Programski jezici 1

Visoka Tehnička škola strukovnih studija – Niš

Profesor: dr Mirko R. Kosanović

mirko.kosanovic@vtsnis.edu.rs

Asistent: Miloš M. Kosanović

milos.kosanovic@vtsnis.edu.rs

ESPB bodovi: 6

Semestar: II

Fond časova: 2+0+2

Programski jezici 1

Literatura:

1. B. Kernighan, D. Ritchie, **Programski jezik C**, Prentice Hall Software Series, CET 2003.

2. Laslo Kraus, **Programski jezik C sa rešenim zadacima**, Akademska misao, Beograd 2006

3. Vladimir Ćirić, Uvod u programiranje i programski jezik C, Elektronski fakultet, Niš 2014

4. Mirko Kosanović, Slajdovi **sa predavanja**

Programski jezici 1

Polaganje ispita:

- Predispitne obaveze:
  - ✓ Laboratorijske vežbe - **obavezne** 0 - 20
  - ✓ Predavanja 0 - 10
  - ✓ I kolokvijum 0 - 20
  - ✓ II kolokvijum 0 - 20

Ukupno 0-70 poena, **minimum 30** za izlazak na ispit

- Ispit 0 - 30

Programski jezici 1

Ocene:

51 - 60 : 6 (šest)

61 - 70 : 7 (sedam)

71 - 80 : 8 (osam)

81 - 90 : 9 (devet)

91 - 100 : 10 (deset)

Programski jezici 1

Sadržaj predmeta

1. Osnovni pojmovi o programiranju, projektovanje, pisanje, prevođenje, izvršavanje i testiranje programa
2. C jezik - karakteristike, razvojno okruženje
3. Struktura C programa (tipovi podataka, ključne reći, naredbe, operandi, operatori, izrazi i izkazi)
4. Osnovne algoritamske strukture u C jeziku. Kontrola toka podataka (sekvenca, selekcija, petlje)
5. Unos i prikaz podataka, standardne ulazno/izlazne naredbe
6. Podprogrami, deklaracija, definicija i pozivi funkcija, argumenti i povratne vrednosti funkcija

7. Prvi kolokvijum

Programski jezici 1

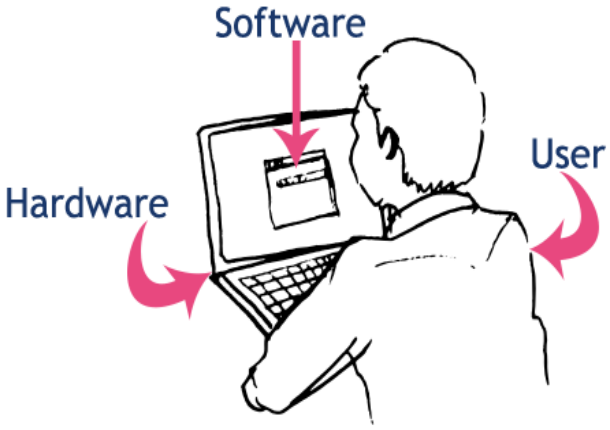
Sadržaj predmeta

8. Složene strukture i izvedeni tipovi podataka, pokazivači, reference
9. Rad sa nizovima
10. Rad sa poljima i matricama
11. Rad sa znakovnim nizovima – stringovima
12. Upravljanje datotekama i pretprocesorske direktive
13. Rekurzija i razgranate strukture – stablo, i strukture za brzo traženje podataka

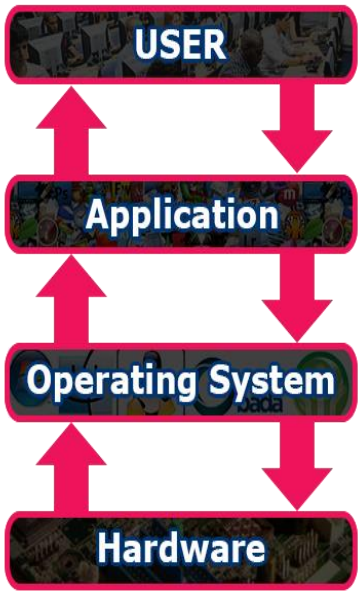
14. Drugi kolokvijum

I - Uvod Programske jezike

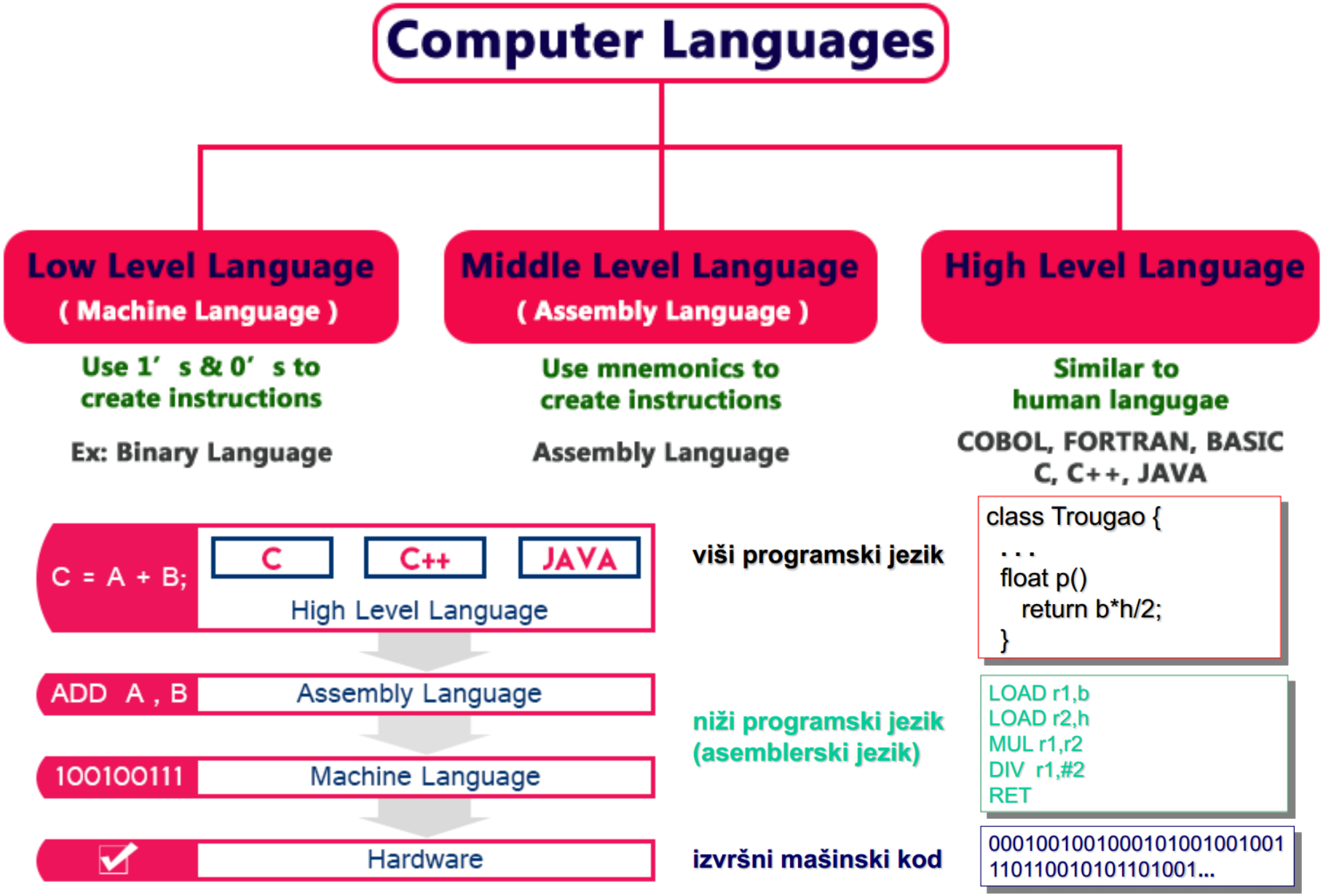
- Ljudi međusobno komuniciraju putem nekog jezika
- Komunikacija između računara i ljudi odvija se korišćenjem računarskih programa (hardver – softver – korisnik)
- Računarski programi se kreiraju putem velikog broja različitih programskih jezika kao što su Basic,Pascal, C, C++, Java, PHP,...
- Svaki programski jezik sastoji se od seta definisanih reči i seta pravila koja se koriste za kreiranje instrukcija (naredbi) programa



PROGRAMIRANJE je proces zadavanja skupa naredbi u nekom programskom jeziku kako bi se izvršila neka aktivnost, odnosno, rešio određeni problem.



I - Podela programskih jezika



I – Etape u pisanju programa

- Rešavanje problema korišćenjem računarskih programa se može razložiti na više etapa:

1. **Definisanje problema** - postupak u kome naručilac i programer na nekom jeziku (srpski, engleski, ...) definišu koje probleme program treba da rešava. Iako nije neophodno, poželjno je da naručilac ima osnovno informatičko znanje, kako bi se lakše sporazumeo sa programerom i kako bi se izbegle eventualne greške.
2. **Analiza problema** - obuhvata definisanje ulaznih i izlaznih podataka, moguća ograničenja njihovih vrednosti, kao i matematički model koji će biti korišćen za rešavanje datog problema.
3. **Definisanje algoritma** - rešava problem, što podrazumeva definisanje uređenog niza pravila kojima se rešava određeni tip problema.
4. **Projektovanje programa** - izbor platforme i programskog jezika, i definisanje arhitekture samog programa i načina čuvanja podataka.
5. **Kodiranje** - prevođenje pravila definisanih algoritmom na konkretan programski jezik.Dobro definisan algoritam olakšava pisanje programa

I – Etape u pisanju programa

6. **Testiranje** – treba blagovremeno da otkrije i ukloni grešake. Testovi pomoću kojih se ispituje funkcionalnost programa treba da obuhvate sve opsege ulaznih promenljivih, kao i sve moguće grane u izvršenju programa. Potrebno je izvršiti i testiranje robustnosti programa u slučajevima unosa neodgovarajućih podataka od strane korisnika.
7. **Analiza rezultata** - poređenje dobijenih rezultata sa teorijskim ili eksperimentalnim rezultatima,kao i modifikaciju modela u slučajevima kada dobijeni rezultati nisu u granicama dozvoljene tolerancije
8. **Isporuka programa** - program se putem različitih medija (DVD, FTP, internet,...) stavlja na raspolaganje naručiocu da ga samostalno koristiti
9. **Održavanje programa** - podrazumeva obuku korisnika, ispravku uočenih nedostataka i prilagođavanje programa zahtevima korisnika.

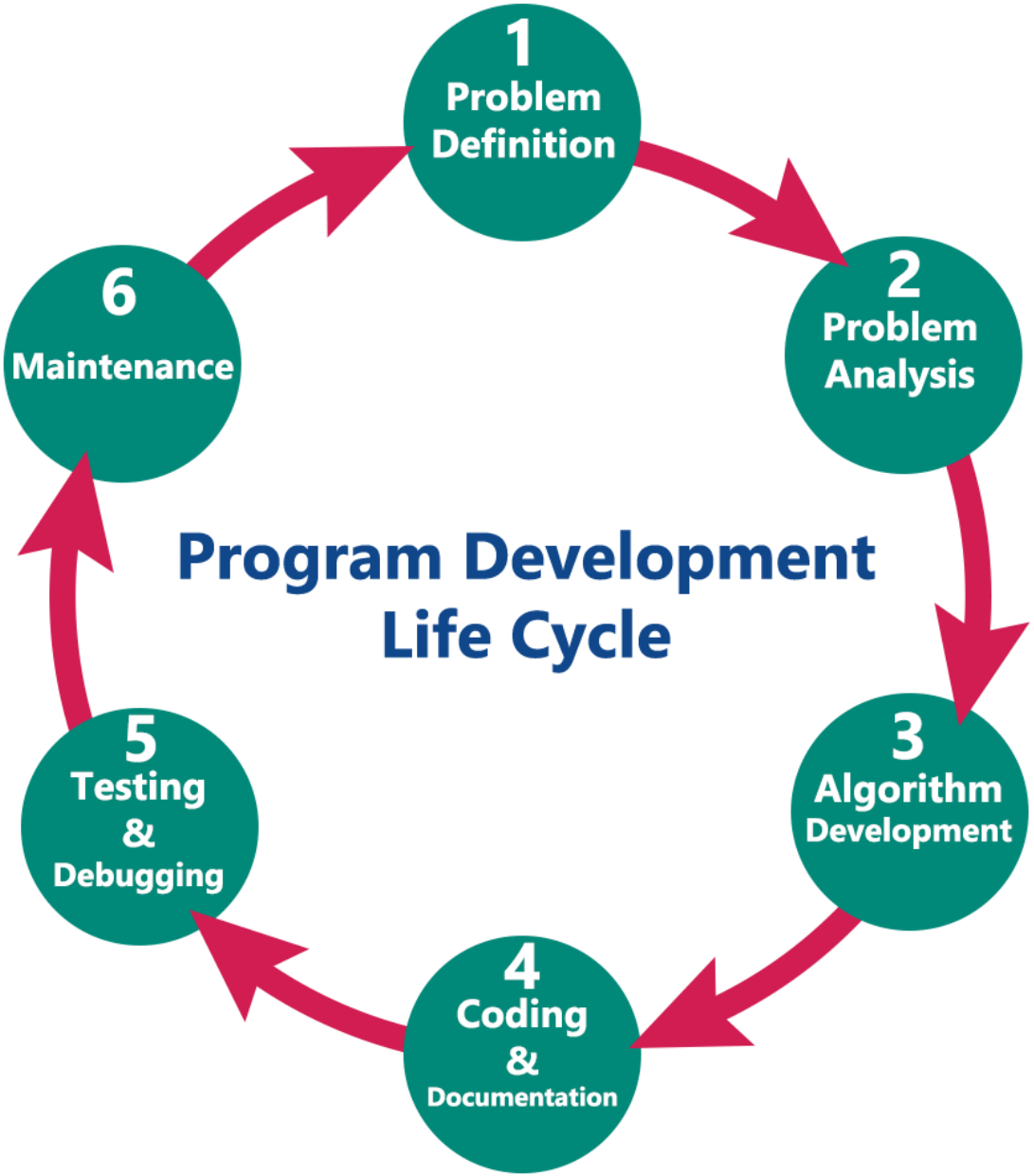
***U slučaju razvoja velikih programskih rešenja postoje različiti tipovi ciklusa kroz koje prolazi svaki program. Izbor najpogodnijeg tipa zavisi od namene programa, broja učesnika na projektu, strategije kompanije koja se bavi razvojem i mnogih drugih parametara*.**

I – Etape u pisanju programa

6. **Testiranje** – treba blagovremeno da otkrije i ukloni grešake. Testovi pomoću kojih se ispituje funkcionalnost programa treba da obuhvate sve opsege ulaznih promenljivih, kao i sve moguće grane u izvršenju programa. Potrebno je izvršiti i testiranje robustnosti programa u slučajevima unosa neodgovarajućih podataka od strane korisnika.
7. **Analiza rezultata** - poređenje dobijenih rezultata sa teorijskim ili eksperimentalnim rezultatima,kao i modifikaciju modela u slučajevima kada dobijeni rezultati nisu u granicama dozvoljene tolerancije
8. **Isporuka programa** - program se putem različitih medija (DVD, FTP, internet,...) stavlja na raspolaganje naručiocu da ga samostalno koristiti
9. **Održavanje programa** - podrazumeva obuku korisnika, ispravku uočenih nedostataka i prilagođavanje programa zahtevima korisnika.

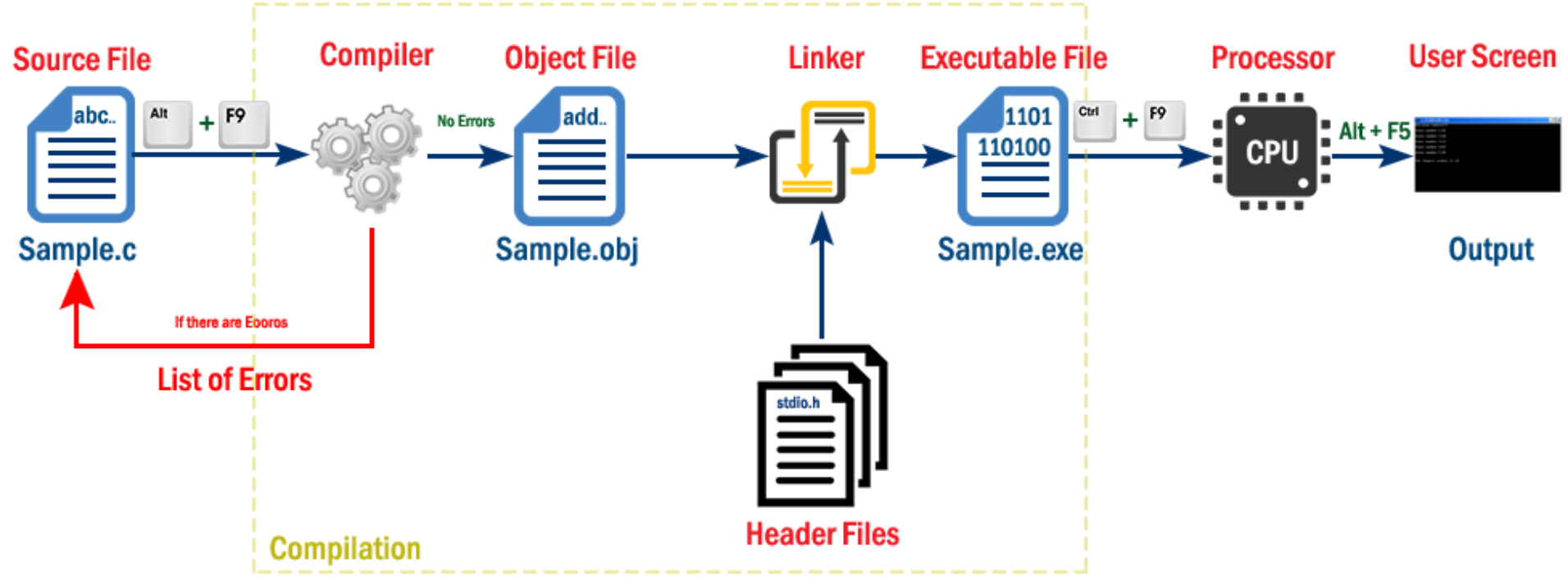
***U slučaju razvoja velikih programskih rešenja postoje različiti tipovi ciklusa kroz koje prolazi svaki program. Izbor najpogodnijeg tipa zavisi od namene programa, broja učesnika na projektu, strategije kompanije koja se bavi razvojem i mnogih drugih parametara*.**

I - Etape u pisanju programa

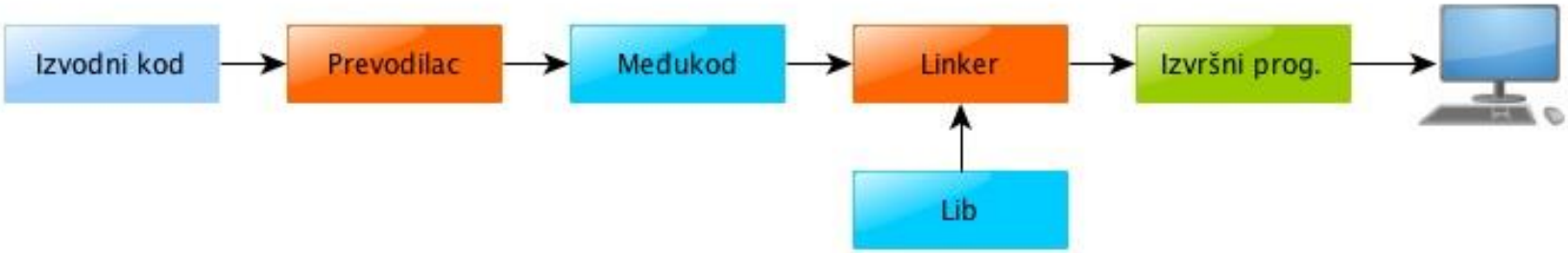


I - Prevođenje programa

- Proces prevođenja (kompajliranja) je proces u kome se izvorni kod napisan u jeziku višeg nivoa prevodi u format instrukcija koje računar razume. Programi koji vrše ovaj proces zovu se **prevodioci**
- Posao prevodioca je da od jednog ili više tekstualnih datoteka koje sadrže **izvorni kod** napisan na jeziku višeg nivoa prevede, spoji i kao izlaz da izvršnu datoteku koja sadrži instrukcije razumljive od strane računara, **izvršni kod**.



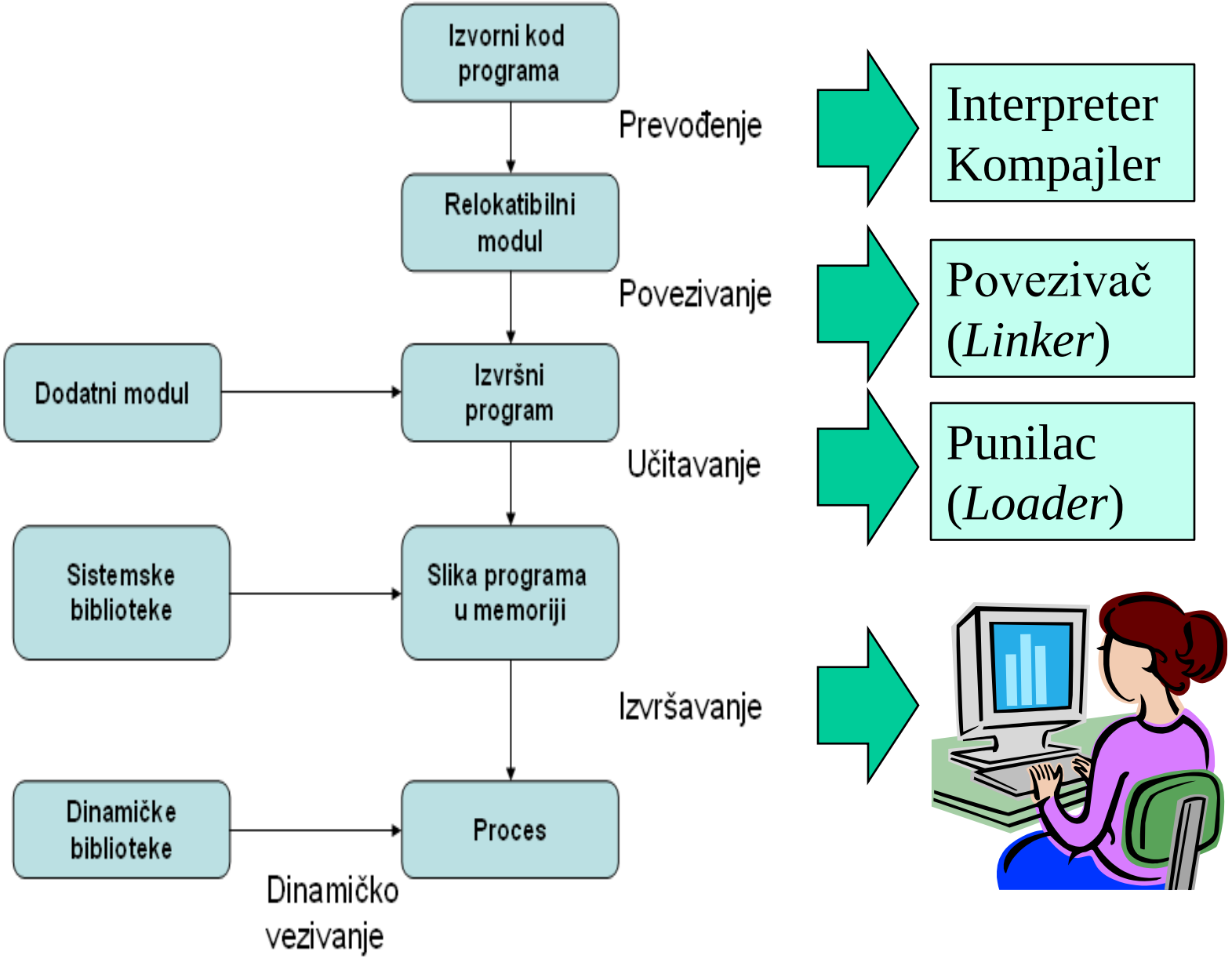
I - Faze prevođenja programa



1. Pomoću **editora teksta** piše se **izvorni kod** na jeziku višeg nivoa
2. Tekst sa **izvornim kodom** smešta se na disk pod nekim imenom
3. Pokreće se **prevodilac** i prosleđuje mu se naziv datoteke sa izv.kodom
4. **Prevodilac** transformiše instrukcije iz izvornih datoteka u mašinske instrukcije i obično pravi neki **posredni binarni fajl**
5. **Posredni binarni fajlovi** se onda spajaju pomoću programa koji se naziva **linker** i generiše se datoteka sa **izvršnim kodom**

- **Izvršna datoteka** se sada može startovati, što u principu znači učitavanje sadržaja instrukcija iz izvršne datoteke u memoriju računara
- Kod nekih programskih jezika kao što je Java, proces se malo razlikuje
- Kao proizvod prevođenja dobija se takozvani **bajtkod** (**bytecode**), kod sličan mašinskom jeziku, koji je namenjen za izvršavanje preko posebnog programa koji se zove **virtuelna mašina**.

I – Proces prevođenja programa



I Istorijat razvoja program.jezika

Globalno razlikujemo pet klasa računarskih jezika:

	Generacija	**Opis**
1.	prva generacija	mašinski jezik
2.	druga generacija	asemblerski jezik
3.	treća generacija	viši program.jezici (HLL)
4.	četvrta generacija	novi jezici 4GL
5.	peta generacija	parametarizacija

I - Istorijat programskih jezika

Prva generacija – mašinski jezik

- Mikroprocesor i drugi logički sklopovi računara imaju svoj vlastiti programski jezik koji se naziva mašinski jezik, a sastoji se od nizova binarnih reči koje predstavljaju neke instrukcije
- Program napisan u mašinskom jeziku nazivamo izvršni program ili izvršni kod budući da ga računar može neposredno izvršiti.
- Mašinski jezik je određen arhitekturom računara i zavisi od procesora.
- Izvršni program je mašinski zavistan, što znači da se kod napisan na jednom računaru može izvršavati jedino na računarima istog tipa.
- Svaka instrukcija, na hardverskom nivou, direktno upravlja radom mašine, tj. pojedinim gradivnim blokovima.
- Instrukcije su numeričke,predstavljene u formi binarnih oblika od 0 i 1
- Programiranje je naporno i podložno velikom broju grešaka
- Efikasnost programiranja je niska
- Programi nerazumljivi korisniku
- Direktno se pristupa resursima mašine
- Veća brzina izvršenja programa i efikasnije korišćenje memorije

I - Istorijat programskih jezika

Druga generacija – asemblerski jezik

- Svaka instrukcija se predstavlja mnemonikom, kao na primer ADD
- Odnos između asemblerskih i mašinskih instrukcija je 1:1
- Programiranje na simboličkom jeziku ima niz nedostataka:
  - ✓ potrebno je vršiti detaljizaciju algoritma tako da elementarnim algoritamskim koracima odgovaraju dejstva simboličkih naredbi,
  - ✓ raznovrsnost računara dovela je do raznovrsnosti simboličkih jezika,
  - ✓ svakom konkretnom računaru odgovara specifičan simbolički jezik.
  - ✓ program napisan na simboličkom jeziku za jedan računar ne može bez veće ili manje prerade da se izvršava na drugom.

Mašinski kod	Asemblerski jezik
0001110110000000	LD$R1.<A
0000000011111111	ADD $R1.<B
0001110000000000	STO$R1.<C
0000000011111100	ADC5
0001111000000000	BDC13
0000000001110010	CDS

I - Istorijat programskih jezika

Treća generacija – HLL (*High Level Language)*

- Programiranje je znatno olakšano.
- Ovi jezici su bliski čovekovom jeziku i operativnoj terminologiji
- Skraćeno je vreme obuke u programiranju i izbegnute su teškoće oko detalja u vezi sa programiranjem na mašinskom ili simboličkom jeziku
- Ovi jezici su nezavisni od strukture samog računara.
- Program napisan na ovom jeziku može da se izvršava na svakom računaru koji ima prevodilac (kompajler) za ovaj jezik,
- Postoji mogućnost izmene programa i iskustva između korisnika jednog problemsko-orjentisanog jezika.
- Kompajler prevodi programske iskaze u odgovarajuće sekvence instrukcija na mašinskom nivou
- U principu jedan iskaz na HLL-u se prevodi u **n** (**n >= 1**) instrukcija na mašinskom (asemblerskom) jeziku
- Jednostavno programiranje, veća efikasnost, lako ispravljanje grešaka
- Nema direktni pristup resursima mašine i neefikasno iskorišćenje memorije i izvršni programi su znatno duži

I - Istorijat programskih jezika

Četvrta generacija – novi jezici 4GL

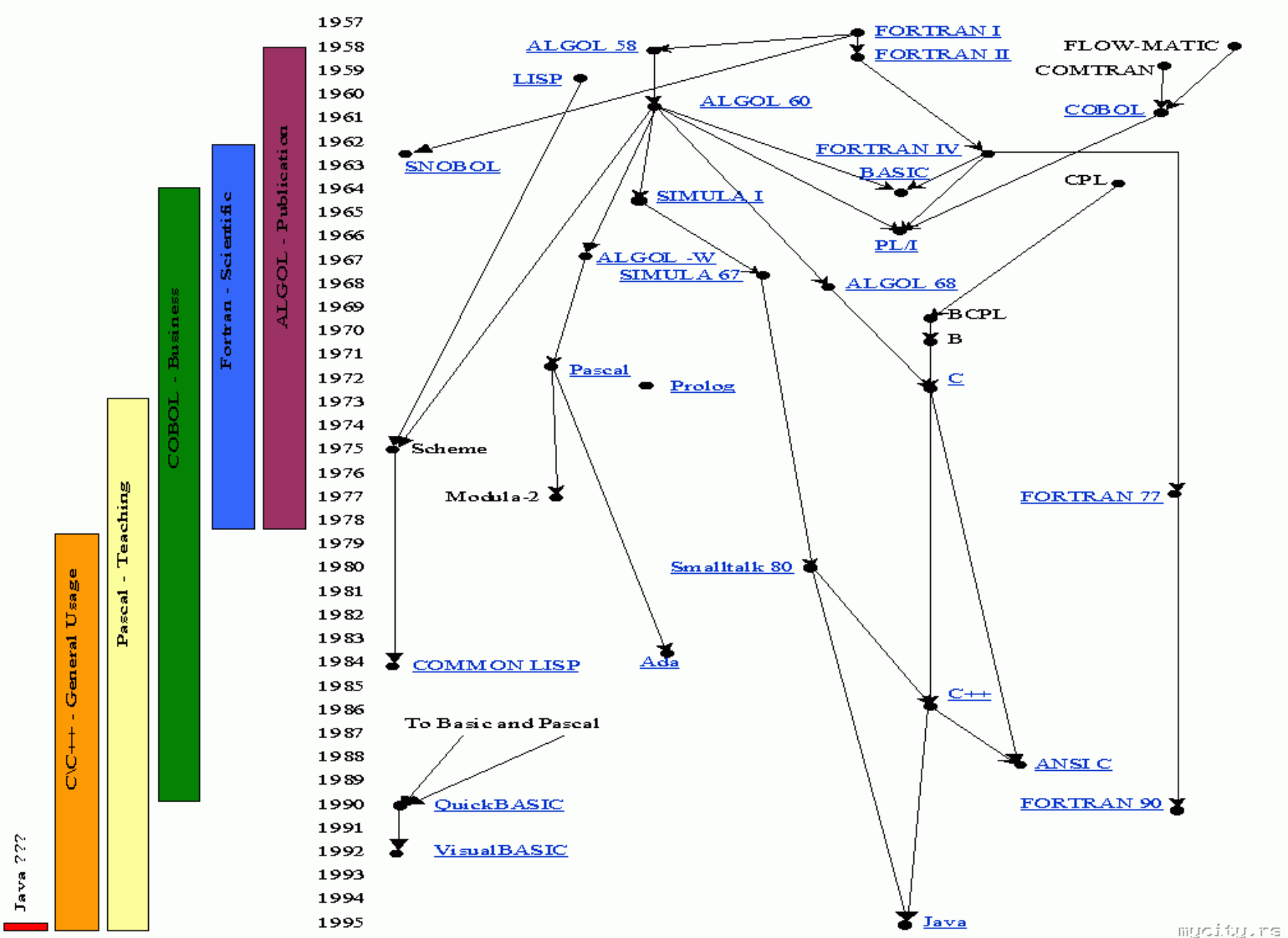
- Dizajnirani kao unapređenje i specijalizacija viših programskih jezika
- Razvoj jezika četvrte generacije počeo je 70 godina dvadesetog veka
- Tačna definicija ove kategorije jezika nije precizno fiksirana
- Jezici četvrte generacije su generalno neproceduralni, dizajnirani su za obradu velike količine podataka bez fokusiranja na individualne bajtove i generalno su specijalizovani za određenu namenu.
- Neki ih smatraju podgrupom takozvanih domenski specifičnih jezika.
- Oni su generalno vezani za rad sa bazama podataka i koriste grafički korisnički interfejs da olakšaju rad korisniku.
- Osnovni tipovi jezika četvrte generacije su između ostalog jezici za obradu podataka kao SAS, SPSS i Strata, takozvani jezici bez koda
- Novi tipovi računarskih jezika se karakterišu sledećim osobinama:
  - ✓ Implementiraju veštačku inteligenciju (primer je LISP)
  - ✓ Jezici za pristup bazama podataka (primer je SQL)
  - ✓ Objektno-orijentisani jezici (primeri su C++, Java i dr.)
  - ✓ Pojava skript jezika

I - Istorijat programskih jezika

Peta generacija

- Peta generacija programskih jezika je grupa jezika koji su bili u razvoju 80-ih godina dvadesetog veka i čija je budućnost neizvesna.
- Osnovna paradigma ovih jezika je da umesto da sam rešava problem programer postavlja parametre za program koji onda na osnovi njih traži rešenje.
- Ideja je da se programer oslobodi razmišljanja o detaljima implementacije, pogotovo implementacije rutinskih algoritama da bi se mogao u potpunosti posvetiti definisanju problema koji treba rešiti.
- U periodu kada su nastajali mnogi su ih smatrali budućnošću programa
- Ubrzo se pokazalo da je definisanje efikasnog algoritma za rešenje problema na bazi njegove definicije samo po sebi izuzetno komplikovan zadatak koji se ne može lako automatizovati.
- Zbog ovog problema većina projekata je napuštena u toku ranih devedesetih.

I Razvoj programskih jezika



Pozicije najzastupljenijih programa

Programming Language	2018	2013	2008	2003	1998	1993	1988
Java	1	2	1	1	18	-	-
C	2	1	2	2	1	1	1
C++	3	4	3	3	2	2	5
Python	4	7	6	12	26	16	-
C#	5	5	7	9	-	-	-
Visual Basic .NET	6	13	-	-	-	-	-
JavaScript	7	9	8	7	21	-	-
PHP	8	6	4	5	-	-	-
Perl	9	8	5	4	3	11	-
Ruby	10	10	9	19	-	-	-
Objective-C	18	3	45	47	-	-	-
Ada	28	15	17	14	7	7	2
Lisp	31	12	14	13	6	4	3

I Rasprostranjenost program.jezika

Feb 2018	Feb 2017	Change	Programming Language	Ratings	Change
1	1		Java	14.988%	-1.69%
2	2	⌄	**C**	**11.857%**	**+3.41%**
3	3	⌃⌃	C++	5.726%	+0.30%
4	5	⌃	Python	5.168%	+1.12%
5	4	⌃⌃	C#	4.453%	-0.45%
6	8	⌄	Visual Basic .NET	4.072%	+1.25%
7	6	⌃⌃	PHP	3.420%	+0.35%
8	7	⌃	JavaScript	3.165%	+0.29%
9	9	⌃	Delphi/Object Pascal	2.589%	+0.11%
10	11	⌃	Ruby	2.534%	+0.38%
11	-	⌃	SQL	2.356%	+2.36%
12	16	⌃⌃	Visual Basic	2.177%	+0.30%
13	15	⌃	R	2.086%	+0.16%
14	18	⌃	PL/SQL	1.877%	+0.33%
15	13	⌄	Assembly language	1.833%	-0.27%
16	12	⌄⌄	Swift	1.794%	-0.33%
17	10	⌄⌄	Perl	1.759%	-0.41%
18	14	⌄⌄	Go	1.417%	-0.69%
19	17	⌄	MATLAB	1.228%	-0.49%
20	19	⌄	Objective-C	1.130%	-0.41%

I Ukupan broj program.jezika ~250

(Visual) FoxPro: FoxPro, Fox Pro, VFP
4th Dimension/4D: 4D, 4th Dimension
ABAP
ABC: ABC (exceptions: -tv -channel)
ActionScript: ActionScript, AS1, AS2, AS3
Ada
Agilent VEE
Algol
Alice: Alice (confidence: 90%)
Angelscript
Apex
APL
Applescript
Arc
AspectJ
Assembly language: Assembly, Assembly language
ATLAS
AutoIt
AutoLISP
Automator
Avenue
Awk: Awk, Mawk, Gawk, Nawk
Bash
Basic: Basic (confidence: 0%)
BBC BASIC
bc
BCPL
BETA: BETA (confidence: 10%)
BlitzMax: BlitzMax, BlitzBasic, Blitz Basic
Boo
Bourne shell: Bourne shell, sh
C shell: Csh, C shell (confidence: 90%)
C#: C#, C-Sharp, C Sharp, CSharp, CSharp.NET, C#.NET
C++
C++/CLI
C-Omega
C: C (exceptions: -"Objective-C")
Caml
Ceylon
CFML: CFML, ColdFusion
cg: cg (confidence: 80%, exceptions: -"computer game" -"computer graphics")
Ch: Ch (exceptions: +ChScite)
CHILL
CIL
CL (OS/400): CL (exceptions: -Lisp), CLLE
Clarion
Clean: Clean (confidence: 43%)
Clipper
Clojure: Clojure, ClojureScript
CLU
COBOL
Cobra
CoffeeScript
COMAL
Common Lisp
Crystal
cT
Curl
D: D (confidence: 90%, exceptions: -"3-D programming" -"DTrace"), dlang
Dart
DCL
Delphi/Object Pascal: Delphi, Delphi.NET, DwScript, Object Pascal, Pascal (confidence: 95%)
DiBOL: DBL, Synergy/DE, DIBOL
Dylan
E: E (exceptions: +specman)
ECMAScript
EGL
Eiffel
Elixir
Elm
Emacs Lisp: Emacs Lisp, Elips
Erlang
Etoys
Euphoria
EXEC
F#: F#, F-Sharp, FSharp, F Sharp
Factor
Falcon
Fantom
Felix: Felix (confidence: 86%)
Forth
Fortran
Fortress
Gambas
GNU Octave
Go: Go, Golang
Gosu
Groovy: Groovy, GPATH, GSQL, Groovy++
Hack
Haskell
Haxe
Heron
HPL
HyperTalk
Icon: Icon (confidence: 90%)
IDL: IDL (exceptions: -corba -interface)
Inform
Informix-4GL
INTERCAL
Io
Ioke
J#
J: J (confidence: 50%)
JADE
Java
JavaFX Script
JavaScript: JavaScript, JS, SSJS
JScript
JScript.NET
Julia
Korn shell: Korn shell, ksh
Kotlin
LabVIEW
Ladder Logic
Lasso
Limbo
Lingo
Lisp
LiveCode: Revolution, LiveCode
Logo: Logo (confidence: 90%, exceptions: -tv)
LotusScript
LPC
Lua: Lua, LuaJIT
Lustre
M4
MAD: MAD (confidence: 50%)
Magic: Magic (confidence: 50%)
Magik
Malbolge
MANTIS
Maple
MATLAB
Max/MSP
MAXScript
MDX
MEL
Mercury
Miva
ML
Modula-2
Modula-3
Monkey
MOO
Moto
MQL4: MQL4, MQL5
MS-DOS batch
MUMPS
NATURAL
Nemerle
Nim: Nim, Nimrod
NQC
NSIS
NXT-G
Oberon
Object Rexx
Objective-C: Objective-C, objc, obj-c
OCaml: Objective Caml, OCaml
Occam
OpenCL
OpenEdge ABL: Progress, Progress 4GL, ABL, Advanced Business Language, OpenEdge
OPL
Oxygene
Oz
Paradox
Pascal: Pascal (confidence: 5%)
Perl
PHP
Pike
PILOT: PILOT (confidence: 50%, exceptions: -"Palm Pilot programming")
PL/I: PL/1, PL/I
PL/SQL
Pliant
PostScript: PostScript, PS
POV-Ray
PowerBasic
PowerScript
PowerShell
Processing: Processing (exceptions: +"sketchbook")
Programming Without Coding Technology: (confidence: 50%), VB (confidence: 50%)
Programming Without Coding Technology, Visual Basic: Visual Basic (confidence: 50%), VB (confidence: 50%), VBA, VB6
PWCT
Prolog
Pure Data: Pure Data, PD
PureBasic
Python
Q
R: R (confidence: 90%, exceptions: +"statistical")
Racket
REBOL
REXX
Ring
RPG (OS/400): RPG (confidence: 80%, exceptions: -role), RPGLE, ILERPG, RPGIV, RPGIII, RPG400, RPGII, RPG4
Ruby
Rust
S-PLUS: S-PLUS (exceptions: +statistical)
S: S (exceptions: +statistical)
SAS
Sather
Scala
Scheme: Scheme (exceptions: -tv -channel)
Scratch
sed
Seed7
SIGNAL: SIGNAL (confidence: 10%)
Simula
Simulink
Slate: Slate (confidence: 57%)
Smalltalk
Smarty
SPARK
SPSS
SQR
Squeak
Squirrel
Standard ML: Standard ML, SML
Stata
Suneido
SuperCollider: SuperCollider (confidence: 80%)
Swift
TACL
Tcl: Tcl/Tk, Tcl
Tex
thinBasic
TOM: TOM (confidence: 50%)
Transact-SQL: T-SQL, Transact-SQL, TSQL
TypeScript
Vala/Genie: Vala, Genie
VBScript
Verilog
VHDL
Visual Basic .NET: Visual Basic .NET, VB.NET, Visual Basic.NET, Visual Basic
WebDNA
Whitespace
Wolfram
X10
xBase
XBase++
Xen
Xojo: REALbasic, Xojo
XPL
XQuery
XSLT
Xtend
yacc
Yorick
Z shell: Z shell, zsh

Hvala na pažnji !!!

Pitanja

? ? ?