

Introducere în programare

2016 - 2017

1

Bogdan Pătruț,

bogdan@info.uaic.ro

după Corina Forăscu

<http://profs.info.uaic.ro/~introp/>

Conținut și obiective

- inițiere în utilizarea unui limbaj de programare
- însușirea tehniciilor de bază în proiectarea programelor
- însușirea bunelor practici de scriere de cod „curat”
- însușirea tehniciilor de muncă independente, dar și în grup, pentru un proiect mic
- introducere graduală în limbajul C++, cu accent pe implementarea de structuri de date și soluții algoritmice, prezentate în cursul *Structuri de date*
- înțelegerea practicilor de lucru „cu sursă deschisă” (open source)

Organizare

- Colaboratori:
 - prof. Stelian Hadâmbu
 - colab. Alexandru Cîtea
 - colab. Robert Ojoc
 - colab. Daniel Popescu
- Program:
http://profs.info.uaic.ro/~orar/discipline/orar_ip.html +
Consultații (Luni-Miercuri, 13-14, C 414)
- Manieră de comunicare:
 - grupul de pe Facebook „Introducere în programare”
 - e-mail cu subiectul [IP]...

Evaluare

- Metode: laborator, teme și proiecte (evaluare pe parcurs), examene (evaluare directă)
- **forme:**
 - Examene scrise (**E1, E2**) (săptămâna 8 - test din teorie și săptămâna 16 - evaluare practică, prin probleme de laborator), punctate cu note 1..10
 - Teme (**T1, T2, T3**), punctate cu note 1..10: săptămânile 2-4,5-7,9-11
 - Proiect (**P**), punctat cu note 1..10: săptămânile 13-14
 - bonusuri pentru activitatea la laborator
 - Participare la laborator și Activitate zilnică (**L**)
 - Soluții prompte & originale, participare la discuții, competiții (**B** - bonus!)
- **criterii de promovare** (standarde minime de performanță):
 - $E1 \geq 5, E2 \geq 5$
 - $(T1+T2+T3)/3 \geq 5$
 - $P \geq 5$
- **Punctaj final** (pondere a formelor de evaluare în formula notei finale):
 - $PF = (E1+E2)/2 * 40\% + T * 30\% + P * 30\% + L * 6\% + B * 6\%$
- **Nota finală**: normele ECTS, curba lui Gauss (% ↗: 10, 25, 30, 20, 10, 5)

Bibliografie

- H. Schildt. *C++. Manual complet.* Teora, 2000.
- B. Stroustrup. *The C++ Programming Language.* Adisson-Wesley, 3rd ed., 1997.
- ISO International Standard ISO/IEC 14882:2011(E) – *Programming Language C++* <http://isocpp.org/std/the-standard>
- W. Savitch. *Absolute C++, 5th edition.* Addison-Wesley, 2012.
- W. Savitch. *Problem Solving with C++, 8th Edition.* Addison-Wesley, 2012.
- L. Negrescu. *Limbajele C și C++ pentru începători.* Volumul II: Limbajul C++ Ed. Albastră, 2006.
- B. Pătruț. *Aplicații în C și C++.* Ed. Teora, 1998-2001 (în grupul de Facebook)

Cursuri similare folosite

- Corina Forăscu, Cristian Gațu, Mădălina Răschip, Marian Baltă, Dorel Lucanu (FII, anii anteriori)

- Introduction to C++, MIT OpenCourseWare

<http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-096-introduction-to-c-january-iap-2011/index.htm>

- An Introduction to the Imperative Part of C++, Rob Miller, David Clark, William Knottenbelt

<http://www.doc.ic.ac.uk/~wjk/C++Intro/>

- B. Pătruț. *Programare procedurală*. Curs, Universitatea "Vasile Alecsandri" din Bacău (în grupul de Facebook)

Programarea calculatoarelor

- activitate informatică de elaborare a produselor-program, a programelor (software) necesare
- Programarea informatică contine următoarele subactivități: specificarea, proiectarea, implementarea, documentarea și întreținerea produsului program.
- proces ce conduce de la formularea unei probleme (de calcul) la un program executabil pe un calculator
- programming implică activități precum
- sarcini înrudite sunt testarea, depanarea, întreținerea codului sursă, implementarea sistemului construit, managementul acestuia etc; acestea pot fi considerate ca părți ale procesului de programare, dar cel mai adesea sunt asociate termenului de dezvoltare software
- Ingineria software (ingineria programării) combină tehnici ingineresci cu practici de dezvoltare software
- În ingineria software, programarea (implementarea) este privită ca una din fazele procesului de dezvoltare software
- există o permanentă dezbatere dacă scrierea de programe este o formă de artă, un meșteșug sau o disciplină inginerescă

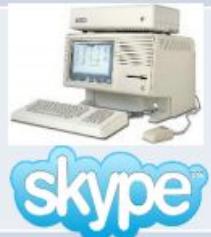
Paradigme de programare

- **Programarea structurată** este o paradigmă a programării apărută după anul 1970 datorită complicării crescânde a programelor de calculatoare. A apărut ca un model nou de programare, în scopul de a crea noi tehnici de programare apte de a produce programe care să fie sigure în funcționare, pe o durată mai lungă.
- **Programarea imperativă**, în contrast cu programarea declarativă, este o paradigmă de programare care descrie calculul ca instrucțiuni ce modifică starea unui program. În aproape același fel în care modul imperativ din limbaiele naturale exprimă comenzi pentru actiuni, programele imperative sunt o secvență de comenzi pentru actionarea calculatorului. **Programarea procedurală** este o metodă obisnuită de executare a programării imperative și de aceea cei doi termeni sunt folosiți deseori ca sinonime.
- **Programarea declarativă** este un stil de programare în care se descrie logica unui calcul, fără să se prezinte modul de execuție: programarea declarativă răspunde la întrebarea ce trebuie calculat și nu la întrebarea *cum* se face acest lucru. Programarea logică și programarea bazată pe reguli se înscriu în paradigma programării declarative.
- **Programarea orientată pe obiecte** (sau programarea orientată obiect) este o paradigmă de programare, axată pe ideea încapsulării, adică grupării datelor și codului care operează asupra lor, într-o singură structură. Un alt concept important asociat programării orientate obiect este polimorfismul.
- **Programarea funcțională** este o paradigmă de programare care tratează calculul ca evaluare de funcții matematice și evită starea și datele mutabile. Se pune accent pe aplicarea de funcții, spre deosebire de programarea imperativă, care folosește în principal schimbările de stare.

Istoria C++

- 1979: Bjarne Stroustrup – Simula, C cu clase ... ->
- 1983: C++ (comparat cu C)
- clase
 - moștenire
 - inlining (inline keywords and Class Member Functions)
 - argumente implicate pentru funcții
 - verificare puternică a tipurilor
 - funcții virtuale
 - supraîncărcarea funcțiilor
 - referințe cu simbolul &
 - cuvântul cheie *const*
- 1998: STL, C++98 - primele standarde C++
- 2003, 2005, 2011, 2014 - C++03, C++0x, C++11, C++14
- 2017 – e plănită o nouă revizuire

Limbaje de programare (1)

1957-1959	FORTRAN (FORmula TRANslation) LISP (List Processor) COBOL (Common Business-oriented language) Oldest languages still in use High-level	Terminator's vision has samples of COBOL source code	Supercomputing appl., AI devel., business software	NASA ATMs, Credit cards
1970	PASCAL (<- Blaise Pascal) High-level, for teaching data structuring	Niklaus Wirth (Turing '84) 	Teaching programming	
1972	C General-purpose, with many derivatives (C#, Java, Perl, PHP, Python)	Dennis Ritchie (Turing '83, K.Thomson) 	Cross-platform programming, System prog., Unix prog, computer game devel.	UNIX® early www servers & clients

Limbaje de programare (2)

1979-1983-...	<p>C++ High-level, OO, expands C</p>	 <p>Bjarne Stroustrup</p>	<p>Commercial appl. devel., embedded software, server/client applic., video games</p>	
1983	<p>Objective C (OO extension of C) High-level, general purpose, OO, expands C with messages based on Smalltalk</p>	 <p>Brad Cox & Tom Love</p>	<p>Apple programming (OS X and iOS) WWDC '14: Swift = Objective C without C WWDC '15: Swift 2.0</p>	
1987	<p>PERL (Practical Extraction and Reporting Language) High-level, general-purpose, interpreted, multi-paradigm language</p>	 <p>Larry Wall</p>	<p>Text processing, graphics programming, system administration, network programming, finance, bioinformatics</p>	

Limbaje de programare (3)

1989-1991	PYTHON (for <i>Monty Python's Flying Circus</i>) High-level, general purpose, multi-paradigm language	Guido van Rossum 	WAD, software devel., information security, biologists, bioinformatics	  
1993	RUBY (a collab's birthstone) High-level, general purpose, OO. Designed (Ada, C++, Perl, Lisp Python) for productive & enjoyable programming	Yukihiro Matsumoto 	Web appl. devel.,	  
1995	JAVA (for the coffee consumed) High-level, general-purpose, cross-platform, multi-paradigm language	James Gosling (Sun Microsystems) 	network programming, WAD, GUI devel., software devel.	   

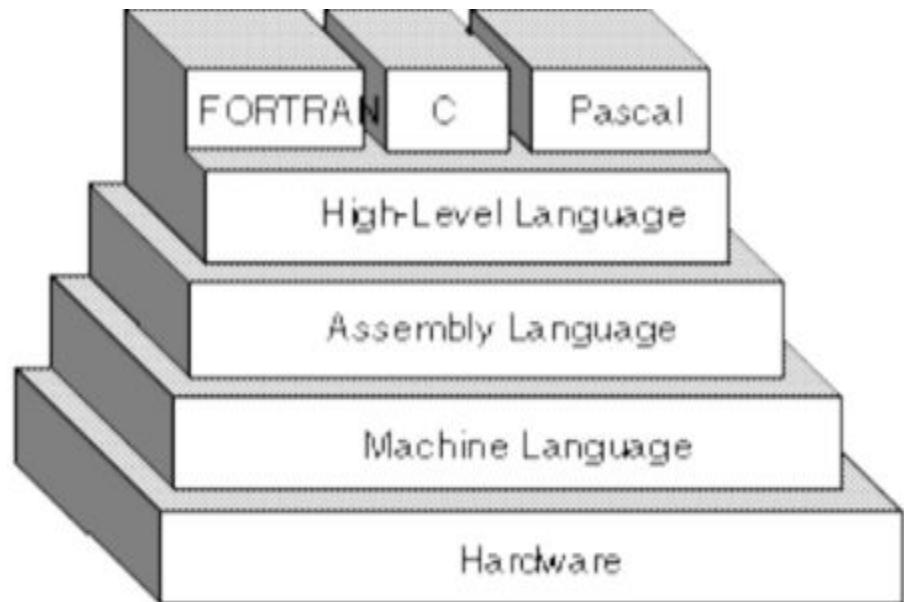
Sursa: Corina Forăscu

Limbaje de programare (4)

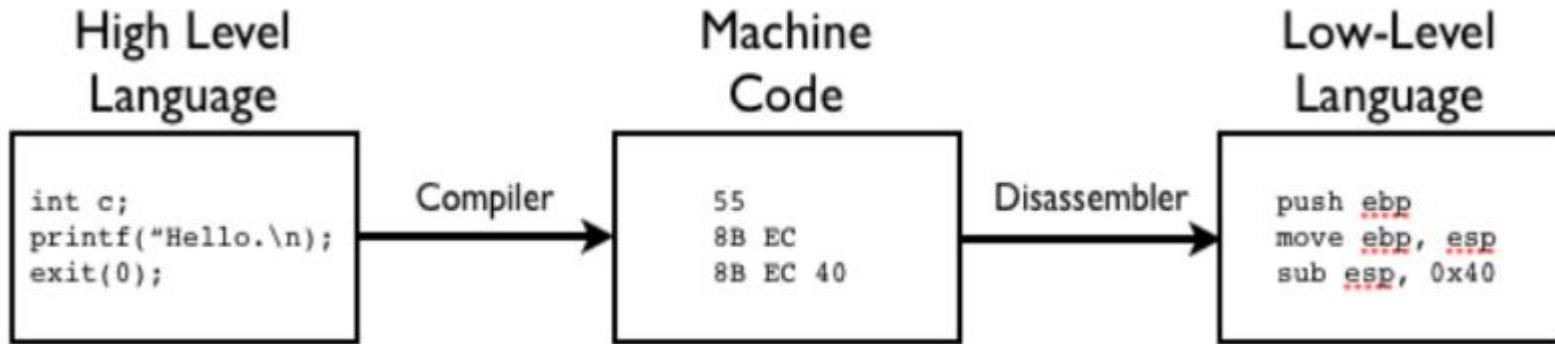
1995	<p>PHP (“Personal Home Page” -> Hypertext Pre-processor) General purpose, open source for building dynamic web pages; influenced by C++, Perl, Java</p>	<p>Rasmus Lerdorf</p> 	<p>Building / maintaining dynamic web pages, server side devel.</p>	
1995	<p>Java Script (after “Mocha”) High-level, scripting, OO, imperative, functional. Designed (C, Java, Python, Scheme) for web programming (esp. client side)</p>	<p>Brendan Eich</p> 	<p>Dynamic web development, PDF docs, web browsers, widgets,</p>	

C++ - noțiuni fundamentale (1)

- Este unul din cele peste 2000 limbaje de programare
- Originile C++
 - limbajele de programare de nivel scăzut (cod mașină, limbaje de asamblare)
 - limbajele de programare de nivel înalt (C, ADA, COBOL, FOTRAN)
 - programarea orientată pe obiecte



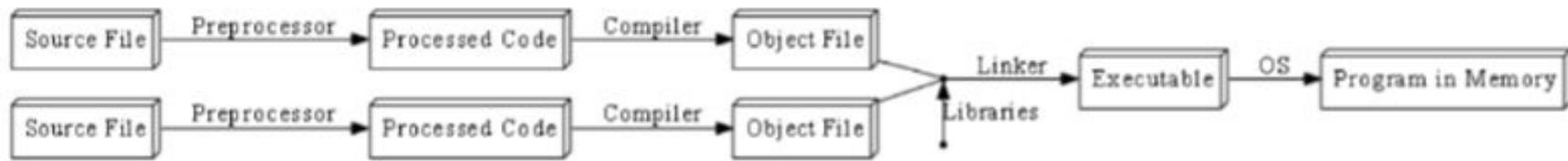
C++ - noțiuni fundamentale (2)



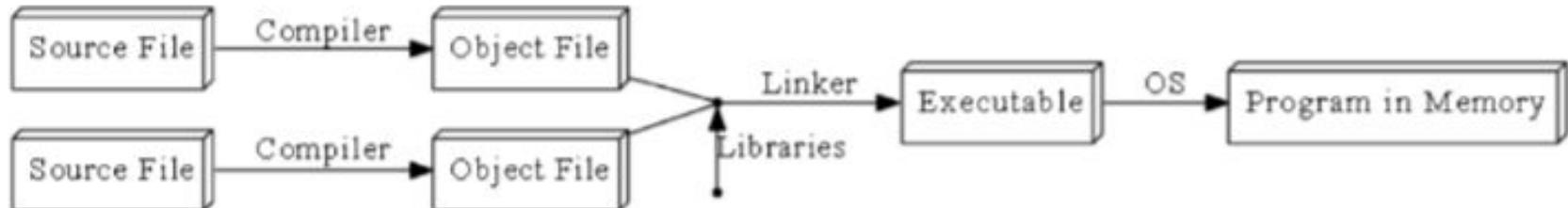
- ușor de programat (mai puțin timp de scris, mai scurt și mai ușor de citit)
- portabil - programele pot rula pe diferite calculatoare fără modificări sau cu puține modificări

Compilarea

- În C++



- generică



Depanarea

- Bug - eroare de programare, greu de sesizat
- Erorile de compilare:
 - probleme suprinse de compilator, în general rezultând din încălcarea regulilor de **sintaxă**
 - sau prin **folosirea incorrectă** a tipurilor de date, alocării memoriei
 - sau erori în etapa de **link-editare**;
- Erori din timpul rulării: programul nu face ceea ce te aștepți să facă („calculatorul face ceea ce i-ai spus să facă, nu ceea ce ai vrea să facă”)

C++ - noțiuni fundamentale (3)

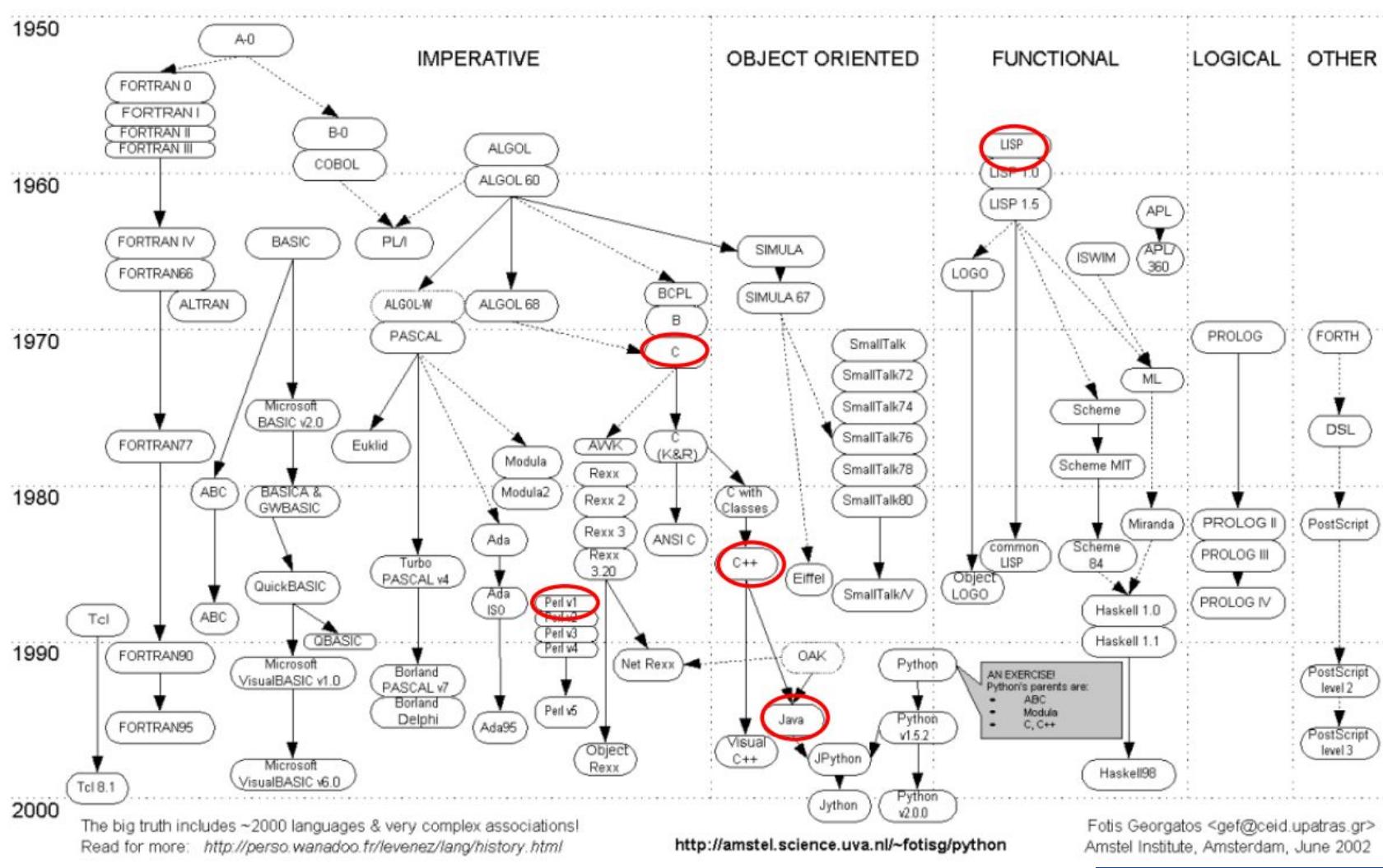
- (1998) limbaje deschise standardizate ISO
- limbaje compilate (se compilează direct în codul nativ al unei mașini)
- limbaje interpretate
- limbaj nesigur puternic tipizat (programatorul trebuie să știe exact ce trebuie să facă)
- acceptă atât variabile de tipuri definite explicit, cât și deduse
- acceptă verificarea statică (la compilare), dar și dinamică a tipurilor (în timpul execuției programului)
- oferă o gamă variată de paradigme de programare
- portabil
- o colecție foarte mare de biblioteci

C++ - noțiuni fundamentale (4)

- Limbaj ce acceptă mai multe **paradigme de programare**:
 - generică: se pot folosi tipuri ce vor fi specificate mai târziu și vor fi instanțiate în funcție de necesități
 - imperativă: *cum* (secvențe de instrucțiuni ce trebuie executate de calculator)
 - structurată: *control intuitiv* (ordinea în care instrucțiunile sunt executate)
 - orientată pe obiecte: *obiecte* + atribute + proceduri (metode)
 - declarativă: *ce* calcule trebuie făcute și nu cum să fie făcute
 - funcțională: programele văzute ca evaluarea unor *funcții* matematice

Evoluția limbajelor de programare

Arborele genealogic, 1950-2000



De ce C++ ? (1)

- portabilitate (Windows, Apple, Linux, UNIX)
- întreținere structurată
- concis
- mai puțin cod
- no run-time overhead
- mai sigur
- mai rapid (viteza de ansamblu este mai mare decât în alte limbaje de programare)
- şabloane şi programare generică
- o multitudine de biblioteci
- limbajul preferat al programatorilor în crearea de jocuri
- limbaj de start pentru alte limbaje de programare

De ce C++ ? (2)

Oct 2014	Oct 2013	Change	Programming Language
1	1		C
2	2		Java
3	3		Objective-C
4	4		C++
5	6	▲	C#
6	7	▲	Basic
7	5	▼	PHP
8	8		Python
9	12	▲	Perl
10	9	▼	Transact-SQL
11	17	▲	Delphi/Object Pascal
12	10	▼	JavaScript
13	11	▼	Visual Basic .NET
14	-	▼	Visual Basic
15	21	▲	R
16	13	▲	Ruby
17	81	▼	Dart
18	24	▲	F#
19	-	▲	Swift
20	14	▲	Pascal
		▼	

oct.15	oct.14	Change	Programming Language
1	2	▲	Java
2	1	▼	C
3	4	▲	C++
4	5	▲	C#
5	8	▲	Python
6	7	▲	PHP
7	13	▲	Visual Basic .NET
8	12	▲	JavaScript
9	9		Perl
10	16	▲	Ruby
11	11		Delphi/Object Pascal
12	31	▲	Assembly language
13	14	▲	Visual Basic
14	3	▼	Objective-C
15	19	▲	Swift
16	20	▲	Pascal
17	27	▲	MATLAB
18	23	▲	PL/SQL
19	29	▲	OpenEdge ABL
20	15	▼	R

Termeni

- limbaj de nivel scăzut / limbaj de nivel înalt
- interpretarea / compilarea unui program
- cod sursă / cod obiect
- parsare
- executabil
- token
- fișier header (bibliotecă)
- declarație / definiție (a unei funcții)
- sintaxă / semantică
- erori de compilare / erori semantice
- bug / depanare

Dacă limbajele de programare ar fi vehicule...



C a fost marele multi-funcțional: compact, puternic, merge oriunde și de încredere în situații în care viața ta depinde de el.

Dacă limbajele de programare ar fi vehicule...



C++ este noul C— de două ori mai puternic, de două ori mai mare, lucrează în medii ostile, dar dacă încerci să-l folosești fără atenție și fără antrenament special, probabil te vei prăbuși.

Dacă limbajele de programare ar fi vehicule...



C# este C++ cu mai multe caracteristici de siguranță, astfel încât să îl poată folosi și civilii obișnuiți. Arată puțin stupid, dar are în cea mai mare parte aceeași putere, atât timp cât stai în apropierea pompelor de carburant, de magazine auto și de confortul civilizației.

Dacă limbajele de programare ar fi vehicule...



Java este o altă încercare de a îmbunătăți C. Într-un fel își face treaba, dar e mult mai lent, mai voluminos, poluează peste tot, iar oamenii vor crede că ești un țărănoi.

Referinte

- www.cplusplus.com
- www.tiobe.com/
- en.cppreference.com/
- www.greenteapress.com/thinkcpp/index.html
- ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-096-introduction-to-c-january-iap-2011/
- kickassinfographics.com/history/the-evolution-of-computer-languages-infographic/
- crashworks.org/if_programming_languages_were_vehicles/