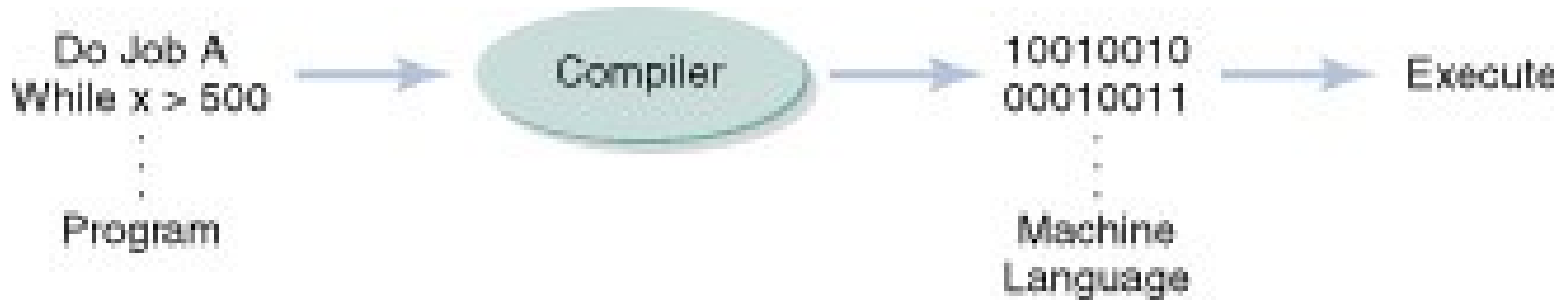


**Skriptno programiranje:
visokonivojsko programiranje
za 21. stoletje**

Prevajalniki in interpreterji

- **Preslikava programa v strojno kodo**
- **Prevajalniki**
 - Preslikajo program v nov program v strojnem jeziku
- **Interpreterji**
 - Postopno prevajanje vsake posamezne vrstice
 - Izvedba posamezne vrstice
 - Pretvorba in izvedba poteka vrstica za vrstico

Kako svoje delo opravi prevajalnik



Kako svoje delo opravi interpreter



Uvod

- **Sistemski programski jeziki se bistveno niso spreminjali že 30 let.**
- **Skriptni jeziki nudijo drugačen način programiranja:**
 - Namenjeni lepljenju.
 - Večinoma brez tipov.
- **Skriptni jeziki so na pohodu:**
 - Izboljšave v skriptni tehnologiji.
 - Hitrejši računalniki.
 - Spremembe v mešanici aplikacij.
- **Ključna misel: stroga tipizacija ni vedno dobra.**

O čem bomo še govorili

- **Zgodovina jezikov za sistemsko programiranje:**
 - Dvig nivoja programiranja.
 - Strožja tipizacija.
- **Skriptni jeziki:**
 - Dvig nivoja programiranja.
 - Ni tipov.
- **Pomen skriptnega programiranja narašča.**
- **Objektno usmerjeno programiranje**

Zbirni jezik

ADDI R4, 21, R2 → **ADDI** **R4** **R2** **21**
101011001000001000000000000010101

- Ukaze opisujemo s simboli namesto z binarnimi števili
- V programu so razvidne posebnosti računalnika:
 - En stavek za vsako računalniško instrukcijo.
 - Eksplicitno upravljanje z registri, skladom ipd.
- Nobene strukture: vse izgleda enako.

Težka tvorba in vzdrževanje velikih programov.

Visokonivojski jeziki

- Pojavijo se v poznih 1950-ih letih: Lisp, Fortran, Algol.
- Vsak stavek se prevede v več inštrukcij:
 $x = 2*(y + z);$
- **Sistemske programski jeziki** so se razvili iz Algola:
 - PL/1, Pascal, C, C++, Java.
 - Razvoj programov hitrejši kot v zbirnem jeziku.
 - Majhna izguba učinkovitosti.
 - Nadomestilo za zbirni jezik.
- **Dve ključni lastnosti sistemskih programskih jezikov:**
 - Višji nivo.
 - Stroga tipizacija.

Visokonivojsko programiranje

- **Podrobnosti so skrite in obravnavane avtomatsko:**
 - Dodeljevanje registrov.
 - Zaporedja klicev procedur.
 - Krmilne strukture: **if**, **while**, itd.
- **Rezultat: Pisanje krajših programov, večja učinkovitost.**
- **Primer meritve:**
 - 8 datotek v jeziku C
 - 5 različnih programerjev
 - 3-7 strojnih instrukcij na programsko vrstico.

Stroga tipizacija

- **Tipizacija: stopnja, do katere je pomen podatka vnaprej omejen na njegovo uporabo.**
- **Von Neumannovi stroji s v bistvu brez tipizacije:**
 - Vsaka beseda lahko vsebuje kakršnokoli vrsto vrednosti.
 - Pomen določa uporaba.
- **Sistemske programski jeziki so strogo tipizirani:**
 - Spremenljivke moramo definirati skupaj z njihovim tipom.
 - Podatki in koda so ločeni; ne moremo tvoriti kode kar mimogrede.
 - Zapisi (records) imajo deklarirano strukturo.
 - Argumenti procedur morajo biti določenega tipa.

Prednosti tipizacije

- **Pojasni uporabo, razlikuje med stvarmi, ki so različne.**
- **Prevajalniki lahko napake odkrijejo bolj zgodaj.**
- **Prevajalniki lahko informacijo o tipih uporabijo za izboljšanje performans:**
 - Tvorba “integer” inštrukcij za “integer” podatke.
 - Brez tipizacije je potrebno več preverjanja med časom izvajanja.

Skriptni jeziki

- **Sistemske programski jeziki:**
 - Tvorba kompleksnih algoritmov in podatkovnih struktur.
 - Delamo z majhnimi bloki.
 - Primeri: delo s podatkovnimi bazami, operacijski sistem, multimedijski strežniki.
- **Skriptni jeziki:**
 - Lepilo: lepljenje obstoječih večjih komponent.
 - Kompleksnost je v povezovanju.
 - Primeri: GUIs, poslovne aplikacije
- **Primeri skriptnih jezikov:**
 - bash, Tcl, Visual Basic, Perl, JavaScript

Skriptni jeziki: brez tipizacije

- Vse je predstavljeno na enak način (n.pr. z nizi).
- Pomen je določen z uporabo.
- Rezultat: lepljenje, enostavna ponovna uporaba.
- Primer: spremenljivke Visual Basic.
- Primer: Filtri LINUX :
`select | grep blabla | wc`
- **Stroga tipizacija otežuje lepljenje in ponovno uporabo:**
 - Tipi, vmesniki omejujejo uporabo.
 - Potrebna je konverzija kode in ponovno prevajanje.
 - Binarne aplikacije to otežujejo.

Primer Tcl/Tk

- **Tvorba preprostega gumba:**

```
button .b -text Hello! -font {Times 16} \  
-command {puts hello}
```

- **Microsoft Foundation Classes terjajo 25 vrstic.
Koda za nastavljanje fonta:**

```
CFont *fontPtr = new CFont();  
fontPtr->CreateFont(16, 0, 0, 0, 700, 0, 0, 0,  
ANSI_CHARSET, OUT_DEFAULT_PRECIS,  
CLIP_DEFAULT_PRECIS, DEFAULT_QUALITY,  
DEFAULT_PITCH|FF_DONTCARE, "Times New Roman");  
buttonPtr->SetFont(fontPtr);
```

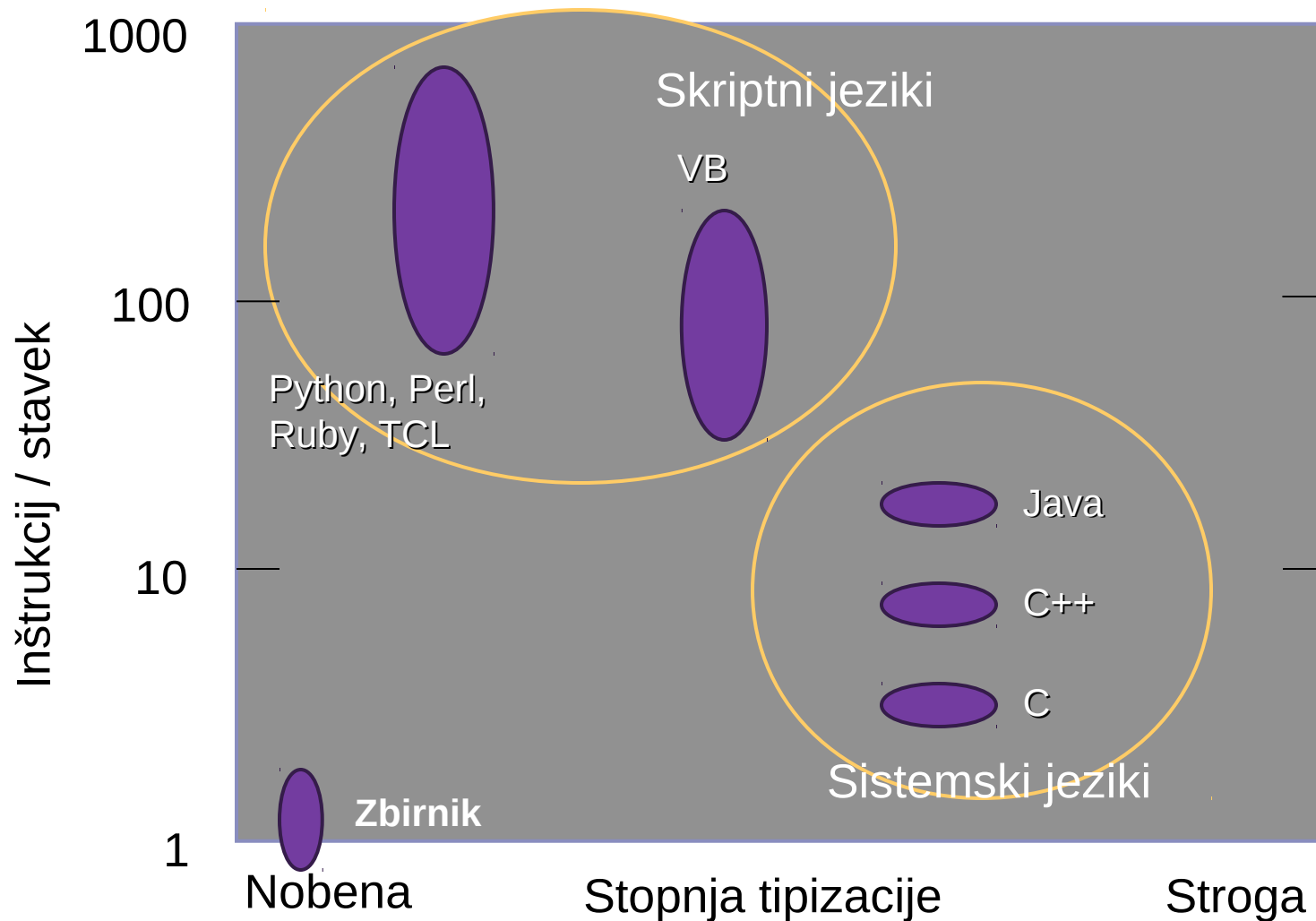
- **Java zahteva 7 vrstic.**

Še o skriptnih jezikih

- **Interpretirani:**
 - Hiter razvoj.
 - Mimogrede razširimo aplikacije.
 - Manj učinkoviti od sistemskih programskih jezikov.
- **Visok nivo: Vsak stavek opravi več dela.**
 - Skripte: 100-1000 inštrukcij/stavek.
 - Sistemsko programiranje: 5-10 inštrukcij/stavek.
- **Hiter razvoj:**
 - 5-10x hitrejši kot s sistemskimi programskimi jeziki (za lepljenje nalog).
- **Še varni: odkrivanje napak med izvajanjem .**

Manj učinkovita uporaba računalnikov, bolj učinkovita uporaba ljudi.

Nivoji jezikov in produktivnost



Različna orodja za različne naloge

- **Sožitje systemskega programiranja in skript:**
 - Komponente tvorimo s sistemskimi programskimi jeziki.
 - Komponente lepimo s skriptnimi jeziki.
- **Večina platform vsebuje oboje:**

Platforma	Sis. Prog.	Skriptni j.
OS/360	BAL, PL/1	JCL
LINUX Tcl	C, c++	bash, csh, Perl,
MS Windows	C++, c#	Visual Basic
Internet	Java	JavaScript

Izbira jezika

- Ali aplikacija uporablja kompleksne algoritme in podatkovne strukture?
- Ali aplikacija obdeluje velike množice podatkov (>10,000 enot)?
- Ali so funkcije aplikacije dobro definirane, fiksirane?

Če da, potem izberi sistemski programski jezik.

- Ali je glavna naloga povezovanje komponent oziroma aplikacij?
- Ali aplikacija rokuje z različnimi stvarmi?
- Ali se funkcije aplikacij hitro spreminjajo?
- Ali mora biti aplikacija razširljiva?
- Ali aplikacija veliko dela z nizi?

Če da, potem izberi skriptni jezik.

Skriptni jeziki se razvijajo

- UNIX shell scripting
- Rexx
- Tcl, TK
- Perl
- Python
- PHP
- Ruby

Nekateri imajo **visokonivojske, objektno usmerjene značilnosti**, kar jih dela za močna orodja za razvoj aplikacij.

Skriptno programiranje je na pohodu

- **Nove aplikacije:**
 - GUI – grafični uporabniški vmesniki.
 - Internet.
 - Poslovne aplikacije.
 - Ogradja komponent (Component frameworks).
- **Tehnologija skriptnega programiranja se izboljšuje:**
 - Hitrejši računalniki.
 - Boljši jeziki.
- **Skupnost programerjev se spreminja:**
 - Več priložnostnih programerjev.
- **Večina današnjih aplikacij je skriptnih.**

Bodočnost systemskega programiranja

- **Ali bodo systemski programski jeziki “izginili”, kot so zbirni jeziki?**
- **Skriptni jeziki niso primerni za nekatere stvari:**
 - Kompleksni algoritmi in podatkovne strukture.
 - Aplikacije, kjer je performansa kritična.
- **Systemske programske jezike bomo še rabili za:**
 - Tvorbo komponent.
 - Strežnike s fiksnimi funkcijami in operacijske sisteme.
- **Pri večini drugih namenov pa bodo skriptni jeziki nadomestili systemsko programiranje.**

Objektno usmerjeno programiranje

- **Vroča tema v programskih jezikih:**
 - Stroga tipizacija, dedovanje.
 - Skrajšanje časa razvoja, povečanje ponovne uporabe?
- **Resnične prednosti so morda skromne (20-30%?).**
- **Ne dviguje nivoja programiranja ali pospešuje ponovne uporabe:**
 - Še vedno delamo z majhnimi enotami.
 - Še vedno prevajamo.
 - Strogo tipizirani vmesniki otežujejo ponovno uporabo.
- **Uporaba dedovanja ima tudi slabosti:**
 - Povezuje skupaj implementacije razredov.
 - Razumeti moramo hierarhijo razredov.
 - Razrede ne moremo kar neodvisno ponovno uporabljati.

Še o objektih

- **Glavna prednost objektov:**
 - Enkapsulacija.
 - Standardni protokoli (dedovanje vmesnikov).
- **Prednosti OO srečamo tudi v skriptnih jezikih:**
 - Python
 - Perl 5.0
 - Object Rexx
 - Incr tcl
 - JavaScript
- **Objekti v skriptnih jezikih so tipično netipizirani.**

Zaključki

- **Skriptno programiranje je v osnovi različno od systemskega programiranja:**
 - Brez tipizacije, interpretirano.
 - Namenjeno lepljenju aplikacija.
 - Žrtvuje čas izvajanja.
 - 5-10x hitrejši razvoj aplikacij.
- **Cilj: dvig nivoja programiranja:**
 - Stroga tipizacija otežuje ponovno uporabo.
 - API so slabi!
- **Skriptno programiranje je že danes pomembno in bo še bolj v naslednjih 10 letih.**

Skriptno programiranje in agenti

- **Inteligentni vidiki:**
 - Inferenčni stroji, platforme za druge programske jezike (sistemski programski jeziki? Lisp?).
 - S skriptnimi jeziki lahko specificiramo pravila in aktivner podatke.
 - Je inteligenca visoko povezljiva?
- **Mobilni vidiki, primerni za skriptno programiranje:**
 - Potrebna je integracija informacij iz različnih virov.
 - Dinamičnost.
 - Prenosljivost.

Za konec...

“Uporabi pri svojem delu najboljša orodja. V svoji “orodni omarici” imej tako skriptne kot sistemske programske jezike”

- Bill Venners

<http://www.artima.com/commentary/langtool.html>

