Gimnazija „Vuk Karadžić“

Trstenik, Vuka Karadžića 11

Seminarski rad iz primene računara

GENERACIJE PROGRAMSKIH JEZIKA

Profesor:

Bojana Stevanović, prof

Učenik:

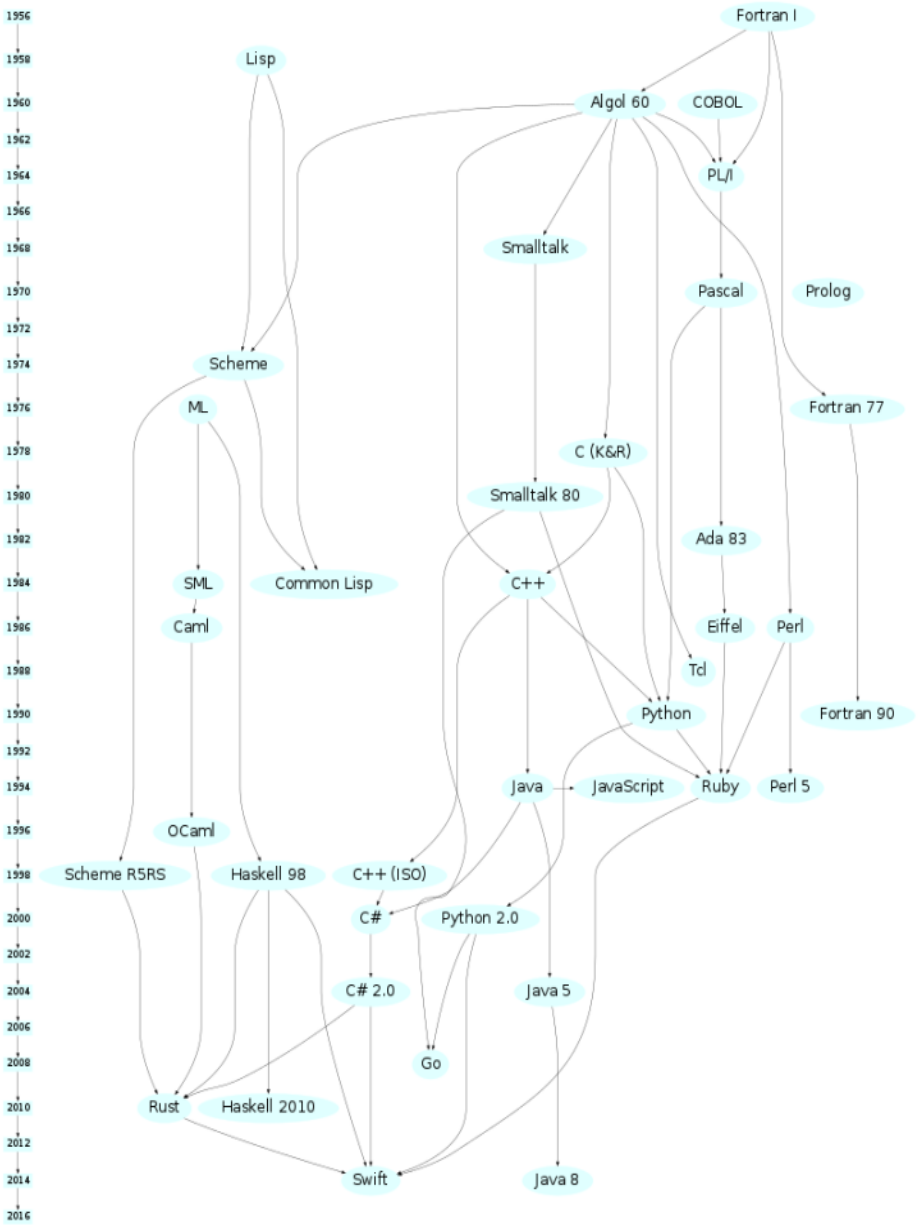
Martin Gobeljić, I-1

Trstenik, januar 2022

Sadržaj

1. Uvod

Programski jezici su počeli i nastavili da se razvijaju uporedo sa računarima, tj. sa hardverom. Programiranje u današnjem smislu je nastalo pojavom Fon Nojmanovog tipa računara. Na prvim racunarima tog tipa moglo je da se programira samo na mašinski zavisnim programskim jezicima, a od polovine 1950-ih nastali su jezici viseg nivoa koji su drastično olakšali programiranje. Prvi programski jezici zahtevali su od programera da bude upoznat sa najfinijim detaljima računara koji se programira. Programi napisani za jedan racunar mogli su da se izvršavaju isključivo na istim takvim računarima i prenosivost programa nije bila moguca. Viši programski jezici namenjeni su ljudima, a ne mašinama i sakrivaju detalje konkretnih racunara od programera. Oni omogućavaju prenosivost programa.



2. Prva generacija programskih jezika

Programski jezik prve generacije (1GL) je programski jezik na nivou mašine.

2.1. O programskim jezicima I generacije

Jezik prve generacije (programski) (1GL) je grupa programskih jezika koji su jezici na mašinskom nivou koji se koriste za programiranje računara prve generacije. Prvobitno, nijedan prevodilac nije korišćen za sastavljanje ili sklapanje jezika prve generacije. Uputstva za programiranje prve generacije uneta su preko prekidača na prednjoj ploči računarskog sistema.

Instrukcije u 1GL su napravljene od binarnih brojeva, predstavljenih 1 i 0. Ovo čini jezik pogodnim za razumevanje mašine, ali mnogo težim za tumačenje i učenje od strane programera.

Slika 2.1. Mašinski kod

Jezici prve generacije su u velikoj meri prilagođeni specifičnom računaru i CPU-u, pa je prenosivost koda značajno smanjena u poređenju sa jezicima višeg nivoa.

Savremeni programeri i dalje povremeno koriste kod na mašinskom nivou, posebno kada programiraju funkcije nižeg nivoa sistema, kao što su drajveri, interfejsi sa firmverom i hardverski uređaji. Savremeni alati kao što su kompajleri izvornog koda se koriste za proizvodnju mašinskog nivoa iz jezika višeg nivoa.

3. Druga generacija programskih jezika

Termin je skovan da pruži razliku od mašinskih nezavisnih programskih jezika treće generacije (3GL) (kao što su COBOL, C ili JavaScript) i ranijih programskih jezika prve generacije (mašinski kod)

3.1. O programskim jezicima II generacije

Programski jezici druge generacije – Asemblerski jezici imaju sledeća svojstva:

- Linije unutar programa direktno odgovaraju komandama procesora
- Programer može čitati i pisati kod. Da bi se pokrenuo na računaru, mora se konvertovati u mašinski čitljiv oblik, proces koji se zove asembler
- Jezik je specifičan za određenu porodicu procesora i okruženje

Jezici druge generacije se ponekad koriste za delove kernela ili drajvera uređaja, a ponekad se koriste u video igrama, grafičkim programima i drugim intenzivnim programima.

U savremenim programima, asemblerski jezici druge generacije se retko koriste.Programiranje na jezicima druge generacije može doneti prednosti u pogledu brzine, ali nekoliko nedostataka je dovelo do njegovog opadanja:

Programiranje je izraženo u vidu pojedinačnih instrukcija procesora, a ne logike višeg nivoa.

Memorija niskog nivoa i detalji o hardveru moraju se ručno upravljati, što je često sklono greškama.

Programi zavise od mašine, tako da se za svaku ciljnu arhitekturu mašine moraju napisati različite verzije.

Velika većina programa je napisana u programskom jeziku treće generacije ili programskom jeziku četvrte generacije. Glavna prednost asemblera, brzina, degradirala se činjenicom da dobro napisan C kod često može biti brz ili čak brži od ručno pisanog asemblera

Jezici druge generacije su možda najznačajniji na svom mestu u istoriji računarstva. Dugo vremena su asemblerski jezici druge generacije bili jedina dobra opcija za razvoj za mnoge mašine, kao što su NES ili Commodore 64. Druga generacija jezika predstavljala je ogroman korak udaljavanja od tradicije programera koji se prilagođavaju potrebama mašine i prvi korak ka tome da mašina bude prilagođena programeru, fenomen koji će se ponoviti u svim narednim generacijama programskih jezika.

3.2. Prednosti i mane

Prednosti:

- Programiranje na jezicima druge generacije može doneti prednosti u pogledu brzine
- Programi napisani na ovim jezicima zauzimaju malo memorije

Mane:

- Programiranje je izraženo u vidu pojedinačnih instrukcija procesora, a ne logike višeg nivoa.
- Memorija niskog nivoa i detalji o hardveru moraju se ručno upravljati, što je često sklono greškama.
- Programi zavise od mašine, tako da se za svaku ciljnu arhitekturu mašine moraju napisati različite verzije.

4. Treća generacija programskih jezika

Programski jezici treće generacije (3GLs) se odlikuju visokim nivoom instrukcija - jedna naredba ovih programskih jezika prevodi se u više naredbi mašinskoj jezika. Ovi programski jezici su nezavisni od računara i bolje prilagođeni programeru u odnosu na programske jezike prve i druge generacije, ali imaju manje specifičan fokus u odnosu na programske jezike četvrte i pete generacije.

4.1. O programskim jezicima III generacije

Za rad u ovim programskim jezicima nije potrebno poznavati arhitekturu, instrukcije i registre računara za koji se programira, pa su programi prenosivi, tj. jedan program je radio na više računara. Ovo je napredak u odnosu na drugu generaciju, odnosno asemblerske jezike.

Kod se piše uz pomoć engleskih reči koje označavaju: deklaraciju promenljivih, konstanti, blokove, petlje, uslovna grananja. Time se znatno olakšava učenje i pisanje koda.

Jedna od osnovnih karakteristika ovih programa je akcenat postavljen na strukturi, te se često klasificiraju kao strukturni jezici - svi događaji se dešavaju po predviđenoj strukturi programskog koda.

Program napisan u programskom jeziku treće generacije se zove Source code ili Source program i on se prevodi na mašinski jezik pomoću kompajlera (eng. Compiler).

4.2. Programski jezik FORTRAN

Ovaj programski jezik se prvi put pojavio 1957. godine, napravila ga je kompanija IBM i njegovo ime je skraćenica od Formula Translation.

Potisnuo je asemblerske jezike i prihvatili su ga naučnici i vojska. Prvobitno je korišten za proračun parametara u nuklearnoj elektrani, a posle za vojne projekte i svemirska istraživanja.

Aktuelna verzija FORTRAN-a je iz 2003, on se ne razlikuje puno od prvobitne verzije osim dodatka nekih novih naredbi i uklonjenih nepravilnosti.

Uticao je na jezike koji će kasnije nastati: C, Basic,...

4.3. Programski jezik C

C programski jezik spada u proceduralne programske jezike, razvijen je u ranim 70-im godinama 20. veka. Zaslužan za stvaranje ovog programskog jezika je Denis Riči, ali takođe treba pomenuti i naučnike Kena Tompsona i Martina Ričardsa. Nastao je u istraživačkom centru Belove laboratorije u Nju Džerziju za potrebe operativnog sistema UNIX.

U ovom programskom jeziku su programirane neke verzije operativnih sistema Windows i UNIX. Danas se C koristi u velikoj meri, prvenstveno za sistemsko programiranje.

Operativni sistem Android je delom programiran u programskom jeziku C.

Uticao je na programske jezike: C++, Java, C#, PHP, JavaScript,...

4.4. Programski jezik C++

Ovaj programski jezik je nastao kao proširenje C-a, u vidu objektno orijentisanog programiranja tokom 1980-ih godina. Originalo ime mu je bilo “C sa klasama” (eng. C with classes). U 1983. godini taj naziv je promenjen u C++. Zvanično je objavljen 1985. godine.

C++ se koristi za pisanje i programiranje: aplikacija, editora za rad sa videom i grafikom, operativnih sistema, video igri,...

```cpp
#include <iostream>
using namespace std;
int main()
{
    cout << "How to compile your C++ code in Visual Studio Code\n";
    cout << "https://bolajiayodeji.com\n";
    cout << "@iambolajiayo\n";
    return 0;
}
```

Slika 4.1. Program napisan u programskom jeziku C++

5. Četvrta generacija programskih jezika

Četvrta generacija programskog jezika (4GJ) je računarski programski jezik zamišljen kao dorada stila jezika klaisfikovanih kao treća generacija programskog jezika. Iako se definicija četvrta generacija programskog jezika promenila tokom vremena, može da se odlikuje kao operative sa više velikih podataka odjednom umesto da se fokusira samo na bitove i bajtove. Jezici četvrte generacije često se porede sa domen specifičnim jezicima (DSJ). Neki tvrde da je 4GJ podskup DSJ.

5.1. O programskim jezicima IV generacije

Koncept 4GJ je razvijen od 1970-ih godina kroz 1990-e, preklapajući se sa većinom razvoja 3GJ. Dok su jezici 3GJ kao C, C++, C#, Java, i Javaskripti i dalje popularni za širok spektar upotrebe, jezici 4GJ kao što je prvobitno definisano su našli užu primenu. Neki napredne jezici 3GJ kao Pajton, Rubi, i Perl kombinuju neke 4GJ sposobnosti u okviru opšte namene u okruženju 3GJ. Takođe, biblioteke sa karakteristikama 4GJ su razvijene kao dodaci za najpopularnije jezike 3GJ. Ovo je zamaglilo razliku između 4GJ i 3GJ.

Rana ulazna šema za 4GJ podržavala je unos podataka u okviru limitacije od 72-simbola buhene kartice(8 bajtova se koristi za sekvenciranje) gde će oznaka jedne kartice da identifikuje vrstu ili funkciju. Sa razumnom upotrebom nekoliko karata, 4GJ špil može da ponudi širok spektar prerade i sposobnosti izveštavanja dok je ekvivalentna funkcionalnost kodirana u jezicima 3GJ mogla podvesti, možda, celu kutiju ili više kartica. Metafora 72-simbola je nastavila još neko vreme dok je hardver napredovao do veće memorije i terminalnog interfejsa. Čak i sa svojim ograničenjima, ovaj pristup je podržao visoko sofisticirane aplikacije.

5.2. Podela

1. Tablom-vođeno programiranje, umesto korišćenja koda, programer definiše svoju logiku izborom operaciju u unapred utvrđenoj listi memorijskih ili tabelarnim podacima manipulisanim komandama. Drugim rečima, umesto kodiranja, programer koristi tabelom-vođene algoritme programiranja. Dobar primer ove vrste jezika 4GJ je PoverBilder. Ove vrste alata mogu da se koristi za poslovni razvoj aplikacija i obično se sastoji u paketu koji omogućava i poslovnu manipulaciju podataka i izveštavanja, zbog toga oni dolaze sa GUI ekranima i editorima prijava.

2. Izveštaj-generatorski programski jezici uzimaju opis formata podataka i izveštaj koji generišu i od toga oni ili generišu potreban izveštaj direktno ili generišu program za generisanje izveštaja.

3. Obični generatori upravljaju online interakcijama sa korisnicima aplikacionog sistema, ili prave programe za to.

4. Više ambicioni jezici 4GJ pokušavaju da automatski generišu čitave sisteme iz izlaza kejs alata, specifikacije ekrana i izveštaja, a možda i specifikaciju neke dodatne logike procesuiranja.

5. Upravljanje podacima jezika 4GL kao što su SRS, SPSS i Stata obezbeđuju sofisticirane komande kodiranja za manipulaciju podataka, preoblikovanje fajlova, izbor predmeta i dokumentaciju podataka u pripremi za statističku analizu i izveštavanje.

5.3. Programski jezik Perl

Perl je slobodni, nezavisni od platforme i interpretirani programski jezik kojeg je razvio Amerikanac Leri Vol 1987. godine. Nastao je kao sinteza programskog jezika C, nekih komandi operativnog sistema juniks i drugih elemenata. Skraćenica PERL potiče od naziva Practical Extraction and Report Language koji u potpunosti onjašnjava najjače osobine perl-a, praktičnost, izdavanje i analiza datoteka i podataka,generisanje izlaznih podataka, i programski jezik.

```
#!/usr/bin/env perl

use 5.018;
use strict;
use warnings;
use autodie;

use List::MoreUtils qw[uniq];

sub format_result { join "\n", map {'REGION: '.join

sub merge_regions_and_update_caches
 {
  my ($regions, $regions_of_point, $current_line, $
  ${ $regions } = [
   [ map { @{ $_ } } @{ $regions_of_point } ],
   grep {my $r = $_; (scalar grep { $r == $_ } @{ $
    @{ ${ $regions } }];
  for my $line ($current_line, $previous_line)
   {
    for my $item ( @{ ${ $line } })
     {
```

Slika 5.1. Program ispisan u programskom jeziku Perl

5.4. Programski jezik Python

Python je programski jezik visokog nivoa opšte namene. Podržava, u prvom redu imperativni, objektno-orijentisan i funkcionalni stil programiranja. Sintaksa jezika Pajton omogućava pisanje veoma preglednih programa. Programi pisani u Pajton jeziku se najčešće interpretiraju. Uz interpretator se obično isporučuje i veoma razvijena standardna biblioteka modula.

```
134    class GameObject:
135
136        def __init__(self, image_path, x, y, width, height):
137            # Player image import and resize
138            object_image = pygame.image.load(image_path)
139            self.image = pygame.transform.scale(object_image, (width, height))
140
141            self.x_pos = x
142            self.y_pos = y
143
144            self.width = width
145            self.height = height
146
147        # Draw the object by blitting it onto the background(game_screen)
148        def draw(self, background):
149            background.blit(self.image, (self.x_pos, self.y_pos))
150
```

Slika 5.2. Program ispisan u programskom jeziku Python

Grafik 5.1. Rast popularnosti programskog jezika Python

5.5. Programski jezik PHP

PHP specijalizovani je skriptni jezik prvenstveno namenjen za izradu dinamičnog veb sadržaja i izvodi se na strani servera. Stekao je svoju popularnost zbog svoje jednostavnosti i sintakse nasleđene iz programskog jezika C. Tokom vremena jezik se proširivao i sticao mogućnosti za objektno orijentisano programiranje, naročito od verzije 5.0.

```php
<?php

// Random PHP code snippet!

function create_category_feeds($categories = NULL) {

    global $wpdb, $title, $headcomments;

    if ($categories == NULL) {
        $sort_column = 'term_id';
        $query = "SELECT * FROM $wpdb->term_taxonomy
                  JOIN $wpdb->terms ON ( $wpdb->term_taxonomy.term_id = $wpdb->terms.term_id )
                  WHERE $wpdb->term_taxonomy.taxonomy = 'category' AND $wpdb->terms.term_id > 0 AND count
                  ORDER BY $wpdb->terms.name ASC";
        $categories = $wpdb->get_results($query);
    }

    $catsnum = count($categories);

    foreach ($categories as $category) {
        $link = '<link rel="alternate" type="application/rss+xml" title="';
        $link = $link . $title . ': ' . $category->name;
        $link = $link . '" href="' . get_category_rss_link(0, $category->term_id, $category->name) . '" /
        echo "\t" . $link . "\n";
    }

    $hcomlink = '<link rel="alternate" type="application/rss+xml" title="';
    $hcomlink = $hcomlink . $title . ': Comments';
```

Slika 5.3. Program ispisan u programskom jeziku PHP

5.6. Programski jezik Ruby

Ruby je objektno orijentisani programski jezik. U sebi kombinuje sintaksu inspirisanu jezicima Perl i Ada, sa objektno orijentisanim osobinama nalik jeziku Smalltalk (Smalltalk), a deli i neke osobine sa jezicima Pajton, Lisp, Dylan i CLU. Rubi je jednoprolazni interpretirani jezik. Njegova glavna implementacija je slobodni softver pod licencom otvorenog koda.

```ruby
require "time"

class InvoiceItem
  attr_reader :id, :item_id, :invoice_id, :created_at

  attr_accessor :quantity, :unit_price, :updated_at

  def initialize(params)
    @id = params[:id].to_i
    @item_id = params[:item_id].to_i
    @invoice_id = params[:invoice_id].to_i
    @quantity = params[:quantity].to_i
    @unit_price = BigDecimal(params[:unit_price])
    @created_at = Time.parse(params[:created_at].to_s)
    @updated_at = Time.parse(params[:updated_at].to_s)
  end

  def unit_price_to_dollars
    @unit_price.to_f
  end
end
```

Slika 5.4. Program ispisan u programskom jeziku Ruby

6. Peta generacija programskih jezika

Programski jezik pete generacije (skraćeno kao 5GJ) je programski jezik baziran na rešavanju problema korišćenjem ograničenja data programu, umesto korišćenja algoritma koje je napisao programer. Većina Ograničeno-baziranih i programsko logičkih jezika i neki deklarativni jezici su peta generacija jezika. Predstavljen je 90-tih godina.

Dok su programski jezici četvrte generacije dizajnirani za izradu posebnih programa, jezici pete generacije su osmišljeni tako da računar reši dati problem bez programera. Na ovaj način, korisnik samo treba da brine o tome koje probleme treba rešiti i koje uslove treba da ispuni, bez brige o tome kako će primeniti rutinu ili algoritam za njihovo rešavanje. Jezici pete generacije uglavnom se koriste u istraživanjima veštačke inteligencije.

Peta generacija programskih jezika još uvijek se nalazi u pionirskom dobu.

Najpoznatiji programski jezici pete generacije su: LISP i PROLOG,OPS5,MERCURY

6.1. Prednosti

- Jezici pete generacije omogućavaju korisnicima da komuniciraju sa računarskim sistemom na jednostavan i lak način.
- Programeri mogu koristiti normalne engleske reči tokom interakcije sa računarskim sistemom.
- Jezici pete generacije mogu se koristiti za brzo i efikasno prenošenje baze podataka.

6.2. Programski jezik Lisp

```
0.  METEOR((
1.  (*   (ROSE) (FLOWER)  * (SIMPLE REPLACEMENT))
2.  (*    ((*P THE WORKSPACE IS))         *   (DEBUG PRINTOUT))
3.  (*  (IS A ROSE)    (0     *  (DELETION))
4.  (*   (A FLOWER IS)  (3 1 2)           *  (REARRANGEMENT))
5.  (*  ((*P WS2))              *)
6.  (*  ( FLOWER)  ( 1 OF RED)       *     (INSERTION) )
7.  (*   (A FLOWER)  (THE 2)   * (REPLACEMENT IN CONTEXT))
8.  (*  ((*P WS3))             *)
9.  (*   (FLOWER)                        *  (NO OPERATION))
10. (*     (RED)   (1 1)         * (DUPLICATION))
11. (*  ((*P WS4))             *)
12. (*   (OF ($.1))   (1)     *(SINGLE UNKNOWN CONSTITUENT))
13. (*    (($.1))    (QUESTION 1)             *  (FIRST CONSTITUENT))
14. (*    ( (*P WS5))         *)
15. (*   ( ($.2)  FLOWER ($.3))   (3 2 1) * (N CONSECUTIVE CONSTITUENTS
16. (*  ((*P WS6))             *)
17. (*  (FLOWER $ ROSE)  (1 3)    *   (UNKNOWN NUM OF CONSTITUENTS))
18. (*    ( (*P WS7))        *)
19. (*    ($)   (START C A B D)  * (REPLACING ENTIRE WORKSPACE))
20. (*  (START ($.1)  $ D)   (1 3 2 4)  *)
21. (*  ((*P WS8))
22. (*   ($)          END)
23. )(A ROSE IS A ROSE IS A ROSE))
```

Slika 6.1. Program ispisan u programskom jeziku Lisp

Lisp je programski jezik koji je projektovao Džon Makarti krajem 1950-tih i jedan je od najstarijih programskih jezika. Iako je Lisp opštenamjenski programski jezik, obično se kaže da je jezik veštačke inteligencije, odnosno oblasti u kojoj se najviše koristi.

6.3. Programski jezik Prolog

Prolog (PROgramming in LOGic) je deklarativan programski jezik namenjen rešavanju zadataka simboličke prirode. Prolog se temelji na teorijskom modelu logike prvog reda. Početkom 1970-ih godina Alain Kolmerauer i Filipe Rousel na Univerzitetu u Marselju, zajedno sa Robertom Kovalskim sa Odeljka Veštačke Inteligencije na Univerzitetu u Edinburgu razvili su osnovni dizajn jezika Prolog.

```
Prolog in Elixir
?- assert(append([], Xs, Xs)).
true
?- assert((append([X | Ls], Ys, [X | Zs]) :- append(Ls, Ys, Zs))).
true
?- append([1,2],[a,b],X).
X = [1,2,a,b]
true
?- append(X,Y,[1,2,3]).
X = []
```

Prolog ima avaznu ulogu u oblasti veštačke inteligencije. Lako moze otkriti da li neke izjave slede logiku ili ne. Programi napisani u Prologu često su manji, lakše razumljivi i lako održivi.

6.4. Programski jezik Mercury

Mercury je logički programski jezik pete generacije.Korsiti se za pravljenje velikih programa, brz je i efikasan. Sintaksa Prologa i Mercurya je ista ali oba jezika su različita zato sto se Mercuryev sistem modula,tip i način rada razlikuju.

6.5. Programski jezik OPS5

OPS5 (official production system) je programski jezik proizvodnog sistema zasnovanim na pravilima. Pravilo se sastoji od preduslova i rezultirajuće radnje. Sistem proverava u memoriji pravila čija je preduslov proveren. Ako se verifikuje, izvršava se akcija zadovoljenog pravila. Koristi se u veštačkoj inteligenciji,eksperimentalnim sistemima.

```
(P Rule-Example
     (patient-data  ^AGE $A
                    ^HEIGHT ($B < $A)
                    ^NAME $N)
     (monitor-data  ^NAME $N
                    ^PULSE ($C > $B))
     -->
     (Write "Patient "$N" is in critical
     condition")
     )
```

7. Zaključna razmatranja

Svaka generacija programskih jezika je doprinela razvoju računarstva i informatike, kao i unapredila neke stavke prosle generacije:

- Prva generacija programskih jezika je korištena za programiranje na mašinskom nivou
- Druga generacija programskih jezika je olakšala pisanje programa u odnosu na prvu time što je 0 i 1 zamenila sa komandama. Za prevođenje u mašinski kod se koristi Asembler
- Treća generacija programskih jezika je komande druge generacije svela na ljudski jezik. Takođe programi su postali prenosivi i nezavisni od mašine ili porodice procesora. Za prevođenje u mašinski kod se koristi Compiler
- Četvrta generacija programskih jezika je proširenje koncepta treće generacije, jezici su usmereni i ne rade vise sa bitovima i bajtovima
- Peta generacija programskih jezika je osmišljena tako da računar, uz date podatke i ograničenja, reši problem bez ljudsko napisanog algoritma

8. Literatura i reference

https://en.wikipedia.org/wiki/First-generation_programming_language

https://en.wikipedia.org/wiki/Second-generation_programming_language

“Ratvoj programskih jezika” - David Aksović i Nikola Damjanović, 2018.

“III ГЕНЕРАЦИЈА ПРОГРАМСКИХ ЈЕЗИКА” - Martin Gobeljić, 2021.

“IV GENERACIJA PROGRAMSKIH JEZIKA” - Branka Arsić, 2021.

“Пета генерација програмских језика” – Aleksa Minašević, 2021.

Datum predaje:

Komentar:

Ocena: