**General**

Limbajele de programare reprezintă un set de instrucțiuni și sintaxe utilizate pentru a crea programe și aplicații software, precum și pentru a controla și manipula sistemele informatice. Limbajele de programare permit oamenilor să comunice cu calculatoarele într-un limbaj pe care acestea îl pot înțelege. Limbajele de programare sunt utilizate în diverse scopuri, cum ar fi dezvoltarea de site-uri web, dezvoltarea de dispozitive mobile, analiza datelor, dezvoltarea de jocuri și multe altele.

Unele dintre caracteristicile cheie ale limbajelor de programare includ:

* Sintaxa: Regulile și structura specifice utilizate pentru a scrie codul într-un limbaj de programare.
* Tipuri de date: Tipul de valori care pot fi stocate într-un program, cum ar fi numere, șiruri de caractere și booleeni.
* Variabile: Poziții de memorie numite care pot stoca valori.
* Operatori: Simboluri utilizate pentru a efectua operații asupra valorilor, cum ar fi adunarea, scăderea și compararea.
* Structuri de control: Declarații utilizate pentru a controla fluxul unui program, cum ar fi declarațiile if-else, buclele și apelurile de funcții.
* Biblioteci și cadre: Colecții de coduri pre-scrise care pot fi utilizate pentru a îndeplini sarcini comune și pentru a accelera dezvoltarea.
* Paradigme: Stilul sau filozofia de programare utilizată în limbaj, cum ar fi procedural, orientat pe obiecte sau funcțional.
* Există multe limbaje de programare diferite, fiecare având propriile puncte forte și puncte slabe și fiind potrivit pentru diferite tipuri de proiecte. Printre cele mai utilizate limbaje de programare se numără Java, Python, C++, JavaScript și C#. Alegerea limbajului de programare depinde de cerințele specifice ale unui proiect, inclusiv de platforma utilizată, de publicul vizat și de rezultatul dorit.

Limbajele de programare continuă să evolueze și să se schimbe de-a lungul timpului, fiind dezvoltate noi limbaje, iar cele mai vechi sunt actualizate pentru a răspunde nevoilor în schimbare. Programarea este o colaborare între oameni și computere, în care oamenii creează instrucțiuni pe care un computer trebuie să le urmeze (cod) pentru a rezolva probleme.

**Relații cu alte subdomenii ale informaticii**

Limbajele de programare sunt instrumente esențiale pentru informaticieni, deoarece le permit să creeze programe și aplicații software care pot rezolva diverse probleme și îndeplini diverse sarcini. Diferite limbaje de programare au caracteristici, sintaxă și paradigme diferite care le fac potrivite pentru diferite subdomenii ale informaticii. Unele dintre subdomeniile informaticii includ:

* Inginerie software: Acest subdomeniu se concentrează pe proiectarea, dezvoltarea, testarea și întreținerea sistemelor software. Inginerii de software utilizează limbaje de programare precum Java, C#, Python și C++ pentru a crea produse software care să îndeplinească cerințele și specificațiile clienților sau utilizatorilor. De asemenea, ingineria software implică utilizarea metodologiilor de dezvoltare software, cum ar fi agile sau waterfall, pentru a gestiona procesul de dezvoltare software.
* Inteligență artificială: Acest subdomeniu se ocupă cu crearea de mașini sau sisteme care pot îndeplini sarcini care necesită inteligență umană, cum ar fi raționamentul, învățarea, planificarea și luarea de decizii. Inteligența artificială utilizează limbaje de programare precum Python, R, Lisp și Prolog pentru a crea algoritmi și modele care pot imita sau simula cogniția și comportamentul uman. Inteligența artificială include, de asemenea, subdomenii precum învățarea automată, procesarea limbajului natural, viziunea computerizată și robotica.
* Știința datelor: Acest subdomeniu presupune colectarea, analiza și interpretarea unor seturi mari și complexe de date pentru a extrage informații și cunoștințe utile. Oamenii de știință în domeniul datelor utilizează limbaje de programare precum Python, R, SQL și MATLAB pentru a manipula, vizualiza și modela datele folosind diverse tehnici și instrumente, cum ar fi statistica, învățarea automată, bazele de date și extragerea datelor. Știința datelor aplică, de asemenea, analiza datelor în diverse domenii, cum ar fi afacerile, asistența medicală, educația și științele sociale.
* Dezvoltare web: Acest subdomeniu presupune crearea de site-uri web și aplicații web care rulează pe internet sau intranet. Dezvoltatorii web folosesc limbaje de programare precum HTML, CSS, JavaScript, PHP și Ruby pentru a crea pagini web și elemente web interactive, dinamice și receptive. Dezvoltarea web implică, de asemenea, utilizarea de cadre web, cum ar fi Django sau Rails, pentru a simplifica și accelera procesul de dezvoltare web.
* Grafică pe calculator: Acest subdomeniu se ocupă de crearea de imagini și animații cu ajutorul calculatoarelor. Grafica pe calculator utilizează limbaje de programare precum C++, Java, Python și OpenGL pentru a crea programe grafice care pot genera grafică 2D sau 3D folosind diverse tehnici și algoritmi, cum ar fi rasterizarea, ray tracing, umbrirea și redarea. Grafica pe calculator include, de asemenea, subdomenii precum viziunea computerizată, procesarea imaginilor, realitatea virtuală și dezvoltarea de jocuri.

**Probleme interesante**

Dezvoltarea unui nou limbaj de programare sau îmbunătățirea unuia existent este o sarcină provocatoare și plină de satisfacții, care necesită o mulțime de abilități și cunoștințe. Cu toate acestea, dezvoltatorii de limbaje de programare se confruntă, de asemenea, cu probleme și dificultăți interesante pe parcurs. Unele dintre aceste probleme includ:

* Proiectarea sintaxei și a semanticii: Sintaxa și semantica unui limbaj de programare definesc modul în care este scris codul și ce înseamnă acesta. Proiectarea sintaxei și a semanticii unui limbaj de programare reprezintă un echilibru delicat între simplitate, expresivitate, lizibilitate și coerență. Dezvoltatorii de limbaje de programare trebuie să ia în considerare diverși factori, cum ar fi publicul țintă, scopul urmărit, standardele existente și compromisurile dintre diferitele caracteristici și paradigme. De exemplu, Python este cunoscut pentru sintaxa sa simplă și elegantă care pune accentul pe lizibilitate și indentare. Cu toate acestea, unii programatori pot prefera alte limbaje care au o sintaxă mai flexibilă sau mai concisă, cum ar fi Ruby sau Perl.
* Implementarea compilatorului sau a interpretorului: Compilatorul sau interpretorul unui limbaj de programare este software-ul care traduce codul în instrucțiuni executabile pentru un calculator. Implementarea compilatorului sau a interpretorului unui limbaj de programare este o sarcină complexă și tehnică care implică diferite etape, cum ar fi analiza lexicală, analiza parsezării, analiza semantică, generarea și optimizarea codului. Dezvoltatorii de limbaje de programare trebuie să se asigure că compilatorul sau interpretorul este corect, eficient, robust și compatibil cu diferite platforme și arhitecturi. De exemplu, Java este un limbaj compilat care utilizează o mașină virtuală (JVM) pentru a rula pe orice platformă care îl suportă. Cu toate acestea, acest lucru introduce, de asemenea, provocări, cum ar fi supraîncărcarea performanței, gestionarea memoriei și probleme de securitate.
* Testarea și depanarea: Testarea și depanarea sunt etape esențiale în orice proces de dezvoltare de software, dar mai ales în cazul limbajelor de programare. Testarea și depanarea unui limbaj de programare implică verificarea faptului că limbajul se comportă conform așteptărilor și respectă specificațiile sale. Dezvoltatorii de limbaje de programare trebuie să utilizeze diverse metode și instrumente, cum ar fi testarea unitară, testarea de integrare, testarea de regresie, analiza codului, instrumente de depanare și mecanisme de tratare a erorilor. De exemplu, C++ este un limbaj puternic și versatil care suportă mai multe paradigme și caracteristici. Cu toate acestea, acest lucru îl face, de asemenea, predispus la erori și bug-uri care pot fi greu de detectat și corectat.
* Menținerea și evoluția: Limbajele de programare nu sunt entități statice care rămân neschimbate în timp. Limbajele de programare evoluează și se schimbă în mod constant pentru a răspunde noilor nevoi și cerințe ale utilizatorilor și dezvoltatorilor. Întreținerea și evoluția unui limbaj de programare implică adăugarea de noi caracteristici, remedierea erorilor, îmbunătățirea performanțelor, asigurarea compatibilității și actualizarea documentației. Dezvoltatorii de limbaje de programare trebuie să facă față unor provocări diverse, cum ar fi compatibilitatea retroactivă, standardizarea, rata de adoptare, feedback-ul comunității și concurența din partea altor limbaje. De exemplu, JavaScript este un limbaj dinamic și popular care este utilizat pe scară largă pentru dezvoltarea web. Cu toate acestea, el are, de asemenea, multe ciudățenii și inconsecvențe care au fost abordate de versiuni mai noi (ES6+) sau alternative (TypeScript).

**Oameni interesanți**

În domeniul limbajelor de programare există multe persoane și companii interesante care au contribuit la dezvoltarea, inovarea și popularitatea diferitelor limbaje. Printre acestea se numără:

* Guido van Rossum: Este creatorul lui Python, unul dintre cele mai populare și versatile limbaje de programare din lume. A început să lucreze la Python în 1989 ca un proiect de hobby, inspirat de experiența sa anterioară cu limbajele ABC și Modula-3. A vrut să creeze un limbaj ușor de citit, scris și învățat și care să poată susține mai multe paradigme și domenii. A numit limbajul după Monty Python, grupul său de comedie preferat. A fost, de asemenea, dictatorul benevolent pe viață (BDFL) al Python până în 2018, când a renunțat la rolul său și a delegat procesul decizional unei echipe de dezvoltare de bază. În prezent, lucrează la Microsoft ca inginer distins.
* James Gosling: Este părintele Java, unul dintre cele mai utilizate și influente limbaje de programare din lume. A început să lucreze la Java în 1991, în cadrul unui proiect la Sun Microsystems, unde dorea să creeze un limbaj care să poată rula pe orice dispozitiv care suporta mediul său. A fost inspirat de experiența sa anterioară cu limbajele C, C++ și Smalltalk. A numit limbajul după cafeaua Java, pe care a consumat-o foarte mult în timpul dezvoltării sale. De asemenea, a inventat mașina virtuală Java (JVM) și codul de octet Java, care permit ca Java să fie independent de platformă și portabil. În prezent, lucrează la Amazon Web Services ca inginer de marcă.
* Bjarne Stroustrup: Este creatorul lui C++, unul dintre cele mai puternice și versatile limbaje de programare din lume. A început să lucreze la C++ în 1979, ca parte a tezei sale de doctorat la Bell Labs, unde dorea să creeze un limbaj care să extindă capacitățile limbajului C cu caracteristici orientate pe obiecte. A fost inspirat de experiența sa anterioară cu limbajele Simula și BCPL. A numit limbajul C++ după operatorul de incrementare din C, care semnifică o îmbunătățire a C. De asemenea, a inventat conceptele de clase, moștenire, polimorfism, șabloane și excepții în C++. În prezent, lucrează la Morgan Stanley ca director general și la Universitatea Columbia ca profesor invitat.

**Cum studiem noi limbajele de programare la UVT?**

Limbajele de programare sunt o parte esențială a informaticii, deoarece ne permit să creăm programe și aplicații software care pot rezolva diverse probleme și îndeplini diverse sarcini. La UVT, studiem limbajele de programare la diferite cursuri și niveluri, în funcție de specializarea și interesele noastre. Câteva dintre cursurile și limbajele de programare pe care le studiem la UVT sunt următoarele:

* Programare 1, 2 și 3: Acestea sunt cursurile care ne învață cum să folosim C, C++ și Java, trei dintre cele mai utilizate și influente limbaje de programare din lume. Învățăm cum să folosim aceste limbaje pentru programarea sistemelor, dezvoltarea de jocuri, dezvoltarea web și multe altele. Învățăm cum să scriem programe folosind caracteristici orientate pe obiecte, cum ar fi clasele, moștenirea, polimorfismul, șabloanele și excepțiile. De asemenea, învățăm cum să folosim biblioteci și cadre de lucru, cum ar fi STL, Qt, Swing și Spring.
* Programare web: Acesta este cursul care ne învață cum să creăm site-uri și aplicații web folosind HTML, CSS, JavaScript, PHP și MySQL. Învățăm cum să folosim aceste limbaje pentru dezvoltarea web, cum ar fi crearea de pagini web și elemente web interactive, dinamice și receptive. Învățăm, de asemenea, cum să folosim cadre web, cum ar fi Bootstrap, jQuery, Laravel și WordPress.
* Limbaje de programare: Acesta este cursul care ne învață despre teoria și proiectarea limbajelor de programare. Învățăm despre sintaxa și semantica diferitelor limbaje de programare, cum ar fi limbajele imperative, funcționale, logice, de scripting și limbajele specifice unui domeniu. De asemenea, învățăm despre conceptele și tehnicile de compilare și interpretare a limbajelor de programare, cum ar fi analiza lexicală, analiza parțială, analiza semantică, generarea de cod, optimizarea, mașinile virtuale și codul byte.

Acestea sunt câteva dintre exemplele modului în care studiem limbajele de programare la UVT. Limbajele de programare ne permit să ne exprimăm ideile și soluțiile într-un mod pe care computerele îl pot înțelege și executa. Limbajele de programare ne permit, de asemenea, să explorăm diferite aspecte ale calculului și rezolvării problemelor în diverse domenii.

**Literatură intersantă**

* Structure and Interpretation of Computer Programs de Harold Abelson, Gerald Jay Sussman și Julie Sussman. Aceasta este o carte clasică care introduce conceptele și principiile fundamentale ale limbajelor de programare folosind Scheme, un dialect al lui Lisp. Cartea abordează subiecte precum abstractizarea, recursivitatea, modularitatea, structurile de date, structurile de control, abstractizarea metalingvistică și interpretorii. De asemenea, cartea este disponibilă online gratuit la https://mitpress.mit.edu/sites/default/files/sicp/index.html.
* Concepts of Programming Languages de Robert W. Sebesta. Aceasta este o carte cuprinzătoare care analizează principalele caracteristici și probleme de proiectare ale diferitelor limbaje de programare, cum ar fi limbajele imperative, funcționale, logice, orientate pe obiecte, de scripting și concurente. Cartea tratează, de asemenea, sintaxa și semantica limbajelor de programare, precum și implementarea și evaluarea limbajelor de programare. Cartea este potrivită pentru cursurile de licență sau postuniversitare privind limbajele de programare.
* Programming Language Pragmatics de Michael L. Scott. Aceasta este o carte detaliată care acoperă teoria și practica limbajelor de programare, cu accent pe modul în care proiectarea limbajului afectează comportamentul programului. Cartea abordează subiecte precum analiza lexicală și sintactică, rezolvarea numelor, sistemele de tipuri, gestionarea memoriei, tratarea excepțiilor, concurența, programarea funcțională, programarea logică, limbajele de scripting și limbajele specifice unui domeniu. Cartea include, de asemenea, exerciții și studii de caz privind diverse limbaje de programare.
* The C Programming Language de Brian W. Kernighan și Dennis M. Ritchie. Aceasta este o carte clasică care prezintă limbajul de programare C, unul dintre cele mai utilizate și influente limbaje de programare din lume. Cartea acoperă caracteristicile de bază și sintaxa C, precum și funcțiile bibliotecii standard și operațiile de intrare/ieșire. Cartea include, de asemenea, exemple și exerciții care ilustrează utilizarea limbajului C pentru diverse aplicații.
* Limbajul de programare C++ de Bjarne Stroustrup. Aceasta este cartea definitivă despre C++, unul dintre cele mai puternice și versatile limbaje de programare din lume. Cartea acoperă caracteristicile și sintaxa de bază ale limbajului C++, precum și caracteristicile și paradigmele avansate, cum ar fi clasele, moștenirea, polimorfismul, șabloanele, excepțiile, containerele, algoritmii, fluxurile, concurența și programarea generică. Cartea include, de asemenea, exemple și exerciții care demonstrează utilizarea limbajului C++ în diverse domenii.
* Effective Java de Joshua Bloch. Aceasta este o carte practică care oferă cele mai bune practici și orientări pentru scrierea de cod Java de înaltă calitate. Cartea abordează subiecte precum crearea și distrugerea obiectelor, metode comune tuturor obiectelor, clase și interfețe, generics, enums și adnotări, lambdas și fluxuri, metode, concurență, serializare și performanță. Cartea include, de asemenea, exemple și exerciții care ilustrează utilizarea Java în diverse scopuri.