DAŽĀDU PROGRAMMĒŠANAS VALODU ATTĪSTĪBA
EVOLUTION OF DIFFERENT PROGRAMMING LANGUAGES

Author: **Edgars CIMERMANIS**, e-mail: yoummolv@gmail.com
Scientific supervisor: **Dr.sc.ing., profesors Pēteris GRABUSTS,** e-mail: Peteris.Grabusts@rta.lv
Rēzeknes Tehnoloģiju Akadēmija
Atbrīvošanas aleja 115, Rēzekne, Latvija

Abstract. *This article is about programming languages, that has improved over time and is still evolving. The goal of this work is to delve into specifics of programming languages and the logic behind they're evolving. This paper will overlook parts of programming languages to understand, why there is so many of them and how they work.*

Keywords: *Machine code, Assambler, Fortran, COBOL, C++, C#, Java.*

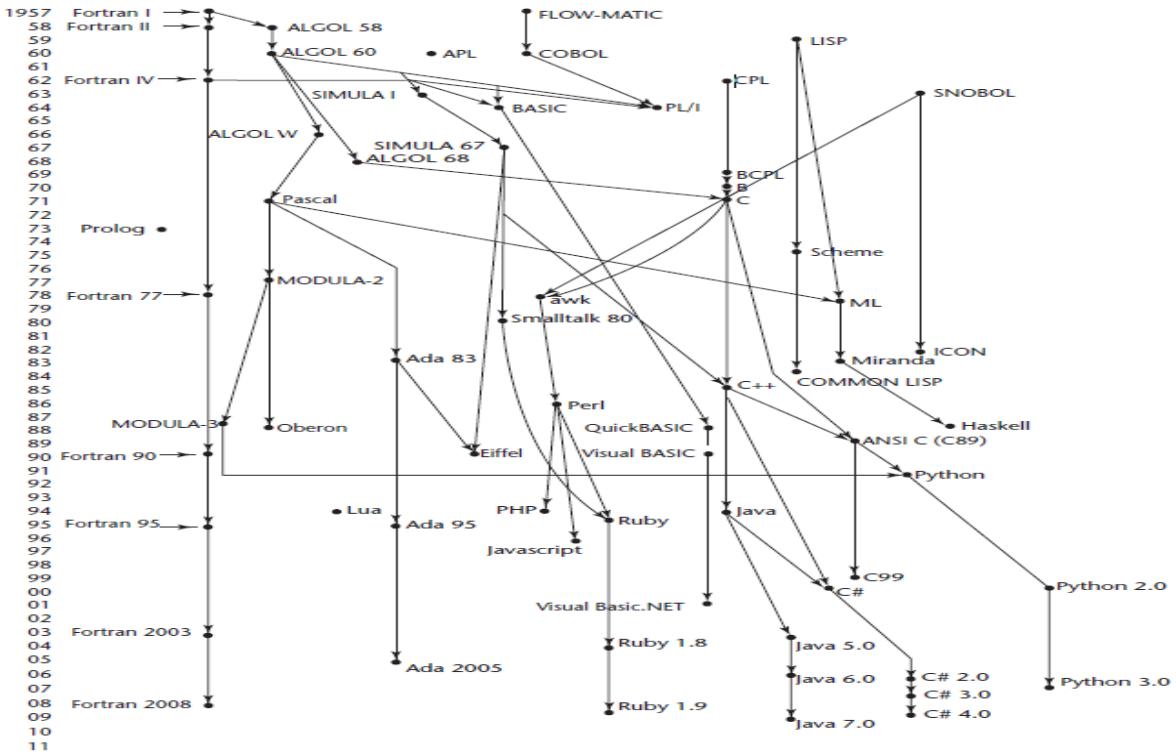
Ievads

Programmēšanas valodu attīstība ir ļoti cieši saistīta ar datoros izmantoto fizisko un elektronisko procesu attīstību.

Par pirmās datoriem līdzīgās mašīnas autoru tiek uzskatīts Čārlzs Bebidžs.

1940. gados tika radīti pirmie elektriskie datori. Sākotnēji, datoru izstrādi virzīja nepieciešamība veikt sarežģītus aprēķinus, kā šifrēšana, atšifrēšana un trajektoriju aprēķināšana militārā sfērā. Tajā laikā, datori bija ļoti lieli, lēni un ļoti dārgi.

Pirmā ļoti plaši lietotā programmēšanas valoda bija Fortran, kuru izstrādāja ***IBM*** no 1954.- 1957. gadam.



1. attēls. Programmēšanas valodu evolūcijas shēma

https://doi.org/10.17770/het2020.24.6746

Programmēšanas valodu izcelšanās

Sākotnēji visi algoritmi tika rakstīti uzreiz mašīnkodā (machine code). Tāda pieeja padarīja sarežģītu esošo algoritmu izstrādes procesu. Mašīnkodā rakstiskas instrukcijas skaitliskā veidā rādīja gan komandas, gan vērtības un atmiņas adreses.

Pirmais solis darba atvieglošanai bija atteikšanās no komandu un to operandu pieraksta tieši tajā formā, kāda tiek izmantota mašīnā. Tādējādi, skaitliskā komandu pieraksta vietā plaši sāka pielietot mnemonisko (vārdisko) pierakstu. Arī atmiņas apgabalus noteiktā veidā nosauca ar vārdiem (identifikatoriem).

Sākotnēji programmētāji izmantoja šādu pierakstu tikai pierakstā uz papīra, taču vēlāk kļuva skaidrs, ka mnemoniskā programmas teksta pārveidošanu mašīnkodā var veikt pats dators. Šim nolūkam tika izstrādātas programmas, kuras tika sauktas par asambleriem (assembler).

Asamblera valodā uzrakstīta programma ir ļoti cieši piesaistīta mašīnai, kurai tā ir rakstīta, tāpēc universālu programmu rakstīšanai bija nepieciešamas augstāka līmeņa – platformu neatkarīgas konstrukcijas. To piedāvāja trešās paaudzes programmēšanas valodas, no kurām tipiskākie pirmie piemēri ir FORTRAN (FORmula TRANslator) un COBOL (COmmon Business-Oriented Language).

Fortran

Fortran ir pirmā plaši pieņemtā augsta līmeņa programmēšanas valoda, kuru bija iespējams kompilēt. Pirmā versija Fortran I deva iespēju programmētājam veidot mainīgos ar nosaukuma garumu līdz 6 simboliem. Mainīgie kuru sākās ar burtiem I, J, K, L, M un N tika izmantoti kā veselu skaitļu mainīgie, bet visi pārējie kā mainīgie priekš skaitļiem ar peldošo punktu.

Fortran II tika izlabots lielākais Fortran I trūkums, un tas bija koda garums kuru varēja apstrādāt kompilators. Ja koda garums pārsniedza 300 rindas, bija liela iespēja ka kompilācijas procesā var rasties mašinkļūda un kompilācija netiks pabeigta.

Fortran III tika izveidota, bet netika plaši izplatīta, Toties Fortran IV kļuva par vienu no visplašāk pielietotajām programmēšanas valodām tā laika. Lielāka atšķirība no Fortran II bija iespēja deklarēt mainīgos ar noteiktiem datu tipu veidiem, izveidota "if" nosacījuma struktūra un iespēja nodot funkcijas parametrus citai funkcijas.

Fortran IV tika aizvietots ar Fortran 77 kas saglabāja visas iespējas piedāvātās iepriekš, bet arī papildināja ar jaunam iespējām, kā piemēram iespēju strādāt ar teksta rindas mainīgajiem "string". Jaunās funkcijas iekļāva iespēju izveidot loģiskā cikla kontroles nosacījumus un papildinot "if" nosacījumu struktūru izveidoja "else" daļu kas ļāva veidot sarežģītākus nosacījumus.

Fortran 90 atšķirība no Fortran 77 bija ļoti dramatiska. Lielākas izmaiņas bija iespējā veidot dinamiskos masīvus, klases, rādītāju ieviešana, iespēja izveidot nosacījumus ar vairākām izvēlē un izveidotās funkcijas varēja izveidot rekursīvas.

Nākošais lielākais uzlabojums Fortran programmēšanas ģimenē nāca ar Fortran 2003, kur tika pievienota objektu-orientētās programmēšanas iespējas.

COBOL

Kaut arī tā ir viena no visizmantotākajām programmēšanas valodām, tai ir ļoti maza ietekme uz valodām kas sekoja pēc tās. Vienīgais izņēmums ir PL/I valoda. Visdrīzāk lielākais iemesls kāpēc COBOL valodai ir tik maza ietekme uz programmēšanas valodu attīstību ir tas, ka tā tika veidota biznesa aplikācijām. Cits iemesls var būt tas ka liela daļa izaugsmes biznesa datornozarē ir notikusi mazos uzņēmumos. Tikai ļoti maza daļa programmēšanas ir notikusi dotajos uzņēmumos, jo ierasti tika pirktas lietotņu paķetes tieši no ražotāja vai veikala un izmantotas pildot noteiktos uzdevumus.

Pirmā tikšanās par COBOL veidošanu tika sponsorēta pateicoties ASV, Aizsardzības departamentam. Tikšanās laikā nolēma kādām jābūt galvenajām programmas īpašībā, kā piemēram,

- Izmantot angļu valodu pēc iespējas vairāk.
- Valodai jābūt viegli pielietojamai, pat ja tas izmaksās programmēšanas valodas veiktspēju, kā iemesls tam bija palielināt lietotāju daudzumu

Kaut arī valodas veidošanas laikā radās daudz strīdu par tās iespējām.

Tomēr 1960. gadā tika publicēts raksts par COBBOL valodu, kuru nosauca par COBOL 60. Viens no jaunievedumiem ko ieviesa COBOL 60 bija hierarhiska datu struktūra (records), kas pirmo reizi parādījās Plankalkül, un tika implementēts valodā COBOL. COBOL stiprā puse bija darbs ar datiem, bet vājā puse bija funkcijas, jo COBOL to bija ļoti maz.

PL/I

Galvenā koncepcija programmēšanas valodas PL/I bija "Viss priekš visiem". 1960. gadu sākumā datoru lietotāji iedalījās 2 lielās grupās, kurām pat nebija vajadzības sadarboties lai veiktu sav nepieciešamos uzdevumus. Tās 2 grupas bija lietotāji kuri nodarbojās ar zinātniskiem aprēķiniem un lietotāji kas nodarbojās ar biznesa saistītiem aprēķiniem. Zinātnieki izmantoja Fortran valodu, tāpēc ka tā piedāvāja iespēju strādāt ar peldošā-punkta datiem un masīviem. Tomēr biznesa sfēras lietotāji izmantoja COBOL valodu, tāpēc ka tā piedāvāja iespēju izmantot decimālu un rakstzīmju simbolu rindas, kā arī efektīvu datu ievadu un izvadu.

Tomēr attīstoties valodu pielietojumam, radās pieprasījums pēc funkcijām kas nebija piedāvātas atbilstošajās valodās. Piemēram zinātniekiem parādījās vajadzība ievadīt lielu daudzumu datu, tāpēc vajadzēja efektīvu veidu kā ievadīt un izvadīt datus. Vienlaikus, biznesa sfēras lietotājiem, veidojot sarežģītas informācijas sistēmas, parādījās vajadzība pēc peldošā-punkta datu un masīvu pielietojums. Turpinoties šīm tendencēm, drīz rastos problēma. IBM nolēma izveidot jaunu programmēšanas valodu kas sevī apkopotu gan Fortran, COBOL, LISP un asamblera iespējas.

Vieglākais veids kā paskaidrot PL/I valodu ir vienkārši nosaukt tās valodu iespējas kuras tā mantoja no saviem priekštečiem. Kā piemēram no valodas ALGOL 60 tā mantoja rekursiju un bloku struktūru, no valodas Fortran IV – kompilēšana notiek atsevišķi izmantojot globālus datus, COBOL 60 – datu struktūras, datu ievade/izvade un ziņojumu ģenerēšanas iespēja un liels daudzums jaunu konstrukciju, kuras tika apvienotas veidojot vienu valodu.

Kaut arī PL/I valoda vairs nav populāra, v ar nosaukt lietas kurās tā bija pirmā:

- Programmām tika atļauts izveidot vienlaicīgi izpildāmās apakšprogrammas. Lai gan tā bija laba ideja, valodā PL/I tā tika vāji attīstīta.
- Bija spējīga atrast un apstrādāt 23 dažādu veidu izņēmumus vai kļūdas.
- Apakšprogrammas varēja izmantot rekursiju, bet to iespēju varēja atspējot, atļaujot daudz efektīvāku savienojamību priekš apakšprogrammām bez rekursijas.
- Rādītāji tika pievienoti, kā jauns datu veids
- Vairāku dimensiju masīvos bija iespēja atsaukties uz noteiktu rindu/kolonnu kā pret jaunu viendimensijas masīvu.

C++

Programmēšanas valoda C++ evolucionēja no programmēšanas valodas C, kas tomēr bija ALGOL 68 pēctecis. Kaut arī valoda C neieviesa lielas izmaiņas programmēšanas valodu attīstība, tā tika plaši izplatīta lielā daudzumā sfēru. C++ pārņēma valodas C iespējas, iedeva iespēju atbalstīt Smalltalk valodas jaunievedumu, objektorientēto programmēšanu. Tas notika 1980. gadā un jaunā programmēšanas valoda vēl joprojām saucās par C valodu. Jaunas iespējas

kuras parādījās uz C valodā bija publiskā/privātā piekļuve pie objektiem, konstuktoru un destruktoru metode, inline funkcijas, noklusējuma parametri.

1984. gadā šo valodu paplašināja, iekļaujot virtuālās metodes, kas nodrošinot dinamisku metožu izsaukumu saistīšanu ar specifiskām metožu definīcijām, metodes nosaukumu un operatora pārslodzi (overloading), kā arī atsauces veidiem. Šī valodas versija tika saukta par C++.

No 1985. gada līdz 1989. gada turpinājās C++ valodas attīstība, galvenokārt balstoties uz lietotāju atsauksmēm. Jaunajā versija ieviesa vairākkārtīgu mantošanu, tas ir, vienai klasei var būt vairākas klases. 2002. gadā, kad Microsoft prezentēja savu .NET platformu, C++ tika pievienota iespēja izmantot .NET funkcionalitāti.

C++ ātri kļuva plaši izplatīta programmēšanas valoda, kuru izmanto intensīvi pat šodien. Viens no iemesliem ir C++ valodas savienojamība ar C valodu. Kas ļauj C++ valodā veidot programmas priekš C valodas un izpildīt tās kā C++ programmas. Vēl viens iemesls tam ir tas ka liela daļa programmētāju bija apguvusi C valodu un C++ valodai attīstoties no C valodas, jaunās valodas apguve bija daudz vienkāršāka. Bet lielākais iemesls ir objektorientētās programmēšanas atbalsts, jo 200. gadu sākumā objektorientētā programmēšana ieguva lielu popularitāti, bet C++ bija vienīgā valoda uz to momentu, kuru komerciāli izmantot lielos projektos.

No negatīvā viedokļa, tāpēc ka C++ valoda bija tik liela un sarežģīta. Vēl problēmu radīja ka C++ mantoja visas drošības problēmas no valodas C. Kas padarīja C++ vājāku par alternatīvām kā Ada un Java.

Java

Java izstrāde tika uzsākta izmantojot C++ valodu. Tika noņemtas dažas konstrukcijas, izmainīja dažas eksistējošas un pievienoja jaunas. Tas padarīja jauno valodu mazāku, saprotamāku un drošāku.

Kā arī daudzām programmēšanas valodām, tās mērķis bija zināms pirms tās veidošanas, un tas bija izveidot valodu ko varētu izmantot sadzīves tehnikā kā, tosteri, mikroviļņu krāsnis vai arī interaktīvie televizori. Stabilitāte bija viens no pašiem svarīgākajiem priekšnosacījumiem programmēšanas valodas veidā. Kaut daudzi uzskatītu stabilitāti ne par pirmo punktu programmēšanas valodas izveidē. Pat ja tostera programmā rastos nestabila kļūda, tas neizraisītu briesmas vai legālas problēmas, ja lielu kļūdu atrastu pēc tam kad tika tiktu pārdotas miljons ierīču? Izmaksas kas rastos ja vajadzētu atsaukt visas ierīces būtu milzīgas. Kaut arī Java tika veidota priekš sadzīves ierīcēm, sākumā ierīces pat netika pārdotas.

Tikai 1993. gadā kad internets kļuva plaši izmantojams pateicoties grafiskām pārlūkprogrammām, tika atklāts ka Java valoda ir piemērota interneta lapu programmēšanai. It īpaši Java sīklietotnes, kuras ir relatīvi mazas programmas kuras interpretēja interneta pārlūkprogrammā.

Viena liela atšķirība start C++ un Java kas tika balstīta uz C++ ir tas, ka Java nav iespējams veidot patstāvīgas programmas. C++ atbalsta gan proceduriālu, gan objektorientētu programmēšanu, Java tikai atbalsta objektorientēto programmēšanu. Vēl viena atšķirība ir tas ka Java atbalsta mantošanu tikai no vienas klases, bet C++ var mantot no vairākām klasēm.

Lielākais uzlabojums programmēšanas pasaule ir programmēšanas valodu drošības uzlabošana un sarežģītās pārlieku liesās C++ valodas uzlabošana izmetot dažas iespējas. Protams tas noder vairāk tikai interneta lietotnēm, bet mūsdienu pasaulē, liela daļa visu aplikāciju ir interneta balstītas.

C#

C# ir balstīts uz C++ un Java valodām, ņemot dažas idejas no Delphi un Visual BASIC valodām. C# mērķis bija būt par valodu komponenšu balstītai programmatūras veidošanai.

Galvenokārt priekš projektiem balstītiem uz .NET struktūras. Dotajā vidē ir iespējams kombinēt komponentes no dažādām programmēšanas valodām veidojot vienu vienotu sistēmu. Visas .NET valodas kuras iekļauj C#, Visual BASIC.NET, Managed C++ (C++ ar .NET atbalstu), F# un JScript.NET izmanto kopēja tipa sistēmu (Common Type System). CTS nodrošina visas .NET valodas ar kopīgu klašu bibliotēku. Visas valodas manto informāciju no viena CTS.

Kaut arī uzskatīts ka Java valodas priekšrocība pret C++ bija tajā, ka Java atmeta daudz sarežģītu, un nedrošu elementu, tomēr C# dizaineri tam nepiekrita un c# mantoja visas C++ iespējas. Drošības problēmas tika izlabotas. Struct struktūra tika uzlabota un kļuva noderīga, bet C++ tai nebija nekādas nozīmes drošības un funkcionalitātes iemeslu dēļ. Vēl viena drošības problēma kura tika izlabota C# valodā bija drošības trūkums kad tika izmantoti rādītāji uz funkcijām un mainīgajiem.

C# bija domāts kā uzlabojums gan C++, gan Java valodām kā vispārējas nozīmes programmēšanas valoda. Kaut gan var apgalvot, ka dažas no tās iezīmēm ir solis atpakaļ, C# skaidri ietver dažas konstrukcijas, kas to virza tālāk no saviem priekšgājējiem. Dažas tā funkcijas, protams, pārņems arī citas programmēšanas valodas nākotnē.

Secinājumi

1. Programmēšanas valodas attīstās gan pēc to vajadzības, gan pielietojuma.
2. Laika gaitā uzlabojās valodu pieejamība, ātrdarbība, samazinājās nepieciešamais laiks darbā ar tām.
3. Programmēšanas valodas, laikam ejot, papildina un uzlabo viena otru, tādējādi programmētāja darbs tiek manāmi samazināts.

Summary

Programming languages evolution deppends on human needs and it's application. The availability and speed of languages has improved over time and the time required to work with them decreased. Programming languages complement and improve each other over time, thus significantly reducing the work of the programmer.

Literatūras saraksts

1. http://home.lu.lv/~janiszu/courses/eprg/eprg.all.pdf Pirmsākumi
2. http://fortranwiki.org/fortran/show/HomePage Fortran
3. http://www.csis.ul.ie/cobol/course/COBOLIntro.htm COBOL
4. https://en.wikipedia.org/wiki/PL/I PL/I
5. https://en.wikipedia.org/wiki/PL/I c++
6. https://en.wikipedia.org/wiki/Java_(programming_language) JAVA
7. https://www.c-sharpcorner.com/blogs/history-of-c-sharp-programming-language c#