****

**MINISTERUL EDUCAȚIEI, CULTURII ȘI CERCETĂRII**

**AL REPUBLICII MOLDOVA**

**Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare, Informatică şi Microelectronică**

**Departamentul Informatică şi Ingineria Sistemelor**

**Ceban Vadim IA - 182**

**Teză de an**

**Tema:”** SE pentru recomandarea profesiilor pentru absolvenții liceelor **“**

### *la cursul de “Sisteme expert"*

Verificat:

**Burlacu Natalia,** *doctor, conf. univ.*

Departamentul Informatică şi IS,

Facultatea FCIM, UTM

**Chișinău – 2021**

Cuprins

[Capitolul I Cercetarea 3](#_Toc65856384)

[Programare logică 4](#_Toc65856385)

[SWI Prolog 6](#_Toc65856386)

[Programare structurată 7](#_Toc65856387)

[Visual Studio 8](#_Toc65856388)

[Limbaje de programare structurată 9](#_Toc65856389)

[Capitolul II Prototipul 11](#_Toc65856390)

[Tema: 11](#_Toc65856391)

[Universul problemei 11](#_Toc65856392)

[Obiectivele: 11](#_Toc65856393)

[Tascurile propuse: 11](#_Toc65856394)

[Explicarea codului: 11](#_Toc65856395)

[Rularea programului. 15](#_Toc65856396)

[Capitolul III Codul sursă 17](#_Toc65856397)

[Concluzii: 32](#_Toc65856398)

[Bibliografie: 32](#_Toc65856399)

**Scopul lucrării**: Efectuarea cercetărilor și analizei a domeniilor de programare structurală și programare logică, de a valorifica cunoștințele și abilitățile de lucru în PROLOG, obținute în cadrul disciplinelor studiate anterior tangente obiectului curent pentru ca și finalitate a produce un prototip a unui baze de cunoștință de transpus într-o aplicație-emulator a unui [SE & SEG](https://else.fcim.utm.md/mod/folder/view.php?id=1030) sub un sistem de operare (Windows, Android, etc.).

# Capitolul I Cercetarea

**Ce este programarea?**

Să presupunem că lucrezi pentru o companie multinațională și că vei avea o întâlnire cu un om de afaceri din China. Limba ta nativă este româna și nu vorbești o boabă de chineză. Totuși, îți aduci aminte că a doua ta limbă este engleza și că omul de afaceri chinez vorbește și engleza. Atunci începi să folosești engleza ca limbă comună, pentru a negocia unul cu celălalt. În același fel, folosim limbajele de programare HTML, CSS, JavaScript, Java etc. pentru a comunica cu calculatorul astfel încât să ne înțelegem corespunzător unul cu celălalt.

Limba nativă a unui calculator este sistemul binar: o listă incredibilă de 1 și 0. Sistemul binar nu este foarte ușor de citit sau scris de către oameni, așa că în schimb am inventat limbajele de programare pc pentru a vorbi cu calculatoarele. Din moment ce calculatoarele nu înțeleg limbajul uman, trebuie să ne întâlnim undeva la mijloc. Totuși, spre deosebire de oameni, calculatoarele nu sunt capabile să înțeleagă o comunicare indirectă și iau mereu lucrurile ad litteram. Dacă le spui ceva, atunci ele vor face exact ceea ce le-ai spus. Calculatoarele nu sunt suficient de inteligente să ia decizii pe cont propriu și nu vor înțelege informațiile care nu sunt scrise precis pentru ele. Din acest motiv, trebuie să te asiguri că le dai instrucțiunile exacte.

Există mai multe limbaje de programare respective între acestea există și mai multe categorii de orientare a programării, aceste categorii sunt numite paradigme. Se cunosc mai multe tipuri de paradigme acestea fiind:

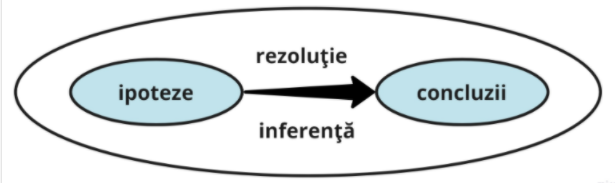
* [Programare declarativă](https://ro.wikipedia.org/w/index.php?title=Programare_declarativ%C4%83&action=edit&redlink=1)
* [Programare funcțională](https://ro.wikipedia.org/wiki/Programare_func%C8%9Bional%C4%83)
* [Programare imperativă](https://ro.wikipedia.org/wiki/Programare_imperativ%C4%83)
* [Programare logică](https://ro.wikipedia.org/w/index.php?title=Programare_logic%C4%83&action=edit&redlink=1)
* [Programarea orientată pe aspecte](https://ro.wikipedia.org/wiki/Programarea_orientat%C4%83_pe_aspecte)
* [Programare orientată eveniment](https://ro.wikipedia.org/wiki/Programare_orientat%C4%83_eveniment)
* [Programare orientată pe obiecte](https://ro.wikipedia.org/wiki/Programare_orientat%C4%83_pe_obiecte)
* [Programare procedurală](https://ro.wikipedia.org/wiki/Programare_procedural%C4%83)
* [Programare structurată](https://ro.wikipedia.org/wiki/Programare_structurat%C4%83)

În cercetarea mea voi atinge două dintre lista aceasta e paradigme acestea fiind: programarea logică și programarea structurată.

## Programare logică

Programarea logică este o metodă pe care informaticienii o folosesc pentru a încerca să permită mașinilor să acționeze, deoarece este utilă pentru reprezentarea cunoștințelor. În programarea logică, logica este utilizată pentru a reprezenta cunoștințele, iar inferența este utilizată pentru a le manipula.

 Logica utilizată pentru a reprezenta cunoștințele în programarea logică este forma clauzală care este un subset al logicii predicate de ordinul întâi. Este folosit deoarece logica de ordinul întâi este bine înțeleasă și capabilă să reprezinte toate problemele de calcul. Cunoașterea este manipulată utilizând sistemul de inferență a rezoluției, care este necesar pentru a demonstra teoremele în logica în formă clauzală. Diagrama de mai jos arată esența programării logice.



Programarea logică se potrivește foarte bine atunci când este aplicată în rezolvarea problemelor din domenii care se ocupă de extragerea cunoștințelor din fapte și relații de bază. Acest tip de programare este bazat pe axiome, reguli de inferență și interogări. Executarea unui astfel de program devine o căutare sistematică întrun set de fapte, folosind un set de reguli de inferență.

Acest tip de programare se bazează pe faptul că, rezolvarea unei problem constă dintr-o reprezentare corespunzătoare a cunoștințelor. Limbajele de programare logică se mai numesc și limbaje declarative. Spre deosebire de limbajele procedural, la care fiecare pas trebuie specificat în detaliu, la limbajele declarative. Se specific cee ace se așteaptă în obținerea soluției (scopul problemei).

Teoria matematică care stă la baza acestei paradigm este logica predicatelor de ordin unu, care lucrează cu entități logice. Logica predicatelor este o logică simbolică al cărui scop este de a reprezenta tipuri de raționament. Atît timp cît calculul predicatelor are reguli și formalități matematice definite printr-o teorie, soluția la o problem specific este complete și efectivă.

**Avantaje:**

* Programarea logică poate fi utilizată pentru a exprima cunoștințele într-un mod care nu depinde de implementare, făcând programele mai flexibile, comprimate și mai ușor de înțeles.
* Permite separarea cunoștințelor de utilizare, adică arhitectura mașinii poate fi schimbată fără a schimba programele sau codul lor de bază.
* Poate fi modificat și extins în moduri naturale pentru a susține forme speciale de cunoaștere, cum ar fi cunoașterea de nivel superior sau de ordin superior.
* Poate fi utilizat în discipline non computaționale, bazându-se pe raționament și mijloace de expresie precise.

Un program scris într-un limbaj de programare logică este constituit din enunţuricare exprimă cunoştinţele relative asupra problemei pe care încearcă să o rezolve programul. Formularea acestor cunoştinţe se bazează pe două concepte de bază:

* Existenţa de obiecte discrete, exprimate prin piese de cunoaştere (cunoştinţedeclarative).
* Existenţa de relaţii între aceste obiecte/enunţuri

**Aplicații ale programării logice**

Calcul simbolic:

* baze de date relaționale,
* logică matematică,
* rezolvare de probleme abstracte,
* întțelegerea limbajului natural,
* rezolvare de ecuatții,
* automatizarea designului de programe,
* inteligentță artiﬁcială,
* analiza structurilor biochimice, etc.

## SWI Prolog

 Prolog (PROgramming in LOGic) este un limbaj de programare declarativ care se bazează pe ideile de programare logică, precum cele discutate mai sus. Prolog a fost creat să arate ca un limbaj de programare și să permită controlarea acesteia de către un programator pentru a avansa cercetarea pentru demonstrarea teoremelor.

În Prolog, logica programului este exprimată în termeni de relații, iar un calcul este inițiat prin executarea unei interogări peste aceste relații. Relațiile și interogările sunt construite folosind tipul unic de date Prolog, termenul .  Relațiile sunt definite prin clauze . Având în vedere o interogare, motorul Prolog încearcă să găsească o [respingere a](https://en.wikipedia.org/wiki/Refutation)[rezoluției](https://en.wikipedia.org/wiki/Resolution_(logic)) interogării negate. Dacă interogarea negată poate fi respinsă, adică se găsește o instanțiere pentru toate variabilele libere care face ca unirea de clauze și setul de singleton constând din interogarea negată să fie falsă, rezultă că interogarea inițială, cu instanțierea găsită aplicată, este o [consecință logică](https://en.wikipedia.org/wiki/Logical_consequence)a programului. Acest lucru face ca Prolog (și alte limbaje de programare logică) să fie deosebit de utile pentru aplicații de baze de date, [matematică simbolică](https://en.wikipedia.org/wiki/Symbolic_mathematics) și analiză a limbajului. Deoarece Prolog permite [predicate](https://en.wikipedia.org/wiki/Predicate_(mathematical_logic)) impure , verificarea [valorii adevărului](https://en.wikipedia.org/wiki/Truth_value) anumitor predicate speciale poate avea un [efect secundar](https://en.wikipedia.org/wiki/Side_effect_(computer_science)) deliberat , cum ar fi imprimarea unei valori pe ecran. Din această cauză, programatorului i se permite să folosească o cantitate de [programare imperativă](https://en.wikipedia.org/wiki/Imperative_programming) convențională atunci când paradigma logică este incomodă. Are un subset pur logic, numit „Prolog pur”, precum și o serie de caracteristici extralogice.

Funcționalitatea de programare a logicii de constrângere a venit destul de târziu în viața SWI-Prolog, deoarece nu avea suportul de bază. Acest lucru s-a schimbat la începutul anului 2004, când variabilele atribuite au fost adăugate limbii. Biblioteca CHR Leuven a fost atunci prima bibliotecă CLP care a fost portată la SWI-Prolog. Menționăm biblioteca INCLP (R) a SWI-Prolog (De Koninck și colab. 2006), care oferă constrângeri neliniare asupra realilor și a fost implementată pe lângă CHR. Mai târziu a venit un port al bibliotecii CLP (QR) a lui Christian Holzbaur și un rezolvator CLP (FD) cu domeniu finit. În cele din urmă, a fost adăugat un rezolvator boolean CLP (B).

SWI-Prolog implementează variabile atribuite (constrângeri) și continuări delimitate după proiectarea în hProlog de către Bart Demoen. Implementarea motorului urmează proiectul propus de Paul Tarau. Tablarea a fost implementată de Benoit Desouter pe baza unor continuări delimitate. Tabelarea a fost extinsă cu preluarea răspunsurilor de către Fabrizio Riguzzi. Implementarea semantica bine fundamentate și depunere elementare urmează XSB și a fost sponsorizat de Kyndi și modul posibil de suport tehnic din special Theresa Swift și David S. Warren.

## Programare structurată

Programarea structurată este o paradigmă de programare care vizează îmbunătățirea clarității, calității și timpului de dezvoltare a unui program de calculator prin utilizarea extensivă a structurilor de flux structurat de control de selecție ( dacă / apoi / altfel ) și repetare (în timp și pentru ), structuri de blocuri , și subrutine . A apărut la sfârșitul anilor 1950, odată cu apariția limbajelor de programare ALGOL 58 și ALGOL 60 , acestea din urmă incluzând suport pentru structuri de bloc. Factorii care contribuie la popularitatea sa și la acceptarea pe scară largă, la început în mediul academic și mai târziu în rândul practicienilor, includ descoperirea a ceea ce este acum cunoscut sub numele de teorema programului structurat în 1966 și publicarea influentei scrisori deschise „ Go To Statement Considered Harmful ” în 1968 de informaticianul olandez Edsger W. Dijkstra , care a inventat termenul „programare structurată”. Programarea structurată este cel mai frecvent utilizată cu abateri care permit programe mai clare în anumite cazuri particulare, cum ar fi atunci când trebuie efectuată gestionarea excepțiilor .

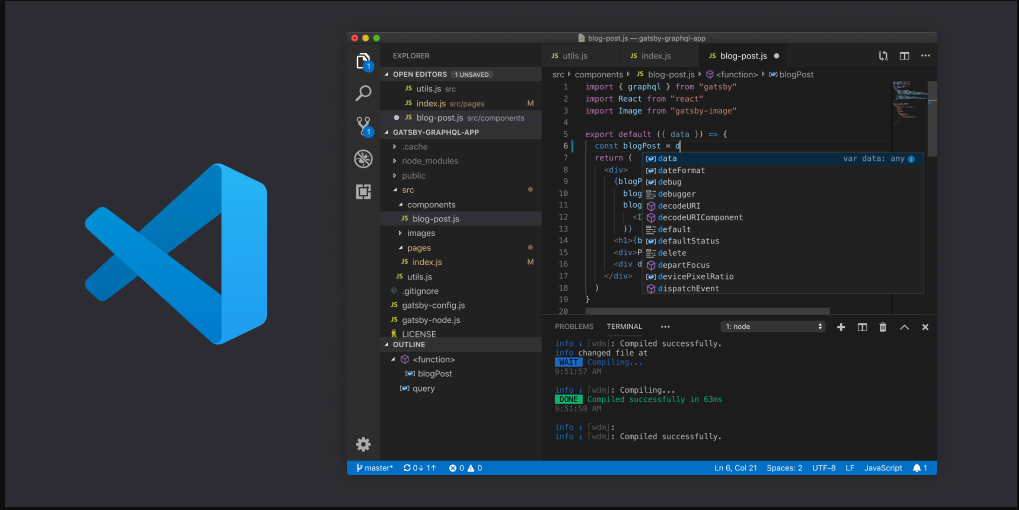
În urma teoremei programate structurate , toate programele sunt văzute ca fiind compuse din structuri de control :

* "Secvenţă"; declarații ordonate sau subrutine executate în ordine.
* "Selecţie"; una sau mai multe instrucțiuni sunt executate în funcție de starea programului. Acest lucru este de obicei exprimat cu cuvinte cheie precum if..then..else..endif . Declarația condițională ar trebui să aibă cel puțin o condiție adevărată și fiecare condiție ar trebui să aibă un punct de ieșire la max.
* "Repetare"; o instrucțiune sau un bloc este executat până când programul atinge o anumită stare sau operațiile au fost aplicate fiecărui element al unei colecții. Acest lucru este de obicei exprimat prin cuvinte cheie , cum ar fi while , repeat , for sau do..until . De multe ori se recomandă ca fiecare buclă să aibă doar un singur punct de intrare (și în programarea structurală originală, de asemenea, un singur punct de ieșire, iar câteva limbaje să impună acest lucru).
* „Recursivitate”; o declarație este executată apelându-se în mod repetat până când sunt îndeplinite condițiile de reziliere. Deși similare în practică cu buclele iterative, buclele recursive pot fi mai eficiente din punct de vedere al calculului și sunt implementate diferit ca o stivă în cascadă.

Creşterea complexităţii aplicaţiilor a impus la începutul anilor `70 apariţia unei noi paradigme în programare : programarea structurată. Scopul era de a dezvolta noi tehnici de programare, care să permită dezvoltarea unor programe fiabile, uşor de elaborat în echipă, uşor de depanat, de întreţinut şi de reutilizat. Sunt cîteva principia existente a programării structurate:

* Un prim principiu al programării strcturare este modulizarea. Pentru proiectarea unor aplicaţii complexe, este necesară descompunerea problemei care trebuie rezolvată în subprobleme relativ independente, pentru fiecare dintre  aceste subprobleme scriindu-se module de program mai simple. Fiecare modul efectuează un set de prelucrări specifice şi este relativ independent de celelalte module, cu care comunică prin intermediul unui set de parametri, care constituie interfaţa. Avantajele sunt multiple. Cum la orice firmă se lucrează în echipă, modulele de program pot fi implementate de mai mulţi programatori. Modificarea unui modul nu afectează celelalte module. Fiecare modul poate fi implementat, testat, depanat, modificat, independent de celelalte.
* Un alt principiu fundamental este structurarea datelor şi a prelucrărilor.  
  Programatorul are posibilitatea de a-şi grupa datele în colecţi, organizarea după anumite reguli, denumite structuri de date.
* Prelucrările asupra datelor sunt structurare separat. Confotm teoremei de structură Böm-Jacopini, orice prelucrare poate fi descrisă prin compunerea a trei structuri fundamentale: strucuta liniară (secvenţială), structura alternativă şi structura repetitive.

## Visual Studio



Un bun mediu de dezvoltare pentru programarea structurată dar și pentru toate direcțiile de programare este Visual Studio.

Visual    Studio    Code    este    un    [editor    de    cod    sursă](https://en.wikipedia.org/wiki/Source-code_editor)  freeware  realizat  de  [Microsoft](https://en.wikipedia.org/wiki/Microsoft)  pentru  [Windows](https://en.wikipedia.org/wiki/Windows)  ,  [Linux](https://en.wikipedia.org/wiki/Linux)  și  [macOS](https://en.wikipedia.org/wiki/MacOS)  .  Funcțiile  includ  suport  pentru  [depanare](https://en.wikipedia.org/wiki/Debugging)  ,  [evidențierea  sintaxei](https://en.wikipedia.org/wiki/Syntax_highlighting)  ,  [completarea  inteligentă  a  codului](https://en.wikipedia.org/wiki/Intelligent_code_completion)  ,  [fragmente](https://en.wikipedia.org/wiki/Snippet_(programming))  ,  [refactorizarea  codului](https://en.wikipedia.org/wiki/Code_refactoring)  și  [Git](https://en.wikipedia.org/wiki/Git)  încorporat  .  Utilizatorii  pot  modifica  [tema](https://en.wikipedia.org/wiki/Theme_(computing))  ,  [comenzile  rapide  de  la  tastatură](https://en.wikipedia.org/wiki/Keyboard_shortcut)  ,  preferințele  și  pot  instala  [extensii](https://en.wikipedia.org/wiki/Plug-in_(computing))  care  adaugă  funcționalități  suplimentare.

Microsoft a lansat codul [sursă](https://en.wikipedia.org/wiki/Source_code) al Visual Studio Code pemicrosoft / vscode( Cod - OSS ) depozit de [GitHub](https://en.wikipedia.org/wiki/GitHub) , sub permisiunea [permis MIT](https://en.wikipedia.org/wiki/MIT_License) , timp ce versiunile de Microsoft sunt [freeware](https://en.wikipedia.org/wiki/Freeware) .

În [Stack Overflow](https://en.wikipedia.org/wiki/Stack_Overflow) 2019 Developer Survey, Visual Studio Code a fost clasat drept cel mai popular instrument de mediu pentru dezvoltatori, 50,7% din 87.317 respondenți au raportat că îl folosesc. Visual Studio Code a fost anunțat pentru prima dată pe 29 aprilie 2015 de Microsoft la conferința [Build](https://en.wikipedia.org/wiki/Build_(developer_conference)) 2015 . O versiune Preview a fost lansată la scurt timp după aceea.

Visual Studio Code este un editor de cod sursă care poate fi utilizat cu o varietate de limbaje de programare, inclusiv [Java](https://en.wikipedia.org/wiki/Java_(programming_language)) , [JavaScript](https://en.wikipedia.org/wiki/JavaScript) , [Go](https://en.wikipedia.org/wiki/Go_(programming_language)) , [Node.js](https://en.wikipedia.org/wiki/Node.js) , [Python](https://en.wikipedia.org/wiki/Python_(programming_language)) și [C ++](https://en.wikipedia.org/wiki/C%2B%2B) .  Se bazează pe cadrul [Electron](https://en.wikipedia.org/wiki/Electron_(software_framework)) , care este utilizat pentru a dezvolta [aplicații web](https://en.wikipedia.org/wiki/Web_application)[Node.js](https://en.wikipedia.org/wiki/Node.js) care rulează pe [motorul de aspect Blink](https://en.wikipedia.org/wiki/Blink_layout_engine) . Visual Studio Code folosește aceeași componentă de editor (denumită în cod „Monaco”) folosită în [Azure DevOps](https://en.wikipedia.org/wiki/Azure_DevOps_Server) (denumită anterior Visual Studio Online și Visual Studio Team Services).

În loc de un sistem de proiect, permite utilizatorilor să deschidă unul sau mai multe directoare, care pot fi apoi salvate în spații de lucru pentru reutilizare viitoare. Acest lucru îi permite să funcționeze ca un editor de cod [agnostic](https://en.wikipedia.org/wiki/Language-agnostic) pentru orice limbă. Suportă o serie de limbaje de programare și un set de caracteristici care diferă în funcție de limbă. Fișierele și folderele nedorite pot fi excluse din arborele proiectului prin intermediul setărilor. Multe caracteristici Visual Studio Code nu sunt expuse prin meniuri sau interfața cu utilizatorul, dar pot fi accesate prin paleta de comenzi.

Visual Studio Code poate fi extins prin [extensii](https://en.wikipedia.org/wiki/Plug-in_(computing)) , disponibile printr-un depozit central. Aceasta include adăugiri la editor și asistență lingvistică. O caracteristică notabilă este abilitatea de a crea extensii care adaugă suport pentru [limbi](https://en.wikipedia.org/wiki/Programming_language) noi , [teme](https://en.wikipedia.org/wiki/Theme_(computing)) și [depanatoare](https://en.wikipedia.org/wiki/Debugger) , efectuează [analize statice de cod](https://en.wikipedia.org/wiki/Static_code_analysis) și adaugă [lintere de cod](https://en.wikipedia.org/wiki/Lint_(software)) folosind [Protocolul de server de limbă](https://en.wikipedia.org/wiki/Language_Server_Protocol) .

Visual Studio Code include mai multe extensii pentru [FTP](https://en.wikipedia.org/wiki/FTP) , permițând utilizarea software-ului ca alternativă gratuită pentru dezvoltarea web. Codul poate fi sincronizat între editor și server, fără a descărca niciun software suplimentar.

Visual Studio Code permite utilizatorilor să seteze [pagina de cod](https://en.wikipedia.org/wiki/Code_page) în care este salvat documentul activ, caracterul de [linie nouă](https://en.wikipedia.org/wiki/Newline) și limbajul de programare al documentului activ. Acest lucru permite utilizarea acestuia pe orice platformă, în orice locație și pentru orice limbaj de programare dat.

## Limbaje de programare structurată

**BASIC** (acronim al expresiei engleze Beginners All-purpose Symbolic Instruction Code) este un [limbaj de programare](https://ro.wikipedia.org/wiki/Limbaj_de_programare) de nivel înalt a calculatoarelor, foarte asemănătoare cu [limba engleză](https://ro.wikipedia.org/wiki/Limba_englez%C4%83), ceea ce îl face ușor de memorat și ideal pentru începători; astfel, generația de utilizatori predecesoare erei [PC](https://ro.wikipedia.org/wiki/Computer_personal)-urilor a luat primul contact cu programarea prin intermediul calculatoarelor [HC](https://ro.wikipedia.org/wiki/HC), care aveau la baza acestora o versiune de BASIC. Este un limbaj simplu de învățat, creat în [1964](https://ro.wikipedia.org/wiki/1964) de către [John G. Kemeny](https://ro.wikipedia.org/w/index.php?title=John_G._Kemeny&action=edit&redlink=1) și [Thomas E. Kurtz](https://ro.wikipedia.org/w/index.php?title=Thomas_E._Kurtz&action=edit&redlink=1) la [Dartmouth College](https://ro.wikipedia.org/w/index.php?title=Dartmouth_College&action=edit&redlink=1). Unul dintre cele mai bune editoare, compilatoare și depanatoare pentru începătorii în acest limbaj este QBASIC (acronim pentru Quick Beginners All-purpose Symbolic Instruction Code), care poate fi descărcat gratuit din [Internet](https://ro.wikipedia.org/wiki/Internet). O variantă de BASIC foarte populară este și [Visual Basic](https://ro.wikipedia.org/wiki/Visual_Basic), creată de către compania [Microsoft](https://ro.wikipedia.org/wiki/Microsoft). Aceasta se adresează mai mult avansaților. Visual Basic permite crearea de aplicații relativ complexe prin manipularea [componentelor](https://ro.wikipedia.org/w/index.php?title=Component%C4%83_(informatic%C4%83)&action=edit&redlink=1) vizuale (grafice) ale programului.

**Fortran** (sau FORTRAN) este un [limbaj de programare](https://ro.wikipedia.org/wiki/Limbaj_de_programare) născut în anul 1950 și care este încă folosit după jumătate de secol de existență. Numele său își are originea din "Formula Translating System". Versiunile mai vechi ale acestui limbaj erau cunoscute drept FORTRAN, dar după Fortran 90 numele nu a mai fost scris în întregime cu majuscule. Deși la început a fost un limbaj [procedural](https://ro.wikipedia.org/w/index.php?title=Procedural&action=edit&redlink=1), cele mai recente versiuni au inclus suport pentru programarea obiectuală.

**Pascal** este unul dintre [limbajele de programare](https://ro.wikipedia.org/wiki/Limbaj_de_programare) de referință în știința [calculatoarelor](https://ro.wikipedia.org/wiki/Computer). Pascal a fost dezvoltat de elvețianul [Niklaus Wirth](https://ro.wikipedia.org/wiki/Niklaus_Wirth) în 1970 pentru a pune în practică [programarea structurată](https://ro.wikipedia.org/wiki/Programare_structurat%C4%83), aceasta fiind mai ușor de [compilat](https://ro.wikipedia.org/w/index.php?title=Compilare&action=edit&redlink=1). Unul din marile sale avantaje este asemănarea cu limbajul natural limba engleză, ceea ce îl face limbajul ideal pentru cei care sunt la primul contact cu programarea. Pascal este bazat pe limbajul Algol și a fost denumit astfel în onoarea matematicianului Blaise Pascal, creditat pentru construirea primelor mașini de calcul numeric. Wirth a mai dezvoltat limbajele Modula-2 și Oberon, similare cu Pascal.Cele mai populare implementări a acestui limbaj au fost Turbo Pascal și Borland Pascal, ambele ale firmei Borland cu versiuni pentru [Macintosh](https://ro.wikipedia.org/wiki/Macintosh) și [DOS](https://ro.wikipedia.org/wiki/DOS), care i-au adăugat limbajului obiecte și au fost continuate cu versiuni destinate programării vizuale pentru Microsoft Windows (utilizate de mediul de dezvoltare [Delphi](https://ro.wikipedia.org/wiki/Embarcadero_Delphi)) și pentru Linux ([Kylix](https://ro.wikipedia.org/wiki/Kylix)). În prezent există și alte implementări mai mult sau mai puțin populare, dar gratuite, printre care se remarcă [Free Pascal](https://ro.wikipedia.org/wiki/Free_Pascal) și [GNU Pascal](https://ro.wikipedia.org/wiki/GNU_Pascal).

Deși în prezent este relativ rar folosit în industria [software](https://ro.wikipedia.org/wiki/Software), el este încă util elevilor și studenților care doresc să se inițieze în programare. Spre deosebire de [BASIC](https://ro.wikipedia.org/wiki/BASIC), care a stat în trecut la baza învățării programării, Pascal este un limbaj structurat.

# Capitolul II Prototipul

Tema:**”** SE pentru recomandarea profesiilor pentru absolvenții liceelor **“**

Universul problemei**:** Partea a doua prezintă baza de cunoștințe creată, aceast system își propune ca scop să ajute abiturienților să aleagă o viitoare profesie în dependență de preferințele sale. Desigur acest system expert poate fi folosit nu doar de abiturienți dar de orice persoană care ar vrea să afle compatibilitatea lui cu o profesie, un domeniu.

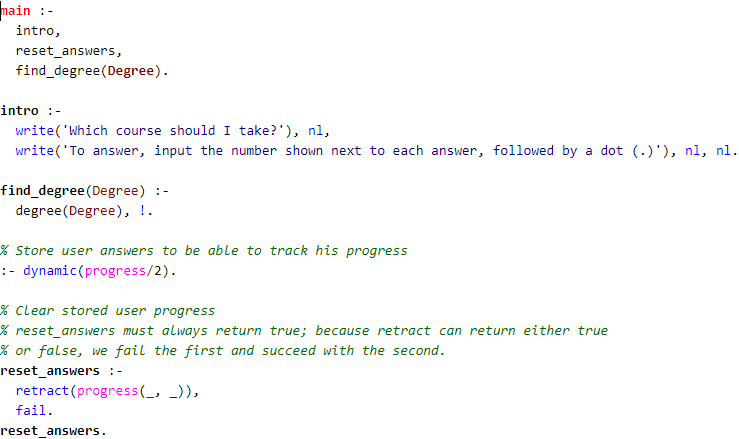
## Obiectivele:

1. Crearea interfeței agreabile pentru utilizator.
2. Acoperirea domeniilor de de interes existente :arta, matematică, fizică, informatică

## Tascurile propuse:

1. Crearea bazei de cunoștințe pentru prototip
2. Descrierea regulilor pentru folosirea corectă a bazei de cunoștință.
3. Crearea întrebărilor și răspunsurilor posibile pentru ușurarea alegerii domeniilor de către utilizator.

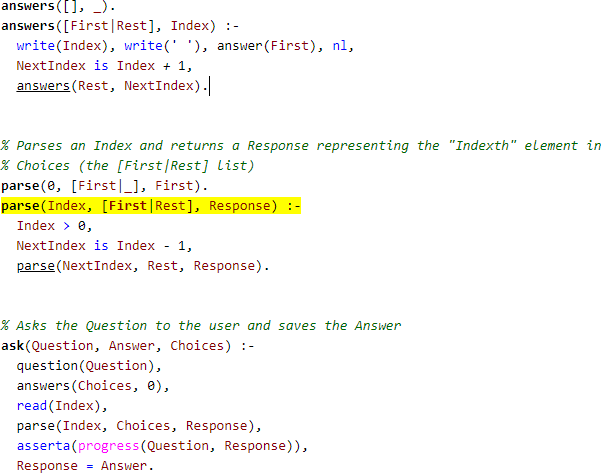
## Explicarea codului:



*Figura 1.*

Programul initial prezintă niște reguli în bază cărora este efectuată cercetarea. În primul rînd avem codul care răspunde de interacțiunea cu utilizatorul acesta este văzut în Figura 1.

Aici avem 3 reguli de bază: intro – care afișează textul principal la intrarea în sistem, reset\_answer – pentru resetarea întrebării și anularea progresului, find\_degree – pentru salvarea gradului de progress a utilizatorului



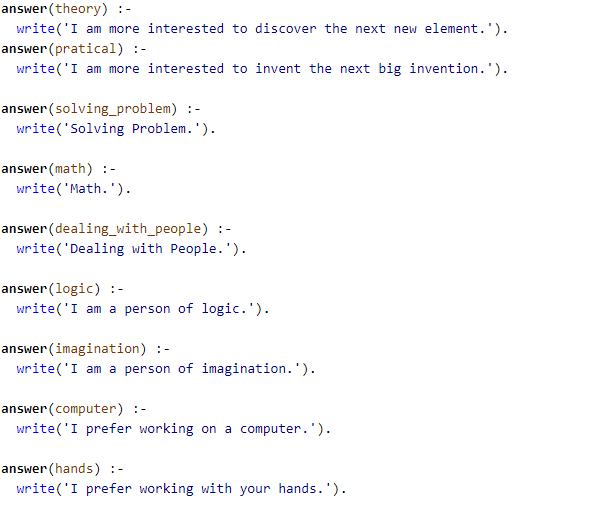
*Figura 2.*

În continuare avem Figura 2, blocul care afișează o listă de răspunsuri frumos formatată, este lista Opțiuni, Indexul este indicele Primului în alegeri. Apoi analizează un index și returnează un răspuns care reprezintă elementul "Indexth" în Alegeri (lista). Ultimul bloc pune întrebarea utilizatorului și salvează răspunsul. Secvența de cod afișată constituie partea de bază pentru rularea programului.



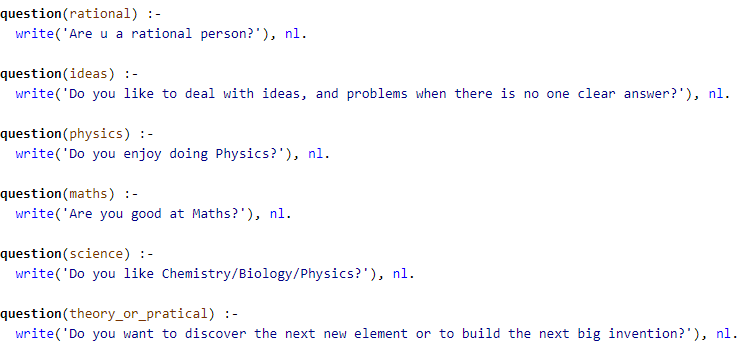
*Figura 3.*

Acest bloc Figura 3 răspunde de alocarea răspunsului la întrebarea oferită de sistem. Aici avem variabila care face referință la blocul de descriere a domeniilor, respectiv în dependență de răspunsul oferet (yes sau no) programul salvează răspunsul pentru folosirea acestuia în continuare.



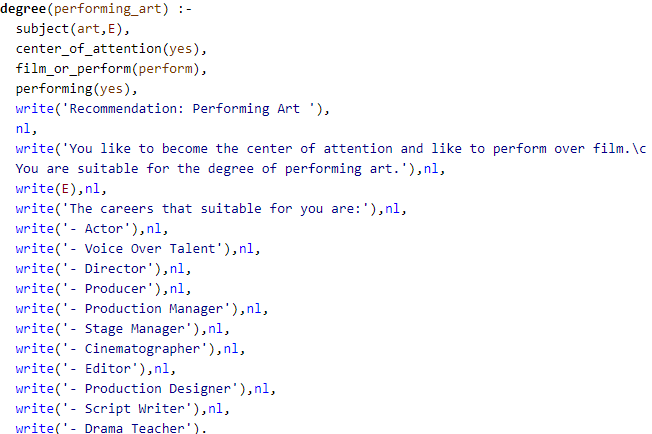
*Figura 4.*

Această parte de cod Figura 4 prezintă un tip de regulă care oferă răspunsurile la întrebările scrise în maniera similar. Acesta este făcut prin menționarea variabilei la care se atribuie răspunsul. A fost folosit parametrul *write* pentru a putea afișa la ecran răspunsul.



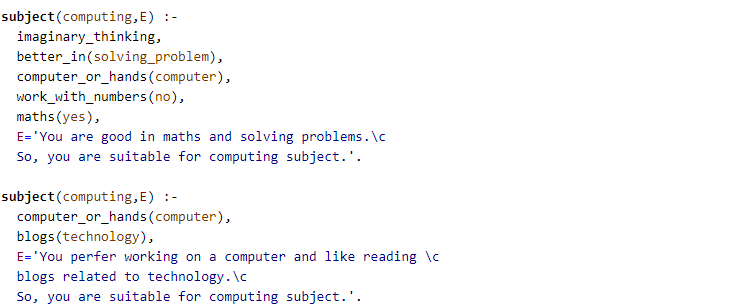
*Figura 5.*

Porțiunea de cod care răspunde de întrebările oferite de sistem este vărută în Figura. Ca și în cazul cu răspunsurile pentru afișarea întrebării la ecram folosim funcția *write.*



*Figura 6.*

Codul care răspunde pentru recomandarea preferinței în baza răspunsurilor Figura 6. Ea prezintă regula care conține subiectul adică domeniul, și caracteristicile domeniului respectiv. Dacă acestea corespund cu răspunsurile utilizatorului sistemul afișează informația.

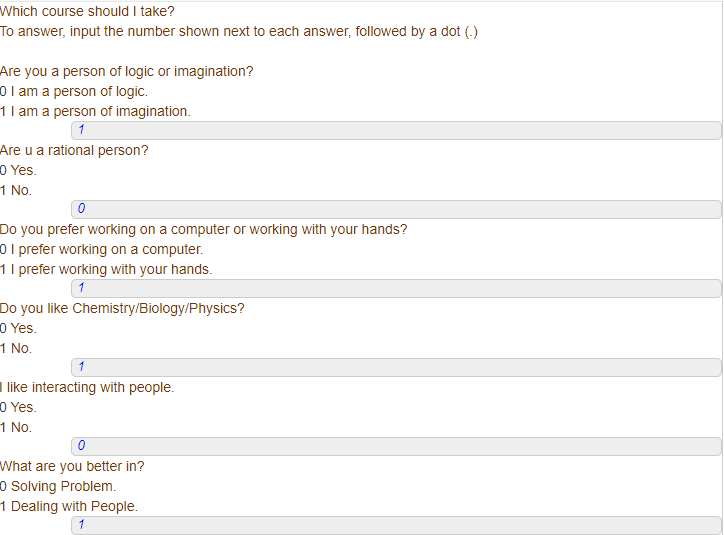


*Figura 7.*

Setul de subiecte ce corespunde domeniilor generale o profesiilor solicitate. Pentru sumarea alegerilor efectuate de utilizator este scris un text de descriere a preșerințelor totale a utilizatorului. Această manipulare a fost efectuată pentru ca utilizatorul să poată vedea care este totalul răspunsurilor sale.

## Rularea programului.

La rulare, programul ne întîlnește cu două propoziții una fiind întrebarea iar alta este instrucțiunea pentru a putea participa la încercarea creată. Prima întrebare este făcută după un șablon predefinit, aceasta este oferită tuturor. Pentru a răspunde la întrbare avem afișat un TextBox în care introduce cifra cu variant de răspuns dorită, apăsînd Enter pe ecran apare următoarea întrebare. Următoarele întrebări sunt afișate în dependență de răspunsul precedent.

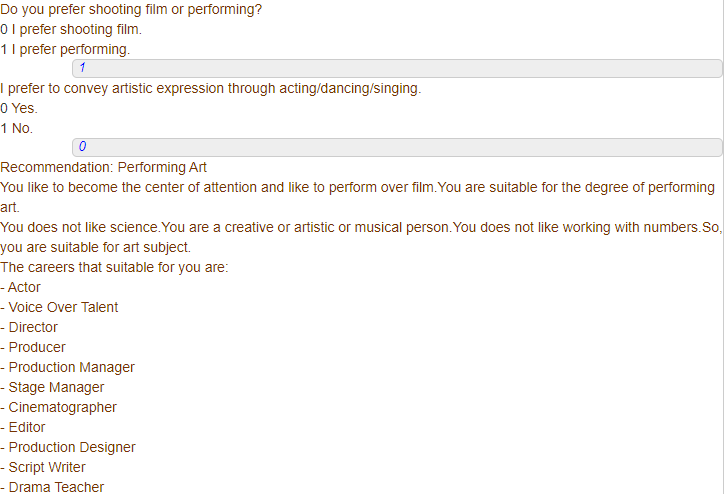
****

*Figura 8.*

După răspunsurile oferite putem observ că persoana este pasionată mai mult de artă, deci toate răspunsurile sun îndreptate în această direcție.

****

*Figura 9*

****

*Figura 10.*

După cum observăm programul a rulat cu success, în urma răspunsurilor oferite de utilizator, sistemul a fost în stare să ofere o serie de domenii în care persoana cu preferințele sale spefice poate să-și regăsească cea mai apropiată de el. La afișare apare textul de totalizare a răspunsurilor oferite iar mai jos sunt domeniile de studiere propuse de system pentru această persoană.

# Capitolul III Codul sursă

main :-

intro,

reset\_answers,

find\_degree(Degree).

intro :-

write('Which course should I take?'), nl,

write('To answer, input the number shown next to each answer, followed by a dot (.)'), nl, nl.

find\_degree(Degree) :-

degree(Degree), !.

% Store user answers to be able to track his progress

:- dynamic(progress/2).

% Clear stored user progress

% reset\_answers must always return true; because retract can return either true

% or false, we fail the first and succeed with the second.

reset\_answers :-

retract(progress(\_, \_)),

fail.

reset\_answers.

% =====

% Rules

% =====

logical\_thinking :- logic\_or\_imagination(logic), rational(yes).

imaginary\_thinking :- logic\_or\_imagination(imagination), rational(no).

physics\_person :- work\_with\_numbers(yes), maths(yes), physics(yes).

% =======

% Subject

% =======

subject(computing,E) :-

logical\_thinking,

better\_in(solving\_problem),

computer\_or\_hands(computer),

work\_with\_numbers(yes),

maths(yes),

E='You are a logical person. \c

You are also good in maths and solving problems using computer. \c

So, you are suitable for computing subject.'.

subject(computing,E) :-

imaginary\_thinking,

better\_in(solving\_problem),

computer\_or\_hands(computer),

work\_with\_numbers(no),

maths(yes),

E='You are good in maths and solving problems.\c

So, you are suitable for computing subject.'.

subject(computing,E) :-

computer\_or\_hands(computer),

blogs(technology),

E='You perfer working on a computer and like reading \c

blogs related to technology.\c

So, you are suitable for computing subject.'.

subject(engineering,E) :-

logical\_thinking,

science(yes),

theory\_or\_pratical(pratical),

better\_in(solving\_problem),

challenge\_yourself(yes),

work\_with\_numbers(yes),

E='You are a logical person and like science. \c

You like to do practical stuff and good in solving problems.\c

You also like to challenge yourself and work with numbers.\c

So, you are suitable for engineering subject.'.

subject(engineering,E) :-

science(yes),

blogs(science),

theory\_or\_pratical(pratical),

better\_in(solving\_problem),

challenge\_yourself(yes),

E='You love science and like to read blogs related to science.\c

You like to do practical stuff and good in solving problems.\c

You also like to challenge yourself.\c

So, you are suitable for engineering subject.'.

subject(science,E) :-

logical\_thinking,

science(yes),

better\_in(solving\_problem),

theory\_or\_pratical(theory),

work\_with\_numbers(yes),

blogs(science),

E='You are a logical person who love science.\c

You are good in solving problem and like theory over practical.\c

You like to work with numbers and read blogs related to science.\c

So, you are suitable for science subject.'.

subject(science,E) :-

science(yes),

theory\_or\_pratical(theory),

E='You love science and prefer theory over practical.\c

So, you are suitable for science subject.'.

subject(business,E) :-

like\_interact(yes),

better\_in(dealing\_with\_people),

planning(yes),

risk(yes),

E='You like to interact with people, dealing with people and like planning.\c

You are also a risk taker.\c

So, you are suitable for business subject.'.

subject(business,E) :-

like\_interact(yes),

better\_in(dealing\_with\_people),

blogs(business),

planning(yes),

E='You like to interact and dealing with people.\c

You also like to read business blog and planning.\c

So, you are suitable for business subject.'.

subject(art,E) :-

imaginary\_thinking,

science(no),

creative\_artistic\_musical(yes),

work\_with\_numbers(no),

going\_museum(yes),

E='You are a person who like to imagine and does not like science.\c

You are also a creative or artistic or musical person.\c

You does not like working with numbers.\c

You like going to movie theater or museum.\c

So, you are suitable for art subject.'.

subject(art,E) :-

science(no),

creative\_artistic\_musical(yes),

work\_with\_numbers(no),

E='You does not like science.\c

You are a creative or artistic or musical person.\c

You does not like working with numbers.\c

So, you are suitable for art subject.'.

subject(hospitality,E) :-

computer\_or\_hands(hands),

like\_interact(yes),

planning(yes),

service\_minded(yes),

serving\_people(yes),

E='You prefer to working with hands.\c

You like to interact with people and planning.\c

You are a service minded person who like to serve people.\c

So, you are suitable for hospitality subject.'.

subject(hospitality,E) :-

computer\_or\_hands(hands),

service\_minded(yes),

serving\_people(yes),

E='You prefer to working with hands.\c

You are a service minded person who like to serve people.\c

So, you are suitable for hospitality subject.'.

subject(hospitality,E) :-

computer\_or\_hands(hands),

serving\_people(yes),

E='You prefer working with hands.\c

You like to serve people.\c

So, you are suitable for hospitality subject.'.

% ======

% Degree

% ======

% Computing

degree(computer\_science) :-

subject(computing,E),

computer\_systems(yes),

technology(develop),

write('Recommendation: Computer Science '),

nl,

write('You are interested in the details of how computer systems or software works.\c

You also prefer to develop technology rather than apply technology.\c

So, you are suitable for the degree of computer science.'),nl,

write(E),nl,

write('The careers that suitable for you are:'),nl,

write('- Chief Technology Officer'),nl,

write('- Software Engineer / Software Architect'),nl,

write('- Mobile App Developer'),nl,

write('- Game Developer'),nl,

write('- System Designer'),nl,

write('- Network Specialist'),nl,

write('- Research Analyst'),nl,

write('- Software Quality Assurance Officer'),nl,

write('- Enterprise Distributed Application Developer').

degree(computer\_science) :-

subject(computing,E),

computer\_systems(yes),

write('Recommendation: Computer Science '),

nl,

write('You are interested in the details of how computer systems or software works.\c

You also prefer to develop technology rather than apply technology.\c

So, you are suitable for the degree of computer science.'),nl,

write(E),nl,

write('The careers that suitable for you are:'),nl,

write('- Chief Technology Officer'),nl,

write('- Software Engineer / Software Architect'),nl,

write('- Mobile App Developer'),nl,

write('- Game Developer'),nl,

write('- System Designer'),nl,

write('- Network Specialist'),nl,

write('- Research Analyst'),nl,

write('- Software Quality Assurance Officer'),nl,

write('- Enterprise Distributed Application Developer').

degree(computer\_science) :-

subject(computing,E),

technology(develop),

write('Recommendation: Computer Science '),

nl,

write('You are interested in the details of how computer systems or software works.\c

You also prefer to develop technology rather than apply technology.\c

So, you are suitable for the degree of computer science.'),nl,

write(E),nl,

write('The careers that suitable for you are:'),nl,

write('- Chief Technology Officer'),nl,

write('- Software Engineer / Software Architect'),nl,

write('- Mobile App Developer'),nl,

write('- Game Developer'),nl,

write('- System Designer'),nl,

write('- Network Specialist'),nl,

write('- Research Analyst'),nl,

write('- Software Quality Assurance Officer'),nl,

write('- Enterprise Distributed Application Developer').

degree(information\_technology) :-

subject(computing,E),

technology(apply),

like\_interact(yes),

planning(yes),

write('Recommendation: Information Technology '),

nl,

write('You are interested to interact with people and like to planning.\c

So, you are suitable for the degree of information technology.'),nl,

write(E),nl,

write('The careers that suitable for you are:'),nl,

write('- IT Specialist'),nl,

write('- Network Specialist'),nl,

write('- System Engineer'),nl,

write('- Web / E-Commerce Architect'),nl,

write('- Database Architect'),nl,

write('- IT Infrastructure Engineer').

degree(information\_technology) :-

subject(computing,E),

write('Recommendation: Information Technology '),

nl,

write('You are suitable for the degree of information technology.'),nl,

write(E),nl,

write('The careers that suitable for you are:'),nl,

write('- IT Specialist'),nl,

write('- Network Specialist'),nl,

write('- System Engineer'),nl,

write('- Web / E-Commerce Architect'),nl,

write('- Database Architect'),nl,

write('- IT Infrastructure Engineer').

% Engineering

degree(electrical\_engineering) :-

subject(engineering,E),

physics(yes),

circuits(yes),

write('Recommendation: Electrical Engineering '),

nl,

write('You like physics and like to deal with circuits. \c

You are suitable for the degree of electrical engineering.'),nl,

write(E),nl,

write('The careers that suitable for you are:'),nl,

write('- Electrical or Electronic Engineer'),nl,

write('- Design Manager in Designing'),nl,

write('- Maintenance of Electrical Power System, Electricity Generation, Transmission, Distribution, Utilisation'),nl,

write('- Electronic System for Operational Safety/Efficiency or Designing Renewable Energy System').

degree(mechanical\_engineering) :-

subject(engineering,E),

physics(yes),

write('Recommendation: Mechanical Engineering '),

nl,

write('You like physics. \c

You are suitable for the degree of mechanical engineering.'),nl,

write(E),nl,

write('The careers that suitable for you are:'),nl,

write('- Mechanical Engineer'),nl,

write('- Production Engineer'),nl,

write('- Failure Analyst Engineer'),nl,

write('- M&E Engineer'),nl,

write('- QC Engineer'),nl,

write('- Manufacturing Engineer'),nl,

write('- R&D Engineer'),nl,

write('- Design Engineer'),nl,

write('- Product Engineer').

% Error, wont go into here

degree(chemical\_engineering) :-

subject(engineering,E),

physics(no),

chemistry(yes),

write('Recommendation: Chemical Engineering '),

nl,

write('You like chemistry. \c

You are suitable for the degree of chemical engineering.'),nl,

write(E),nl,

write('The careers that suitable for you are:'),nl,

write('- Process Engineer'),nl,

write('- Quality Assurance Engineer'),nl,

write('- Chemical & Biochemical Engineer'),nl,

write('- Contamination Engineer').

% Science

degree(biotechnology) :-

subject(science,E),

biology(yes),

genetic\_engineering(yes),

write('Recommendation: Biotechnology '),

nl,

write('You like biology and find generic engineering interesting.\c

You are suitable for the degree of biotechnology.'),nl,

write(E),nl,

write('The careers that suitable for you are:'),nl,

write('- Pharmaceutical Research & Development'),nl,

write('- Pharmaceutical Marketing Director'),nl,

write('- Clinical Trial Manager'),nl,

write('- Clinical Research Scientist'),nl,

write('- Biomedical & Biotechnology Research Scientist'),nl,

write('- Medical & Scientific Product Specialist'),nl,

write('- Medical Laboratories Director'),nl,

write('- Academia (Science Educator)').

degree(pure\_science) :-

subject(science,E),

write('Recommendation: Pure Science '),

nl,

write('You are suitable for the degree of pure science.'),nl,

write(E),nl,

write('The careers that suitable for you are:'),nl,

write('- Analytical/Biomedical chemist'),nl,

write('- Biotechnologist'),nl,

write('- Quality controller'),nl,

write('- Science journalist'),nl,

write('- Process control specialist').

% Business

degree(marketing) :-

subject(business,E),

storytelling(yes),

write('Recommendation: Marketing '),

nl,

write('You like storytelling.\c

You are suitable for the degree of marketing.'),nl,

write(E),nl,

write('The careers that suitable for you are:'),nl,

write('- Chief Executive Officer'),nl,

write('- Chief Marketing Officer'),nl,

write('- Advertising and Promotions Specialist'),nl,

write('- Product Manager'),nl,

write('- Brand Manager'),nl,

write('- Marketing Consultant'),nl,

write('- Management Consultant'),nl,

write('- Retail Expert'),nl,

write('- Customer Relationship Associate / Manager'),nl,

write('- Supply Chain Manager'),nl,

write('- Key Accounts Specialist / Manager'),nl,

write('- Consumer Research Analyst / Manager'),nl,

write('- Events Producer').

degree(accounting) :-

subject(business,E),

work\_with\_numbers(yes),

detail\_oriented(yes),

write('Recommendation: Accounting '),

nl,

write('You like working with numbers and you are a detail oriented person.\c

You are suitable for the degree of accounting.'),nl,

write(E),nl,

write('The careers that suitable for you are:'),nl,

write('- Chief Financial Officer'),nl,

write('- Investment Banker'),nl,

write('- Commercial Banker'),nl,

write('- Financial Controller'),nl,

write('- Internal & External Auditor'),nl,

write('- Management Consultant'),nl,

write('- Tax Consultant'),nl,

write('- Finance Analyst'),nl,

write('- Fund Manager'),nl,

write('- Financial Planner'),nl,

write('- Corporate Treasurer').

degree(business\_management) :-

subject(business,E),

write('Recommendation: Business Management '),

nl,

write('You are suitable for the degree of business management'),nl,

write(E),nl,

write('The careers that suitable for you are:'),nl,

write('- Chief Executive Officer'),nl,

write('- Business Consultant'),nl,

write('- Entrepreneur'),nl,

write('- General Manager'),nl,

write('- Strategic Planner'),nl,

write('- Marketing Manager'),nl,

write('- HR Manager'),nl,

write('- Operation / Project Manager'),nl,

write('- Researcher'),nl,

write('- Organisational Development Manager'),nl,

write('- Procurement Manager').

% Arts

degree(performing\_art) :-

subject(art,E),

center\_of\_attention(yes),

film\_or\_perform(perform),

performing(yes),

write('Recommendation: Performing Art '),

nl,

write('You like to become the center of attention and like to perform over film.\c

You are suitable for the degree of performing art.'),nl,

write(E),nl,

write('The careers that suitable for you are:'),nl,

write('- Actor'),nl,

write('- Voice Over Talent'),nl,

write('- Director'),nl,

write('- Producer'),nl,

write('- Production Manager'),nl,

write('- Stage Manager'),nl,

write('- Cinematographer'),nl,

write('- Editor'),nl,

write('- Production Designer'),nl,

write('- Script Writer'),nl,

write('- Drama Teacher').

degree(digital\_film\_production) :-

subject(art,E),

film\_or\_perform(film),

film(yes),

write('Recommendation: Digital Film Production '),

nl,

write('You like to film over perform.\c

You are suitable for the degree of digital film production.'),nl,

write(E),nl,

write('The careers that suitable for you are:'),nl,

write('- Audio-Visual Producer'),nl,

write('- Transmedia Designer'),nl,

write('- Documentary Filmmaker'),nl,

write('- Creative Entrepreneur'),nl,

write('- Videographer'),nl,

write('- Director'),nl,

write('- Production Designer'),nl,

write('- Video Producer').

% Hospitality

degree(culinary\_art) :-

subject(hospitality,E),

cook(yes),

write('Recommendation: Culinary Art '),

nl,

write('You enjoy cooking.\c

You are suitable for the degree of culinary art.'),nl,

write(E),nl,

write('The careers that suitable for you are:'),nl,

write('- Chef de Cuisine / Executive Chef'),nl,

write('- Banquet Manager'),nl,

write('- Food and Beverage Manager'),nl,

write('- In-flight Food Manager'),nl,

write('- Product Development Manager'),nl,

write('- Restaurant Owner / Entrepreneur').

degree(hotel\_management) :-

subject(hospitality,E),

write('Recommendation: Hotel Management '),

nl,

write('You are suitable for the degree of hotel management.'),nl,

write(E),nl,

write('The careers that suitable for you are:'),nl,

write('- Sales and Marketing Manager'),nl,

write('- Front Office Manager'),nl,

write('- Hotel Human Resource Manager'),nl,

write('- Rooms Division Manager'),nl,

write('- Purchasing Manager'),nl,

write('- Food & Beverage Manager').

degree(gap\_year):-

write('Recommendation: Gap Year '),

nl,

write('Sorry. We cannot help you because you have a \c

a variety of traits.').

% =========

% Questions

% =========

question(logic\_or\_imagination) :-

write('Are you a person of logic or imagination?'), nl.

question(rational) :-

write('Are u a rational person?'), nl.

question(ideas) :-

write('Do you like to deal with ideas, and problems when there is no one clear answer?'), nl.

question(physics) :-

write('Do you enjoy doing Physics?'), nl.

question(maths) :-

write('Are you good at Maths?'), nl.

question(science) :-

write('Do you like Chemistry/Biology/Physics?'), nl.

question(theory\_or\_pratical) :-

write('Do you want to discover the next new element or to build the next big invention?'), nl.

question(better\_in) :-

write('What are you better in?'), nl.

question(bad\_at) :-

write('What are you weak at?'), nl.

question(challenge\_yourself) :-

write('Do you like to challenge yourself?'), nl.

question(lifelong\_learner) :-

write('Are you a lifelong learner?'), nl.

question(try\_new\_things) :-

write('Do you like to explore new things?'), nl.

question(computer\_or\_hands) :-

write('Do you prefer working on a computer or working with your hands?'), nl.

question(like\_interact) :-

write('I like interacting with people.'), nl.

question(serving\_people) :-

write('I dont mind serving people.'), nl.

question(risk) :-

write('I like to take risk'), nl.

question(work\_with\_numbers) :-

write('Do you like working with numbers?'), nl.

question(creative\_artistic\_musical) :-

write('Are you creative or artistic or musical?'), nl.

question(going\_museum) :-

write('Do you enjoy going to movie theater and museums?'), nl.

question(service\_minded) :-

write('Are you service minded with a high stress threshold and want to work with people?'), nl.

question(planning) :-

write('Do you like planning, organising or managing?'), nl.

question(blogs) :-

write('If given a choice, which blogs do you prefer to read?'), nl.

% Question for Computing

question(computer\_systems) :-

write('Are you interested in the details of how computer systems or software works?'), nl.

question(technology) :-

write('Do you prefer to apply or develop technology?'), nl.

% Question for Engineering

question(circuits) :-

write('Do you like to deal with circuits?'), nl.

question(chemistry) :-

write('Do you like Chemistry?'), nl.

% Question for Science

question(biology) :-

write('Do you like Biology?'), nl.

question(genetic\_engineering) :-

write('Do you find genetic engineering intersting?'), nl.

% Question for Business

question(detail\_oriented) :-

write('Are you detail oriented and pay attention to little things?'), nl.

question(storytelling) :-

write('Are you good at storytelling?'), nl.

% Question for Arts

question(center\_of\_attention) :-

write('I like being the center of attention.'), nl.

question(film\_or\_perform) :-

write('Do you prefer shooting film or performing?'), nl.

question(film) :-

write('Are you interested in the arts of storytelling and the language of film?'), nl.

question(performing) :-

write('I prefer to convey artistic expression through acting/dancing/singing.'), nl.

% Question for Hospitality

question(cook) :-

write('Do you enjoy cooking?'), nl.

% ========

% Answers

% ========

% General

answer(theory) :-

write('I am more interested to discover the next new element.').

answer(pratical) :-

write('I am more interested to invent the next big invention.').

answer(solving\_problem) :-

write('Solving Problem.').

answer(math) :-

write('Math.').

answer(dealing\_with\_people) :-

write('Dealing with People.').

answer(logic) :-

write('I am a person of logic.').

answer(imagination) :-

write('I am a person of imagination.').

answer(computer) :-

write('I prefer working on a computer.').

answer(hands) :-

write('I prefer working with your hands.').

answer(yes) :-

write('Yes.').

answer(no) :-

write('No.').

answer(business) :-

write('Forbes or BusinessInsider.').

answer(technology) :-

write('HackerNews or AnandTech.').

answer(science) :-

write('howstuffworks or New Scientist.').

answer(apply) :-

write('I prefer to apply technology.').

answer(develop) :-

write('I prefer to develop technology.').

answer(film) :-

write('I prefer shooting film.').

answer(perform) :-

write('I prefer performing.').

answer(strongly\_disagree) :-

write('Strongly disagree.').

answer(disagree) :-

write('disagree.').

answer(neutral) :-

write('Neutral').

answer(agree) :-

write('agree.').

answer(strongly\_agree) :-

write('Strongly agree.').

% ==========================

% Assign Answer to Questions

% ==========================

science(Answer) :-

progress(science, Answer).

science(Answer) :-

\+ progress(science, \_),

ask(science, Answer, [yes, no]).

physics(Answer) :-

progress(physics, Answer).

physics(Answer) :-

\+ progress(physics, \_),

ask(physics, Answer, [yes, no]).

maths(Answer) :-

progress(maths, Answer).

maths(Answer) :-

\+ progress(maths, \_),

ask(maths, Answer, [yes, no]).

chemistry(Answer) :-

progress(chemistry, Answer).

chemistry(Answer) :-

\+ progress(chemistry, \_),

ask(chemistry, Answer, [yes, no]).

biology(Answer) :-

progress(biology, Answer).

biology(Answer) :-

\+ progress(biology, \_),

ask(biology, Answer, [yes, no]).

theory\_or\_pratical(Answer) :-

progress(theory\_or\_pratical, Answer).

theory\_or\_pratical(Answer) :-

\+ progress(theory\_or\_pratical, \_),

ask(theory\_or\_pratical, Answer, [theory, pratical]).

better\_in(Answer) :-

progress(better\_in, Answer).

better\_in(Answer) :-

\+ progress(better\_in, \_),

ask(better\_in, Answer, [solving\_problem, dealing\_with\_people]).

bad\_at(Answer) :-

progress(bad\_at, Answer).

bad\_at(Answer) :-

\+ progress(bad\_at, \_),

ask(bad\_at, Answer, [math, dealing\_with\_people]).

logic\_or\_imagination(Answer) :-

progress(logic\_or\_imagination, Answer).

logic\_or\_imagination(Answer) :-

\+ progress(logic\_or\_imagination, \_),

ask(logic\_or\_imagination, Answer, [logic, imagination]).

rational(Answer) :-

progress(rational, Answer).

rational(Answer) :-

\+ progress(rational, \_),

ask(rational, Answer, [yes, no]).

ideas(Answer) :-

progress(ideas, Answer).

ideas(Answer) :-

\+ progress(ideas, \_),

ask(ideas, Answer, [yes, no]).

risk(Answer) :-

progress(risk, Answer).

risk(Answer) :-

\+ progress(risk, \_),

ask(risk, Answer, [yes, no]).

challenge\_yourself(Answer) :-

progress(challenge\_yourself, Answer).

challenge\_yourself(Answer) :-

\+ progress(challenge\_yourself, \_),

ask(challenge\_yourself, Answer, [yes, no]).

lifelong\_learner(Answer) :-

progress(lifelong\_learner, Answer).

lifelong\_learner(Answer) :-

\+ progress(lifelong\_learner, \_),

ask(lifelong\_learner, Answer, [yes, no]).

try\_new\_things(Answer) :-

progress(try\_new\_things, Answer).

try\_new\_things(Answer) :-

\+ progress(try\_new\_things, \_),

ask(try\_new\_things, Answer, [yes, no]).

like\_interact(Answer) :-

progress(like\_interact, Answer).

like\_interact(Answer) :-

\+ progress(like\_interact, \_),

ask(like\_interact, Answer, [yes, no]).

serving\_people(Answer) :-

progress(serving\_people, Answer).

serving\_people(Answer) :-

\+ progress(serving\_people, \_),

ask(serving\_people, Answer, [yes, no]).

service\_minded(Answer) :-

progress(service\_minded, Answer).

service\_minded(Answer) :-

\+ progress(service\_minded, \_),

ask(service\_minded, Answer, [yes, no]).

planning(Answer) :-

progress(planning, Answer).

planning(Answer) :-

\+ progress(planning, \_),

ask(planning, Answer, [yes, no]).

work\_with\_numbers(Answer) :-

progress(work\_with\_numbers, Answer).

work\_with\_numbers(Answer) :-

\+ progress(work\_with\_numbers, \_),

ask(work\_with\_numbers, Answer, [yes, no]).

computer\_or\_hands(Answer) :-

progress(computer\_or\_hands, Answer).

computer\_or\_hands(Answer) :-

\+ progress(computer\_or\_hands, \_),

ask(computer\_or\_hands, Answer, [computer, hands]).

creative\_artistic\_musical(Answer) :-

progress(creative\_artistic\_musical, Answer).

creative\_artistic\_musical(Answer) :-

\+ progress(creative\_artistic\_musical, \_),

ask(creative\_artistic\_musical, Answer, [yes, no]).

going\_museum(Answer) :-

progress(going\_museum, Answer).

going\_museum(Answer) :-

\+ progress(going\_museum, \_),

ask(going\_museum, Answer, [yes, no]).

blogs(Answer) :-

progress(blogs, Answer).

blogs(Answer) :-

\+ progress(blogs, \_),

ask(blogs, Answer, [business, technology, science]).

% Computer Science

computer\_systems(Answer) :-

progress(computer\_systems, Answer).

computer\_systems(Answer) :-

\+ progress(computer\_systems, \_),

ask(computer\_systems, Answer, [yes, no]).

technology(Answer) :-

progress(technology, Answer).

technology(Answer) :-

\+ progress(technology, \_),

ask(technology, Answer, [apply, develop]).

% Engineering

circuits(Answer) :-

progress(circuits, Answer).

circuits(Answer) :-

\+ progress(circuits, \_),

ask(circuits, Answer, [yes, no]).

% Science

genetic\_engineering(Answer) :-

progress(genetic\_engineering, Answer).

genetic\_engineering(Answer) :-

\+ progress(genetic\_engineering, \_),

ask(genetic\_engineering, Answer, [yes, no]).

% Business

storytelling(Answer) :-

progress(storytelling, Answer).

storytelling(Answer) :-

\+ progress(storytelling, \_),

ask(storytelling, Answer, [yes, no]).

detail\_oriented(Answer) :-

progress(detail\_oriented, Answer).

detail\_oriented(Answer) :-

\+ progress(detail\_oriented, \_),

ask(detail\_oriented, Answer, [yes, no]).

% Art

center\_of\_attention(Answer) :-

progress(center\_of\_attention, Answer).

center\_of\_attention(Answer) :-

\+ progress(center\_of\_attention, \_),

ask(center\_of\_attention, Answer, [yes, no]).

film\_or\_perform(Answer) :-

progress(film\_or\_perform, Answer).

film\_or\_perform(Answer) :-

\+ progress(film\_or\_perform, \_),

ask(film\_or\_perform, Answer, [film, perform]).

film(Answer) :-

progress(film, Answer).

film(Answer) :-

\+ progress(film, \_),

ask(film, Answer, [yes, no]).

performing(Answer) :-

progress(performing, Answer).

performing(Answer) :-

\+ progress(performing, \_),

ask(performing, Answer, [yes, no]).

% Hospitality

cook(Answer) :-

progress(cook, Answer).

cook(Answer) :-

\+ progress(cook, \_),

ask(cook, Answer, [yes, no]).

% Outputs a nicely formatted list of answers

% [First|Rest] is the Choices list, Index is the index of First in Choices

answers([], \_).

answers([First|Rest], Index) :-

write(Index), write(' '), answer(First), nl,

NextIndex is Index + 1,

answers(Rest, NextIndex).

% Parses an Index and returns a Response representing the "Indexth" element in

% Choices (the [First|Rest] list)

parse(0, [First|\_], First).

parse(Index, [First|Rest], Response) :-

Index > 0,

NextIndex is Index - 1,

parse(NextIndex, Rest, Response).

% Asks the Question to the user and saves the Answer

ask(Question, Answer, Choices) :-

question(Question),

answers(Choices, 0),

read(Index),

parse(Index, Choices, Response),

asserta(progress(Question, Response)),

Response = Answer.

# Concluzii:

În urma efectuării acestei lucrări de curs am reușit să combin toate materialele trecute la cursul de Sisteme Expert. Această lucrare a prezentat un bun start pentru acumularea deprinderilor de lucru în domeniul respectiv. Prologul, limbajul în care a fost creat prototipul, poate nu este cel mai popular și atrpgptor doar că după ce am lucrat în el pot spune că este interesant și destul de util. Un system expert în zilele noastre este o parte importantă deoarece nu fiecare dintre noi ar putea avea discuții cu oameni din diverse domenii care sunt experți, astfel această modalitate este una agreabilă pentru rezolvarea unor probleme din diferite domenii.

Pe parcursul efectuării acestei lucrări a întîlnit unele dificultăți deoarece prologul este un limbaj destul de stric care nu eartă chiar și erori minime, dar cercetînd diverse surse am reușit să depășesc greutățile întîmpinate pentru a ajunge la scopul final bine realizat.

# Bibliografie:

1. SWI Prolog [Resursa electronica] <https://www.swiprolog.org/pldoc/man?section=implhistory>
2. Biblioteca electronica a sistemului [Resursa electronica]

<http://www.math.md/studlib/informatica/prolog.html>

1. Programare logica [Resursa electronica]

<https://ru.scribd.com/doc/37540910/Programare-Logica>

1. Programarea structurata [Resursa electronica]

<https://ro.wikipedia.org/wiki/Programare_structurat%C4%83>

1. Elemente de programare structurata [Resursa electronica]

<https://www.creeaza.com/referate/informatica/sql/Elemente-de-programare-structu887.php>