Conceptele Fundamentale ale Limbajele de Programare

-Scurtă prezentare-

Obiectul studiului în cadrul acestei materii îl reprezintă *limbajele de programare*. Nu ne interesează un anumit limbaj de programare; nu ne interesează să învăţăm să programăm într-un limbaj de programare sau altul. Ceea ce ne interesează este *studiul conceptelor fundamentale* ce stau la baza proiectării limbajelor de programare şi dezvoltarea acestor concepte odată cu evoluţia limbajelor de programare.

Disciplina este studiată în semestrul 4, după ce s-au parcurs sau urmează să se parcurgă mai multe materii din zona Limbaje şi programare: Programarea calculatoarelor, Tehnici de programare, Programare orientată pe obiecte, Structuri de date şi algoritmi, Proiectarea şi analiza algoritmilor, Fundamente de inginerie software. Se poate considera că studenţii au acumulat suficiente cunoştinţe în domeniu pentru a înţelege problemele de principiu care se pun în discuţie.

Problemele care se pun la început: Ce este un limbaj de programare? Criterii pentru clasificarea şi compararea limbajelor de programare. Reprezentări formale pentru descrierea limbajelor de programare. Iată câteva dintre răspunsurile la aceste probleme:

Un **limbaj de programare** reprezintă o notaţie formală prin care se specifică anumite operaţii ce urmează a fi executate (de către un calculator).

Există, în prezent, un număr impresionant de limbaje de programare definite şi implementate. Dintre acestea, doar o mică parte s-au impus şi se utilizează pe scară largă pentru scrierea programelor actuale. Este principalul motiv pentru care se pune problema clasificării şi comparării limbajelor.

Desigur, limbajul de programare nu reprezintă un scop în sine. El trebuie să permită realizarea în mod eficient a unui software de calitate. Rezultă că înainte de a stabili ce înseamnă un limbaj de programare bun trebuie să definim ce înseamnă un program bun. Cele *trei calităţi de bază ale unui sistem de programe* sunt:

1. **Fiabilitatea sistemului**. Fiabilitatea presupune funcţionarea corectă (corespunzătoare specificărilor) a sistemului chiar şi în prezenţa unor incidente hardware sau software.
2. **Mentenabilitatea**. Aceasta presupune calitatea software-ului de a fi uşor de modificat în vederea includerii unor noi facilităţi sau a îmbunătăţirii celor existente.
3. **Eficienţa**. Implică oferirea unor servicii optime în cadrul limitat al resurselor disponibile.

La obţinerea acestor calităţi concură desigur o serie întreagă de factori: metoda de proiectare, uneltele oferite de mediul de programare, algoritmii utilizaţi, factori umani. Un loc central este, desigur, ocupat de limbajul de programare.

De asemenea vom realiza o sistematizare a limbajelor de programare pe familii, sistematizare care uşurează studiul şi înţelegerea lor. La ce ne ajută toate aceste lucruri? Iată câteva dintre răspunsurile posibile:

- aprecierea calităţilor şi lipsurilor unui limbaj de programare;
- înţelegerea mai profundă a limbajelor utilizate în mod curent;
- îmbogăţirea cunoştinţelor de programare cu noi concepte fundamentale;
- învăţarea mai uşoară a unui nou limbaj de programare (fiind familiar cu conceptele de bază încorporate);
- utilizarea eficientă a oricărui limbaj de programare;
- selectarea corectă a limbajului (limbajelor) potrivit pentru o anumită aplicaţie;
- proiectarea unui nou limbaj de programare (eventual a unui subset sau a unei extensii).

Structura cursului, pe capitole, este următoarea:

Capitolul	Conţinuturi
1. Introducere. Calităţile unui limbaj de programare. Clasificări	1.1 Limbajul de programare în cadrul procesului de dezvoltare software 1.2 Criterii pentru evaluarea unui limbaj de programare 1.3 Cele trei famiii de limbaje de programare: imperative, funcţionale, declarative 1.4 Programarea secvenţială şi programarea concurentă 1.5 Scurt istoric al dezvoltării limbajelor de programare
2. Reprezentarea formală a limbajelor de programare	2.1 Sintaxa 2.2 Semantica
3. Implementarea limbajelor de programare	3.1 Interpretarea 3.2 Translatarea 3.3 Considerente comparative 3.4 Depalii privind procesul de compilare
4. Atributele entităţilor unui limbaj de programare	4.1 Domeniul variabilelor 4.2 Durata de existenţă a variabilei. Alocarea memoriei 4.3 Valoarea variabilei. Tipul variabilei
5. Transmiterea datelor ca parametri	5.1 Transmiterea prin adresă, prin copiere, prin nume 5.2 Transmiterea datelor ca parametri, în diverse limbaje de programare 5.3 Transmiterea subprogramelor ca parametri 5.4 Subprograme generice
6. Tipuri de date – prezentare generală	6.1 Tipuri predefinite. Tipuri definite de programator. 6.2 Tipuri scalare. Tipuri de date structurate 6.3 Tipul pointer 6.4 Compatibilitatea tipurilor 6.5 Sistemul de tipuri al limbajului Pascal, al limbajului C, al limbajului Ada, al limbajului Lisp 6.6 Comparaţie. Limbaje puternic tipizate
7. Tipuri de date abstracte	7.1 Descrierea datelor abstracte 7.2 Date abstracte în câteva limbaje de programare moderne: trecere în revistă 7.3 Date abstracte în Modula 2, Ada, C++
8. Limbaje orientate pe obiecte	8.1 Programarea orientată pe obiecte 8.2 Programarea orientată pe obiecte în limbajele Java, C#, Lisp
9. Structuri de control în limbajele de programare	9.1 Structuri de control la nivel de instrucţiune 9.2 Subprograme 9.3 Tratarea excepţiilor
10. Bazele teoretice ale programării funcţionale	10.1 Lambda calculus 10.2 Evaluarea leneşă 10.3 Funcţii de ordin superior 10.4 Tipuri. polimorfism

La laborator se abordează programarea funcţională în limbajele LISP şi ML. Motivul este acela că paradigma *programării funcţionale* este mai puţin studiată la alte materii pe parcursul facultăţii (într-o mică măsură la *Logică şi structuri discrete*)

Evaluare: Examen scris, cu durata de 3 ore, cu subiecte teoretice care vizează înţelegerea conceptelor, problematicii şi aplicării rezultatelor teoretice (pondere de 2/3 din nota de examen) şi cu subiecte aplicative constând în rezolvarea unor probleme în limbajele de programare funcţionale LISP sau ML (1/3 din nota de examen). Nota finală conţine nota de la examen în proporţie de 2/3 şi nota de la activitatea pe parcurs (laborator + un mic proiect), în proporţie de 1/3.

Bibliografie selectivă:

1.Carlo Ghezzi, Mehdi Jarayeri, *Programming Languages*, John Wiley 1987.
2.Ellis Horowitz, *Fundamentals of Programming Languages*, Computer Science Press, 1984.
3.Horia Ciocârlie, *Limbaje de programare. Concepte fundamentale,* Editura de Vest, 2007.
4.Horia Ciocârlie, *Universul limbajelor de programare,* Editura Eurostampa, 2011.