Programiranje za umetnike 1

~ 3 ~

Staša Vujičić Stanković

Računarski sistemi

Struktura savremenog računarskog sistema

- Hardver
- Softver

Hardver

- Hardver računarskog sistema se u opštem slučaju sastoji od:
  - centralne procesorske jedinice (CPU) – obrada podataka
  - operativne memorije – skladište podataka i programa
  - perifernih uređaja
    - spoljašnje memorije
    - ulaznih uređaja
    - izlaznih uređaja
- Nabrojane komponente su povezane magistralama
- Magistrala obuhvata provodnike koji povezuju uređaje
  ali i čipove koji kontrolišu protok podataka

Hardver čine opipljive, fizičke komponente računara

Procesor

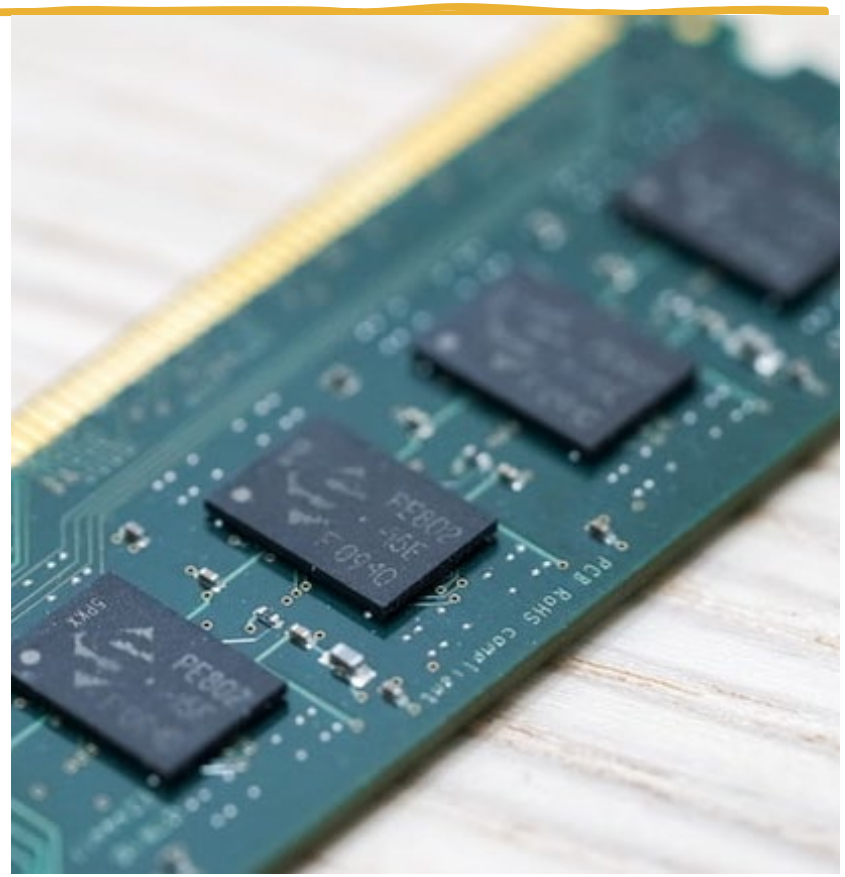
- Sastoji se od kontrolne jedinice i aritmetičko logičke jedinice
- Sadrži registre u kojima se privremeno smeštaju podaci
- Registri širine 8, 16, 32 ili 64 bita
- Brzina procesora meri se u milionima operacija u sekundi MIPS, tj. u broju operacija u pokretnom zarezu u sekundi FLOPS (10GFLOPS u sekundi)
- Više jezgara – paralelno izvršavanje (1, 2 ili 4)
- Radni takt GHz

Memorija

- Memorija je linearno uređeni niz registara (najčešće bajtova), pri čemu svaki registar ima svoju adresu.
- Osnovne karakteristike memorije: kapacitet, vreme pristupa, protok
- Unutrašnje memorije – koriste se samo kada je računar uključen
- Spoljne memorije – skladištenje podataka kada je računar isključen
- Procesor komunicira sa unutrašnjom memorijom, podaci se prebacuju iz spoljne u unutrašnju

Memorijska hijerarhija

- Registri procesora
- Keš
- RAM
- ROM/BIOS
- USB diskovi
- Hard diskovi
- CD, DVD, Blu-ray, magnetne trake



Ulazni i izlazni uređaji

- Ulazni uređaji:
  - tastatura
  - miš
  - ekran osetljiv na dodir
  - skener
  - džojstik…
- Izlazni uređaji:
  - monitori (LED, CRT)
  - štampači (laserski, ink-džet, 3D)
- Veze:
  - bežična veza (BlueTooth)
  - kablovi preko USB priključaka

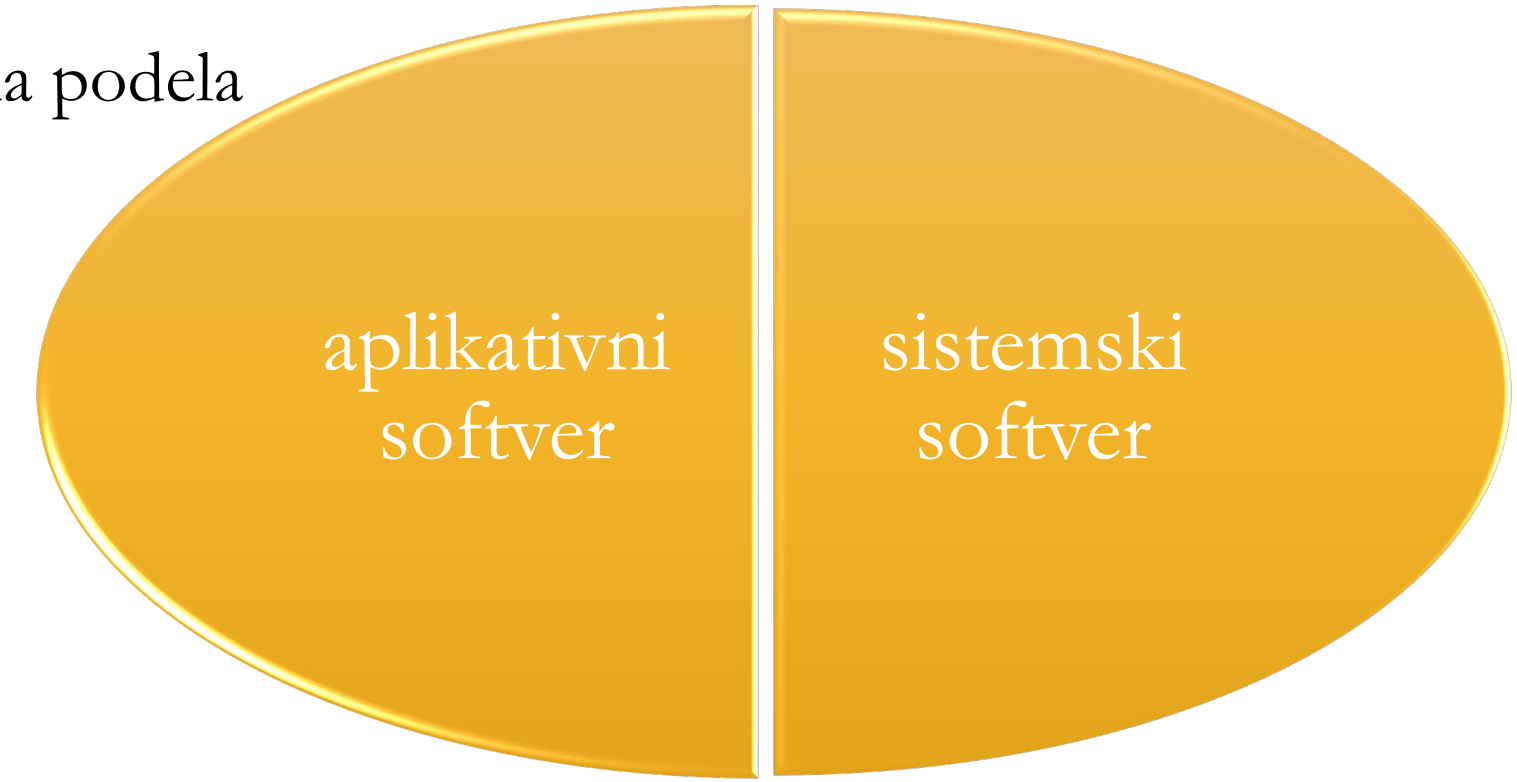
Softver

Softver čine računarski programi i prateći podaci koji određuju izračunavanja koje vrši računar

- Program specifikuje koje operacije treba izvršiti da bi se rešio neki zadatak
- Prvi računari – samo mašinski zavisni jezici
- Polovinom 1950ih godina – jezici višeg nivoa

Klasifikacija softvera

- Osnovna podela je na



Aplikativni softver

- **Aplikativni softver** je softver koji krajnji korisnici računara direktno koriste u svojim svakodnevnim aktivnostima.
  Primera radi:
  - pregledač Veba, klijenti elektronske pošte
  - multimedijalni softver – programi za reprodukciju i obradu slika, zvuka i video sadržaj
  - kancelarijski softver
  - video igre

Sistemski softver

- **Sistemski softver** je softver čija je uloga da kontroliše hardver i pruža usluge aplikativnom softveru
- Granica između sistemskog i aplikativnog softvera nije kruta i postoje programi za koje se može smatrati da pripadaju obema grupama

Sistemski softver

- operativni sistem
- različiti uslužni programi
  - editori teksta
  - alati za programiranje (prevodioci, dibageri, profajleri, integrisana okruženja)

Operativni sistem

- Korisnici operativni sistem često identifikuju sa izgledom ekrana tj. sa programom koji koriste da bi pokrenuli svoje aplikacije i organizovali dokumente.
- To je korisnički interfejs ili školjka – tanak sloj na vrhu operativnog sistema
- Najveći i najznačajni deo operativnog sistema naziva se jezgro
- Jezgro kontroliše i apstrahuje hardver, sinhronizuje rad više programa, raspoređuje procesorsko vreme i memoriju, brine o sistemu datoteka na spoljašnjim memorijama itd.

Teme

Uvod u skript programiranje

Skript jezici

Programske biblioteke

Programska okruženja

Skript jezici

Popularnost skript jezika

- Skript programiranje je važan stil programiranja u velikom broju različitih domena
- Prema trenutnom TIOBE indeksu,
  trećina najpopularnijih programskih jezika pripadaju skript paradigmi
- Python je skript jezik opšte namene čija popularnost sve više raste

TIOBE Index for October 2021

October Headline: Python programming language number 1!

For the first time in more than 20 years we have a new leader of the pack: the Python programming language. The long-standing hegemony of Java and C is over. Python, which started as a simple scripting language, as an alternative to Perl, has become mature. Its ease of learning, its huge amount of libraries, and its widespread use in all kinds of domains, has made it the most popular programming language of today. Congratulations Guido van Rossum! Proficiat! -- *Paul Jansen CEO TIOBE Software*

The TIOBE Programming Community index is an indicator of the popularity of programming languages. The index is updated once a month. The ratings are based on the number of skilled engineers world-wide, courses and third party vendors. Popular search engines such as Google, Bing, Yahoo!, Wikipedia, Amazon, YouTube and Baidu are used to calculate the ratings. It is important to note that the TIOBE index is not about the *best* programming language or the language in which *most lines of code* have been written.

The index can be used to check whether your programming skills are still up to date or to make a strategic decision about what programming language should be adopted when starting to build a new software system. The definition of the TIOBE index can be found here.

Oct 2021	Oct 2020	Change	Programming Language	Ratings	Change
1	3	↑	Python	11.27%	-0.00%
2	1	↓	C	11.16%	-5.79%
3	2	↓	Java	10.46%	-2.11%

October Headline: C# is getting closer to Java

The gap between C# and Java never has been so small. Currently, the difference is only 1.2%, and if the trends remain this way, C# will surpass Java in about 2 month's time. Java shows the largest decline of -3.92% and C# the largest gain of +3.29% of all programming languages (annually). The two languages have always been used in similar domains and thus have been competitors for more than 2 decades now. Java's decline in popularity is mainly caused by Oracle's decision to introduce a paid license model after Java 8. Microsoft took the opposite approach with C#. In the past, C# could only be used as part of commercial tool Visual Studio. Nowadays, C# is free and open source and it's embraced by many developers. There are also other reasons for Java's decline. First of all, the Java language definition has not changed much the past few years and Kotlin, its fully compatible direct competitor, is easier to use and free of charge. -- *Paul Jansen CEO TIOBE Software*

The TIOBE Programming Community index is an indicator of the popularity of programming languages. The index is updated once a month. The ratings are based on the number of skilled engineers world-wide, courses and third party vendors. Popular search engines such as Google, Bing, Yahoo!, Wikipedia, Amazon, YouTube and Baidu are used to calculate the ratings. It is important to note that the TIOBE index is not about the *best* programming language or the language in which *most lines of code* have been written.

The index can be used to check whether your programming skills are still up to date or to make a strategic decision about what programming language should be adopted when starting to build a new software system. The definition of the TIOBE index can be found here.

Oct 2023	Oct 2022	Change	Programming Language	Ratings	Change
1	1		Python	14.82%	-2.25%
2	2		C	12.08%	-3.13%
3	4	^	C++	10.67%	+0.74%

Nastanak skript jezika

- Prvi skript jezik je bio komandni jezik **sh** (Unix shell)
- On je započeo kao kolekcija komandi koje se interpretiraju kao pozivi sistemskih funkcija, npr. upravljanje fajlovima, listanje fajlova, filtriranje fajlova…
- Vremenom su na to dodate promenljive, kontrola toka, funkcije i razne druge mogućnosti, i na kraju je to rezultovalo kompletnim programskim jezikom
- U skladu sa tim, skript jezici su bili korišćeni da se zapiše u fajlu lista komandi – **skript**, koja treba da se interpretira

Skript jezici

- Skript jezik je programski jezik koji služi za pisanje skriptova
- Skript je niz komandi koje se izvršavaju u odgovarajućem izvršnom okruženju (engl. run-time environment)
- Neke skriptove odlikuje da mogu biti izvršeni bez interakcije sa korisnikom (na primer, skriptovi koji se izvršavaju na serveru), dok za neke skriptove važi da se izvršavaju interaktivno u komunikaciji sa korisnikom (na primer, u okviru REPL okruženja (engl. read-eval-print-loop))

Skript jezici

- Upotreba modernih skript jezika se može lako i brzo savladati, čak i od strane početnika ili ljudi koji po struci nisu programeri.
- Sintaksa i upotreba nekih starijih skript jezika je veoma kompleksna i nije za početnike.

Skript jezici

- Skript jezici su u ekspanziji: jako puno događanja na polju razvoja programskih jezika u prethodne dve decenije je u okviru razvoja skript jezika
- Skript jezici imaju veliki broj domena primene, a posebna popularnost dolazi iz upotrebe u veb programiranju (PHP, JavaScript, Perl...)
- U prvobitnom obliku pojavljuju se kao komandni jezici operativnih sistema (npr. bash), danas imaju najrazličitije primene
- Skript jezici mogu imati specifičan domen primene,
  ali mogu biti i jezici opšte namene (npr. Python)

Povezivanje aplikacija

glue languages

Povezivanje aplikacija

- Tradicionalni programski jezici (C/C++, Java) su namenjeni za razvoj samostalnih aplikacija koje imaju za cilj da prime neku vrstu ulaza i na osnovu nje generišu odgovarajući izlaz.
- Međutim, upotreba računara često zahteva manilpulaciju i koordinaciju različitih programa.
- Ručno sprovođenje tih poslova je naporno i sklono greškama tako da se u takvim situacijama najčešće koriste skript jezici.
- Koordinaciju drugim programima moguće je ostvariti i u tradicionalnim programskim jezicima, ali to nije uvek jednostavno.

Primeri koordinacije rada različitih programa

Primer 1 **Fotografija**. Fotograf treba da skine fotografije sa digitalnog fotoaparata, konvertuje u odgovarajući format, rotira slike po potrebi, napravi manje koji su pogodne za brzo razgledanje, indeksira ih po vremenu, temi, napravi bekap na udaljenoj arhivi i ponovo inicijalizuje memoriju...

Primer 2 **Sistem za obračun plata**. Obračunavanje vremena sa kartica, paprinih izveštaja i unosa sa tastature, manipulacija bazama podataka, poštovanje pravnih i institucionalnih regulativa, priprema poreza, doprinosa i medicinskog osiguranja, pravljenje papirnih evidencija za arhivu...

Primer 3 **Kreiranje dinamičkih veb stranica**. Autentikacija i autorizacija, komunikacija sa udaljenim uređajem, manipulacija sa slikama, komunikacija sa serverom, čitanje i pisanje HTML-a…

Poređenje

Tradicionalni jezici

- Tradicionalni jezici (C/C++, Java) adresiraju
  - efikasno izvršavanje
  - lako održavanje
  - portabilnost i
  - statičko otkrivanje grešaka.

Skript jezici

- Skript jezici imaju za cilj da adresiraju
  - fleksibilnost razvoja
  - brz razvoj
  - lokalnu prilagodiljivost i
  - dinamičke provere.

Poređenje

Tradicionalni jezici

- Sistem tipova je obično izgrađen oko primitivnih koncepta kao što su celobrojne vrednosti fiksnih veličina, brojevi u pokretnom zarezu, karakteri i nizovi.

Skript jezici

- Njihovi sistemi tipova, zbog toga teže da podrže programske koncepte višeg nivoa, kao što su tabele, katalozi, liste, datoteke...

Karakteristike skript jezika

Karakteristike skript jezika

- Skript paradigma je često specifična kombinacija drugih paradigmi, kao što su:
  - imperativna paradigma – prisutnost promenljivih, naredbi, petlji, if, switch/case, funkcija/procedura
  - objektno-orijentisana paradigma – prisutnost klasa, objekata, nasleđivanja
  - funkcionalna paradigma – lambda funkcije, funkcije višeg reda
  - kroz biblioteke, može biti prisutna podrška i drugim programskim paradigmama

Karakteristike skript jezika

- Nije uvek lako napraviti razliku između skript-jezika i
  drugih programskih jezika
  ali ipak postoje određene karakteristike na osnovu kojih se mogu izdvojiti
- Primeri skript jezika:

  Unix Shell (sh), Bash, PowerShell, JavaScript, TypeScript,
  PHP, Perl, Python, XSLT, Tcl, VBScript, Lua, Ruby...

Karakteristike skript jezika

- Interaktivno korišćenje/serijska obrada
- Skraćen zapis
- Deklaracije i pravila dosega
- Dinamičko tipiziranje
- Sistemske funkcije
- Manipulacija stringovima i poklapanje obrazaca
- Tipovi podataka visokog nivoa

Interaktivno korišćenje i skraćeni zapis

- Većina skript jezika omogućava interaktivno korišćenje, ali ima i jezika koji su planirani za obradu bez komunikacije sa korisnikom (npr. skriptovi na serverskoj strani)
- Većina skript jezika je interpretatorskog tipa i omogućava obradu linije za linijom ulaza
  - To znači da je izvršavanje skripta moguće čak i kada postoji neka sintaksna greška u kodu: izvršavanje se onda izvodi sve do linije u kojoj se nalazi ta greška.
  - S druge strane kompilirani jezici zahtevaju da je kôd u potpunosti sintaksno ispravan da bi mogao da se prevede i zatim izvrši.

Interaktivno korišćenje /serijska obrada

- Neki skript jezici zahtevaju da se sve komande učitaju pre nego što počne obrada, ali takvi su u manjini (npr. Perl)
  - Perl skriptovi se najpre kompiliraju do međukoda, a zatim se taj međukod interpretira
  - Ove dve faze se izvršavaju uvek zajedno, tj. međukod se ne čuva u zasebnoj datoteci već se pravi svaki put iznova

Skraćeni zapis

- Skraćeni zapis se koristi radi brzog razvoja i interaktivnog korišćenja
- Skraćeni zapis se može ostvariti na različite načine,
  ali većina skript jezika izbegava korišćenje deklaracija
  i opštih delova koda koji su prisutni u tradicionalnim jezicima

Skraćeni zapis

primer

Perl i Python

```
print "Hello world!\n"
```

Java

```
class Hello {
    public static void main(String[] args)
    {
        System.out.println("Hello world!");
    }
}
```

Deklaracije, pravila dosega i dinamičko tipiziranje

- Promenljive se obično ne deklarišu
- Postoje jednostavna pravila dosega. Na primer:
  - U nekim jezicima, sve promenljive su globalne
    (npr. Perl, mada se po potrebi mogu ograničiti dosezi promenljivima)
  - U nekim jezicima, sve promenljive su lokalne
    (npr. PHP, Tcl, ali se po potrebi mogu eksplicitno napraviti globalne promenljive)
  - Python – svaka promenljiva je lokalna za blok u kome joj je dodeljena vrednost

Deklaracije i pravila dosega

- Pravila dosega za tradicionalne jezike su često veoma kompleksna i ne mogu se opisati u jednoj do dve rečenice (kao što je to za skript jezike).
- Na primer, pogledajte pravila dosega za C ili C++.

Dinamičko određivanje tipova promenljivih

Većina skript jezika dinamički određuje tipove podataka

If it looks like a duck, swims like a duck, and quacks like a duck, then it probably is a duck ☺

- U nekim jezicima, tip se proverava neposredno pred korišćenje (PHP, Ruby, Python)
- U nekim jezicima, tip će se interpretirati drugačije u različitim kontekstima (Perl, Tcl)

```
$a = "4"
print $a . 3 . "\n" # '.' je nadovezivanje: 43
print $a + 3 . "\n" # '+' je sabiranje: 7
```

Dinamičko određivanje tipova

- Dinamičko određivanje tipova je posebno važno kada se jezik koristi kao lepak za druge aplikacije
- U takvom scenariju skript mora da bude u mogućnosti da prihvati i prosledi podatke iz različitih sistema koji mogu biti napisani u različitim programskim jezicima a koji mogu čak imati i nekompatibilne sisteme tipova
- Skriptovi obrađuju različite vrste podataka, počevši od podataka iz formi, baza podataka, tabela ili veb stranica. U tom smislu, jednostavan statički sistem tipova bi bio nefleksibilan, a napredni statički sistemi tipova koji uključuju parametarski polimorfizam ili klasne šablone bi zakomplikovao i usporio razvoj.

  => Dinamičko određivanje tipova je pravo rešenje.

Dinamičko određivanje tipova

- Dinamičko određivanje tipova omogućava da se preskoče deklaracije tipova,
  ali odsustvo informacije o tipovima čini skript jezike manje čitljivim
- Dinamičko određivanje tipova preskače potrebu za komplikovanim sistemom tipova,
  ali greške u radu sa tipovima mogu da se detektuju samo u fazi izvršavanja,
  a neke greške mogu da ostanu nedetektovane jako dugo ili zauvek
  (npr. ukoliko ne dovode do pada pri radu, već samo do blago izmenjenog ponašanja)
- Podsetimo se da statičko određivanje tipova omogućava kompajleru da utvrdi da se
  neke greške u tipovima nikada neće desiti u fazi izvršavanja,
  kao i da utvrdi da će se neke greške možda desiti

Dinamičko određivanje tipova

- Neki skriptovi se napišu da se koriste samo jednom, ali mnogi skriptovi se koriste puno puta.
- Ako skript treba da se koristi puno puta, veoma je važno da bude lak za održavanje, kao što su to programi pisani u tradicionalnim jezicima
- Da bi bio lak za održavanje, potrebno je da bude
  - čitljiv
  - dobro dokumentovan
  - dizajniran tako da se može lako menjati i
  - da sadrži što manje grešaka
- Dizajneri modernih skript jezika, kao što je to npr. Python, razumeju važnost održavanja, ali i dalje nije lako pomiriti ciljeve koji su međusobno konfliktni:
  - fleksibilnost u implementaciji i
  - eliminisanje grešaka tipova u fazi izvršavanja

Literatura

- Slajdovi preuzeti od prof. Milene Vujošević Janičić sa predavanja Dizajn programskih jezika – Skript jezici.

Hvala

Staša Vujičić Stanković

stasa.vujicic.stankovic@math.rs

www.matf.bg.ac.rs/~stasa.vujicic.stankovic