvoltatorilor să scrie cod care poate fi utilizat cu obiecte de tipuri diferite, fără a fi nevoie să rescrie codul pentru fiecare tip de obiect.

Incapsularea este o altă paradigmă importantă de programare orientată pe obiecte. Aceasta se referă la capacitatea unui obiect de a ascunde datele și comportamentul său de alte obiecte și de a le face accesibile doar prin intermediul metodelor publice ale obiectului. Acest lucru face obiectele independente și ușor de utilizat și protejează datele și comportamentul obiectului de accesul neautorizat.

Abstracția este o altă paradigmă importantă de programare orientată pe obiecte. Aceasta se referă la capacitatea de a defini o clasă sau o interfață comună pentru un set de obiecte similare, abstractizând astfel caracteristicile lor comune. Acest lucru permite dezvoltatorilor să creeze clase abstracte care să definească o interfață comună pentru mai multe clase care au caracteristici similare, dar care diferă în alte privințe.

Agregarea este o altă paradigmă importantă de programare orientată pe obiecte. Aceasta se referă la capacitatea unei clase de a fi compusă din mai multe obiecte mai mici, fiecare cu propriile caracteristici și comportamente. Acest lucru permite dezvoltatorilor să creeze obiecte complexe prin compunerea mai multor obiecte mai simple, fără a fi nevoie să rescrie întreaga funcționalitate.



Figura 2.2.1 Paradigmele programării orientate pe obiecte

**2.3 PRINCIPALELE PLUSURI A POO**

Paradigma de programare orientată pe obiecte (POO) are mai multe avantaje față de alte paradigme de programare. Acestea includ:

1. Reutilizarea codului - POO permite reutilizarea codului prin intermediul moștenirii și compunerii. Aceasta înseamnă că dezvoltatorii pot crea noi obiecte bazate pe altele deja existente, reducând astfel timpul și efortul necesar pentru a scrie cod nou.
2. Încapsularea și abstractizarea - POO permite încapsularea datelor și a comportamentului obiectelor, astfel încât acestea să nu poată fi accesate sau modificate din exteriorul obiectului. Această caracteristică ajută la prevenirea erorilor și la creșterea securității programelor.
3. Extensibilitatea - POO permite dezvoltatorilor să adauge și să modifice funcționalitatea unui program prin intermediul extinderii obiectelor existente sau prin crearea de noi obiecte. Acest lucru face programul mai flexibil și mai ușor de adaptat la cerințele schimbătoare ale utilizatorilor.
4. Modularitatea - POO permite dezvoltatorilor să împartă programul în module independente, fiecare dintre acestea având o responsabilitate specifică. Această caracteristică ajută la gestionarea mai ușoară a programelor mari și complexe și facilitează testarea și depanarea acestora.
5. Eficiența - POO utilizează tehnicile de programare orientată pe obiecte pentru a îmbunătăți eficiența programelor. Aceasta poate include utilizarea cache-urilor și a metodelor de optimizare a codului pentru a îmbunătăți performanța programului.



Figura 2.4.1 Structura unui program POO

Aceste avantaje fac din POO una dintre cele mai populare și utilizate paradigme de programare din lumea IT.

**2.4 STRUCTURA UNUI POO PROGRAM**

Structura unui program scris cu ajutorul paradigmei de programare orientată pe obiecte (POO) constă în general într-o colecție de obiecte interconectate, care colaborează pentru a realiza o funcționalitate complexă. Fiecare obiect din program are propriile sale date și metode, care sunt încapsulate în obiect, astfel încât acestea să fie accesibile doar prin intermediul interfeței publice a obiectului.

În mod tipic, un program POO este format din clase și obiecte. O clasă reprezintă un tip de obiect și definește datele și metodele care sunt comune pentru toate obiectele de acel tip. Obiectele sunt instanțe ale claselor și reprezintă o copie a datelor și metodelor din clasa respectivă. De exemplu, o clasă "Mașină" ar putea avea variabile de instanță pentru culoare, model și viteză, împreună cu metode pentru accelerare, oprire și întoarcere.

Pentru a crea obiecte în program, se folosește constructorul clasei. Constructorul este o metodă specială care este apelată atunci când se creează un obiect și este responsabil pentru inițializarea variabilelor de instanță ale obiectului.

După ce obiectele sunt create, acestea sunt conectate între ele prin intermediul mesajelor. Mesajele sunt apeluri de metodă care sunt transmise de la un obiect la altul, astfel încât obiectele să poată colabora și să îndeplinească o anumită funcționalitate.

În general, structura unui program POO este modulară și ușor de extins. Obiectele sunt interconectate într-un mod modular, astfel încât modificarea sau adăugarea de funcționalități noi să nu afecteze alte părți ale programului. Acest lucru face programarea orientată pe obiecte o paradigmă populară pentru dezvoltarea de software complex.

**CONCLUZII:**

La finalul lucrării pot spune că programarea se dezvoltă cu o viteză enormă și totdeauna apar noi paradigme de programare, dar totuși la baza rămân cele mai veche, cum ar fi programarea logică și programarea orientată pe obiecte.

Comparând aceste două paradigme a programării, pot spune că ambele paradigme sunt foarte folosite, dar au domeniile sale de aplicare, unde ele au o eficiență maximă și noi, ca programiști, ar trebui să respectăm această deviere pe domenii.

Aceste două paradigme au ale sale plusuri, minusuri și caracteristici, care au fost găsite și explicate pe parcursul lucrării. Anume aceste două paradigme, cred că au cele mai multe diferențe, deoarece ele fac parte din domenii total diferite a programării. Structura unui program scris cu ajutorul paradigmei logice, este full diferit de un program scris cu ajutorul paradigmei orientate pe obiecte.

BIBLIOGRAFII:

1. **[https://ro.wikipedia.org/wiki/Programare\_orientată\_pe\_obiecte](https://ro.wikipedia.org/wiki/Programare_orientat%C4%83_pe_obiecte)**
2. **<https://it.emoldova.org/community/programare-limbaje-logice/programarea-logica/>**
3. **<https://codeberryschool.com/blog/ro/ce-este-programarea-orientata-pe-obiect/>**