Engleski jezik u kontekstu programiranja

Babić Peruško, Lidija

Undergraduate thesis / Završni rad

2018

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Pula / Sveučilište Jurja Dobrile u Puli**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:137:199766

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-03-28**

Image not found or type unknown

Repository / Repozitorij:

Digital Repository Juraj Dobrila University of Pula

zir.nsk.hr

Image not found or type unknown

Sveučilište Jurja Dobrile u Puli

Fakultet informatike u Puli

Lidija Babić

Engleski jezik u kontekstu programiranja

Završni rad

Pula, rujan 2018. godine

Sveučilište Jurja Dobrile u Puli

Fakultet informatike u Puli

Lidija Babić

Engleski jezik u kontekstu programiranja

Završni rad

JMBAG:0303015851, izvanredni student

Studijski smjer: Sveučilišni preddiplomski studij Informatika

Predmet: Engleski jezik

Znanstveno područje: društvene znanosti

Znanstveno polje: informacijske i komunikacijske znanosti

Mentor: izv. prof. dr. sc. Moira Kostić Bobanović

Pula, rujan 2018. godine

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisana Lidija Babić, ovime izjavljujem da je ovaj seminarski rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da nijedan dio seminarskog rada nije napisan na nedozvoljen način, odnosno da nije prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Studentica

Ime i prezime

Sadržaj

1. UVOD 1

2. PROGRAMSKI JEZICI 2

2.1. Razvoj programskih jezika 4

2.2. Usporedba programskog i prirodnog jezika 10

3. ENGLESKI JEZIK – SVJETSKI JEZIK 14

3.1. Povijest engleskog jezika 14

3.2. Engleski jezik i programiranje 17

3.3. Važnost engleskog jezika za učenje programskog jezika i programera 18

4. ENGLESKI JEZIK I C++ 22

4.1. Koncepti programiranja 23

4.2. Izrada programa u C++ 24

4.3. Osnovni koncept programskog jezika 27

5. ZAKLJUČAK 30

Slike 31

Literatura 31

1. UVOD

Programski jezik je jezik kojim se pišu računalni programi koje računalo može izvršiti, a računalo može izvršiti samo one operacije za koje su mu dodijeljene instrukcije odnosno program. U suštini, programski jezik sastavljen je od simbola i definiran preko pravila sintakse i semantike. Programski jezici služe za lakšu komunikaciju s računalom prilikom organiziranja i rukovanja informacijama. Već od samih početaka programskih jezika, počevši od Fortrana, a poglavito Basica, engleski jezik si je širom otvorio vrata u informatici i bez njega je više nezamislivo bilo što ozbiljnije raditi u području informatike.

Engleski jezik spada u skupinu germanskih jezika, a razvio se u kasnom srednjem vijeku s germanskim plemenima koja su naselila britansko otočje. Kako se širio britanski imperij, tako se širio i engleski jezik po britanskom kolonijalnom carstvu i na tim je područjima postao dominantan. Iz tih kolonija nastala je i zadnjih sto godina najveća svjetska sila – Sjedinjene Američke Države, koje su prihvatile engleski jezik kao materinski i nastavile samo širiti svjetsku dominaciju engleskog jezika.

Ovaj rad sastoji se od tri poglavlja nakon kojih slijede zaključak i sažetak.
U prvome poglavlju govorit ćemo o programskim jezicima i njegovu razvoju, usporediti prirodni i engleski jezik i govoriti o samom učenju i poznavanju programskih jezika.
U drugom ćemo govoriti o engleskom jeziku kao svjetskom jeziku, povijesti engleskog jezika i njegovoj važnosti u programiranju.
U trećem poglavlju govorit ćemo o vezi engleskog jezika i programskog jezika C++.

2. PROGRAMSKI JEZICI

Gledajući kroz povijest, ljudi su težili da sa što manje rada postižu veće učinke, koristili su ljudsku snagu, izvore prirodne energije, razvijale su se umjetnost i znanost. Širenjem ljudskih spoznaja razvijala se i potreba za sredstvima koja bi im pomogla u rješavanju svakodnevnih zadaća, problema i postizanja ciljeva sa što manje napora. Pojavom industrijske revolucije, izumom strojeva, povećava se i težnja za prenošenjem rada na strojeve. Izum računala je veliki napredak u tome planu.

Razvoj računala usporedo je pratio i razvoj programskih jezika. Programski jezik je skup detaljnih uputa za pisanje programa koje piše programer koje će računalo u konačnici izvršiti.

Programski jezici dijele se na dvije skupine:

Strojni jezik (engl. *machine languages*)

Jedini jezik koji računalo razumije i svaki drugi program pisan u nekom drugom jeziku potrebno je prevesti u strojni jezik. Prikazan je u binarnom obliku, s nizom kodova koji se označavaju s dvije znamenke, 0 i 1. Pisanje programa na strojnom jeziku je složeno, te iziskuje dobrog specijaliziranog stručnjaka koji poznaje samu građu računala i koristi brojčane kodove.

Simbolički jezici (engl. *assembly language*)

Jezik koji je jednu razinu iznad strojnog jezika. Koristi kratke mnemoničke[1] kodove i omogućuje programeru uvođenje naziva za blokove memorije koji sadrže podatke. Umjesto binarnog oblika koriste se simboli.

[1] Tehnika unaprijeđenog pamćenja.

Simbolički su jezici:

Jezici nižeg nivoa (engl. *low-level language*)

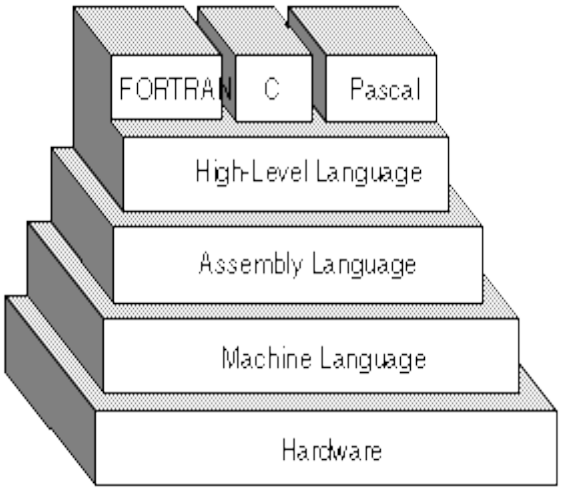
Osmišljeni su za rad i upravljanje hardverom te je njihova glavna funkcija upravljati i manipulirati hardverom. Programi i aplikacije napisani na niskoj razini jezika izravno se mogu izvršiti na računalnom hardveru bez ikakvog prijevoda i tumačenja. Bliži su strojnom jeziku iako je program pisan u njima zbog imena naredbi lakši za razumijevanje. Kod nižih programskih jezika instrukcije su opisane simbolički u odnosu na više programske jezike koji su bliži korisniku gdje su naredbe obično kratice engleskih riječi.

Jezici višeg nivoa (engl. *high-level language*)

Jezici u kojima programi nisu pisani u binarnom obliku te su bliži prirodnom jeziku, a ne jeziku stroja, samim time lakše su razumljivi i olakšavaju posao programeru jer nije potrebno poznavanje same građe računala i nisu ovisni o određenoj vrsti računala, te samim tim pisanje programa je omogućeno ljudima koje nemaju baš veliko znanje i nisu veliki stručnjaci[2].

[2] Dostupno na: https://www.britannica.com/technology/computer-programming-language/Introduction

Slika 1. Programski jezici



Izvor: http://www.webopedia.com/TERM/H/high_level_language.html

2.1. Razvoj programskih jezika

Danas postoji više tisuća programskih jezika i svake godine stvaraju se novi. Razvoj računala pratio je razvoj programskih jezika.

Prva generacija programskih jezika (rane 50-e)

Program prve generacije računala pisan je na strojnom jeziku. Svako računalo imalo je svoj strojni jezik usklađen s komponentama tog računala, te je pisanje programa bilo vrlo složeno i postojala je velika vjerojatnost pogreške što je zahtijevalo posebnog stručnjaka. Računala prve generacije mogla su izvoditi samo jedan program te je nakon njegova završetka računalo trebalo dovesti u početno stanje da bi s radom mogao početi drugi program.

Druga generacija programskih jezika (sredina 50-ih)

Za programiranje su se koristili asembleri gdje je naredba strojnog jezika zamijenjena simbolom odnosno skraćenicom gdje je skraćenica na primjer MUL – množenje i ADD zbrajanje. Prije izvođenja programa svaku naredbu bilo je potrebno prevesti na strojni jezik. Najbliži su strojnom jeziku i njegovim se naredbama izravno djeluje na određene dijelove računala.[3]

Treća generacija programskih jezika (oko 1960. godine)

Kod programskih jezika treće generacije naredbe su bile kratke riječi engleskog govornog područja koje su bliže ljudima i lako se pamte. U odnosu na prvu i drugu generaciju računala, treće generacije izvršavale su milijun operacija u sekundi. Došlo je do razvoja programa koji upravljaju i nadgledaju rad hardvera, memorije računala što je omogućavalo izvršavanje više programa u isto vrijeme (**engl. *multitasking***). Omogućena je obrada podataka u liniji (**engl. *on-line processing*)** gdje se ulazni podaci unose u računalo i odmah se dobije odgovor.

Neki od jezika treće generacije:

Fortran – 1957. godine (engl. *formula translation*) – bio je namijenjen za rješavanje numeričkih zadataka.

Cobol – 1960. godine (engl. *common business-oriented language*) – namijenjen za poslovne potrebe s glavnim karakteristikama masovne obrade podataka te kreiranje i održavanje velikih podataka.

[3] Dostupno na: tesla.carnet.hr
https://tesla.carnet.hr/pluginfile.php/21710/mod_resource/content/2/COURSE_2666625_M/my_files/Sadrzaj/Poglavlje%202/2_2_4%20Programski_jezik.htm?embed=1

Fortran i Cobol moguće je koristiti na različitim računalima.

Pascal – 1970. godine – dobio je ime po matematičaru i filozofu Blaiseu Pascalu te je bio namijenjen za rješavanje algoritma[4] (**engl. *algorithm***) i primjenu strukturiranog programiranja.

C – 1971. – Dennis Ritchie razvio je programski jezik C na temelju programskog jezika B u Bell Telephone Laboratories, Inc.

Tijekom 70-ih i 80-ih prošlog stoljeća jezik se brzo širio te ga je American National Standard Institute (ANSI)[5] standardizirao 1989. godine.

C je jezik niskog nivoa što znači da radi s brojevima i adresama s kojima rade i računala. Pomoću aritmetičko-logičkih operatora objekti se mogu kombinirati i premještati.

Osobine C jezika:

- nije ovisan o tipu računala, što je ostvareno uvođenjem tipova podataka
- konzistentnost i efikasnost
- pristup podacima pomoću adrese
- fleksibilnost
- mali jezik, 32 ključne riječi, prenosivost
- lak za učenje
- program se sastoji od obvezne *main*[6] funkcije s kojom započinje izvršavanje, te drugih.

C jezik je prethodnik C++ jezika te je ostao prilagođen svim sustavima – kako malim, tako i velikim.

[4] Opis za rješavanje nekog problema.
[5] Američki institut nacionalnih standarda je neprofitna organizacija, nadgleda razvoj standarda za servise, proizvode, procese u SAD-u.
[6] Obavezna funkcija, što označava glavni program.

C++ jezik – 1983. godine – razvio ga je Bjarne Stroustrup kao proširenje jezika C s nazivom C s klasama (engl. *C with classes*). Programskom jeziku C++ dodana je visoka razina objektno-orijentiranog programiranja te su ugrađene poboljšane biblioteke (engl. STL – Standard Template Library).[7]
Objektno-orijentirani jezici organizirani su oko podataka pri čemu podaci kontroliraju pristup kodu.

Osobine objektno-orijentiranih jezika

Enkapsulacija (**engl. *encapsulation***) – skrivanje podataka
Nasljeđivanje (**engl. *inheritance***) – preuzimanje svojstva drugog objekta
Polimorfizam (**engl. *polymorphism***) – više oblika

Java – 1995. godine – razvio ju je James Gosling, povijest razvoja pratio je i razvoj JDK (Java Development Kit) i JRE (Java Runtime Environment) izvršnog okruženja.

Osobine:

- široko područje upotrebe
- robusnost
- prenosivost
- otvoreni kod (**engl. *open source***) – dostupan besplatan razvojni alat kroz JDK koji sadrži nužne alate za razvoj aplikacija.

[7] Skup generičkih metoda koje rade s podacima.

Četvrta generacija programskih jezika (početak 80-ih godina)

Programski jezici četvrte generacije specijalizirani su za određeno područje, te nisu još standardizirani. Ovi programski jezici nisu samo namijenjeni programerima već su prilagođeni i krajnjim korisnicima.

Značajnije osobine jezika su:

- više su orijentirani korisniku
- jednostavnost i veća produktivnost programiranja
- nije potrebno veliko poznavanje programskog koda
- manje su mogućnosti za pogreške.

Jezici četvrte generacije razvijali su se kao potreba da se u programske jezike uvedu neproceduralni elementi. Neki od jezika su SQL, MATLAB, PL/SQL.

Peta generacija programskih jezika

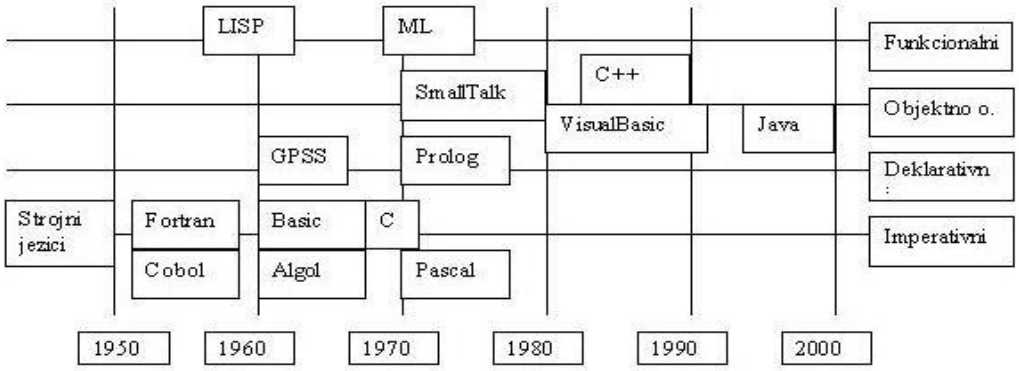
Programski jezici pete generacije još su uvijek u razvoju te im je budućnost neizvjesna. Osnovna ideja ove generacije programskih jezika je da se programer oslobodi rješavanja detalja implementacije i da se posveti u potpunosti problemu koji bi trebalo riješiti.

Najpoznatiji predstavnici su:

- Prolog (engl. *programming in logic*) – matematička logika

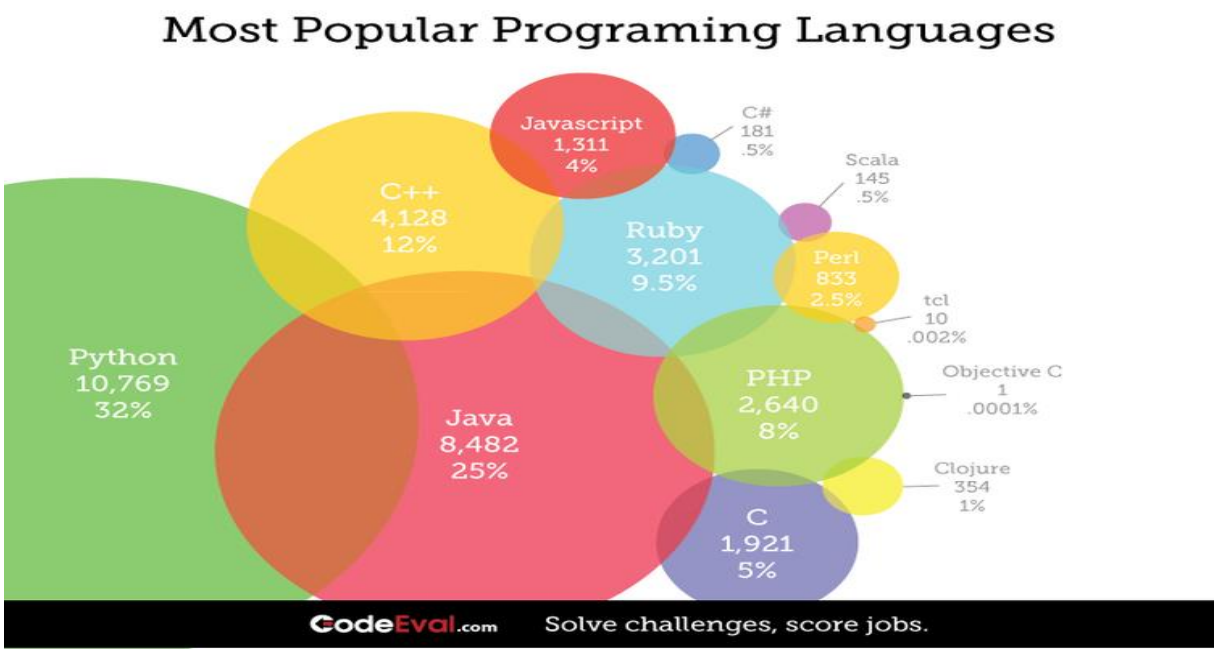
- Lisp – istraživanje na području umjetne inteligencije
- Logo – istraživanje u informatici (engl. *computer science*).

Slika 2. Razvoj programskih jezika



Izvor: http://laris.fesb.hr/predavanja/programskijezici.html

Slika 3. Najpopularniji programski jezici



Izvor: http://codeeval.squarespace.com/?offset=1374092206015

2.2. Usporedba programskog i prirodnog jezika

Jezik je sredstvo komunikacije i mora biti definiran na način da ga razumiju svi sudionici komunikacije. Prirodnim jezikom služe se ljudi, dok su umjetni jezici razvijeni za posebne namjene, te se koriste za pisanje programa i za rad na računalu. Sastoje se od:

- skupa simbola
- pravila za formiranje pojmova
- pravila za pretvaranje pojmova u izraze.

Za učinkovito korištenje programskoga jezika potrebno je njegovo proučavanje i razumijevanje triju osnovnih perspektiva, a to su:

- **leksik –** osnovni element jezika; kategorije leksika programskih jezika su riječi, operatori, identifikatori, ključne riječi te druge

- **semantika –** označava značenje (smisao) koje u programu ima neka konstrukcija programskog jezika

- **sintaksa –** njome se određuje jesu li pravilno formirani program ili dio programa u programskom jeziku. Često je vrlo složena i stoga je važno da programer nauči pravilno formirati rečenice.

Svaki program koji je napisan u programskom jeziku potrebno je prevesti u strojni. Program koji nije napisan na strojnom jeziku zove se izvorni program **(engl. *source program***). Prevođenje izvornog programa u strojni odvija se pomoću prevoditelja čija je zadaća analizirati leksik, sintaksu i semantiku programa, te generirati kod na strojni jezik.

Postoje dva načina prevođenja, a to su pomoću:

- interpretera, koji naredbu prevode liniju po liniju u trenutku izvođenja programa
- kompajleri **(engl. *compiler*)** – prevode programe napisane u izvornom jeziku u strojni jezik.

Kod prirodnog jezika sintaksom se definira način na koji se formiraju pravilne jezične strukture (rečenice), dok je semantika skup načina kojima se određuje rečenica odnosno pruža značenje rečenicama. Leksik je dio sintakse i on definira koje su riječi ili simboli ispravni u jeziku.

2.3. Učenje i poznavanje programskog jezika

Poznavanje programskog jezika do prije nekoliko godina bilo je prava nepoznanica te rezervirana samo za stručnjake i programere koji su svoje znanje stekli uz knjige i u školama. Danas su stvari ipak malo drugačije, razvojem tehnologije te interneta učenje programskih jezika i programiranja postalo je dostupno svima koji imaju interes naučiti programiranje, a da prije toga ne moraju biti stručnjaci. S obzirom na to da obrazovani programeri nisu bili u mogućnosti pokriti sve poslove koji postoje, stvaralo se više prostora za ljude koji nisu stručno osposobljeni za posao programera, a koji su naučili programirati sami ili uz pomoću različitih tečajeva.

U današnje je vrijeme puno lakše naučiti novi programski jezik nego prije vremena interneta, jer postoji mnoštvo *web*-stranica specijalizirano za edukaciju novih programskih jezika, te se uz trud i rad može uspješno savladati bilo da se radi o početniku ili onima koji već imaju određeno znanje. Uz inovativne aplikacije, koje se temelje na učenje uz rad, učenje programskog jezika može biti i zabavno.

Uz učenje na fakultetu, raznim tečajevima, putem interneta, uz poznavanje engleskog jezika, moguće je učiti programski jezik samostalno putem *web*-stranica.

Najpoznatije su:

edX – pruža tečajeve i nudi seminare sveučilišne razine u širokom rasponu disciplina

Coursera – nudi virtualno pohađanje predavanja profesora s brojnih svjetskih sveučilišta

Codeacademy.com – pruža tečajeve za *web*-dizajn CSS, HTML. Korisniku se na ekranu prikazuju zadaci s teorijom, redak za ispisivanje te prozor u kojem se kôd

pretvara odmah u *web*-stranicu. Korisnik sam prolazi kroz tečajeve te može pristupati zahtjevnijim projektima.

Code School – pruža isti princip kao i codeacademy gdje korisnik uči kroz vježbu

Codebabes.com – stranica koja pruža učenje osnova programiranja kroz videolekcije

Khan Academy – besplatni *online* tečaj za programiranje, koji sadrži razne vodiče te korisnika korak po korak vodi kroz učenje

Udemy – nije besplatan, nudi tečajeve koji uz pomoć mentora pomažu u svladavanju i učenju programskih jezika

3. ENGLESKI JEZIK – SVJETSKI JEZIK

Engleski je jezik izvorno jezik engleskog naroda. Danas je engleski glavni jezik Ujedinjenog Kraljevstva, Irske, Sjedinjenih Američkih Država, Kanade, Australije, Novog Zelanda i više od pedeset zemalja svijeta. Zanimljivo je da engleski jezik nije službeni jezik SAD-a, iako je to službeni jezik nekih američkih država.

U svijetu danas postoji preko 400 milijuna izvornih govornika engleskog jezika, a više od milijardu ljudi govori ga kao drugi jezik. Engleski jezik je vjerojatno treći jezik u smislu broja izvornih govornika (nakon mandarinskog i španjolskog) i najvjerojatnije najzastupljeniji jezik na planetu uzimajući u obzir izvorne govornike i one kojima je to drugi jezik.

Slijedom toga engleski se često opisuje kao svjetski jezik (engl. *world language*) ili „globalni jezik franca“[8] (*global lingua franca*). Najčešće je korišteni jezik u međunarodnom poslovanju, telekomunikacijama, objavljivanju novina i knjiga, znanstvenom izdavaštvu, masovnoj zabavi, diplomaciji i drugima.

Engleski jezik koristi sustav pisanja koji se temelji na klasičnoj latiničnoj abecedi koja ima dvadeset i šest slova. Pripada zapadnonjemačkoj skupini indoeuropskih jezika. Veći dio njegova jezika je germanski, također ima i snažan utjecaj latinskog i francuskog.

3.1. Povijest engleskog jezika

Povijest engleskog jezika započela je dolaskom triju germanskih plemena koja su osvajala Britaniju u 5. stoljeću poslije Krista. Plemena Angles, Saxons i Jutes prešla su Sjeverno more od onoga što je danas Danska i sjeverna Njemačka. U to vrijeme stanovnici Britanije govorili su keltski jezik. Većinu keltskih govornika protjerali su napadači prema zapadu i sjeveru, uglavnom na teritorije koje danas čine Wales, Škotska i Irska. Angles dolazi iz „Englaland“ i njihov jezik nazvan je „Englisc“ iz kojeg su izvedene riječi *Engleska* i *engleski*.

[8] Jezik koji se koristi kao zajednički jezik između ljudi koji govore različitim jezicima.

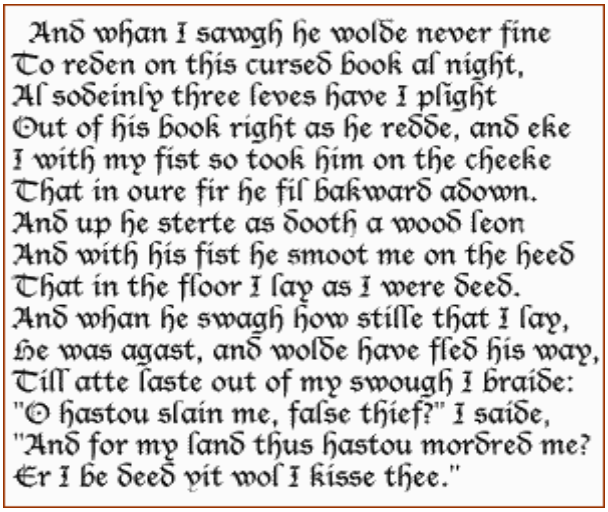
Old English (450. – 1100.)

Osvajačka germanska plemena govorila su sličnim jezicima, koji su u Velikoj Britaniji razvili ono što zovemo „stari Engleski“ **(engl. *Old English***). Nije zvučao i izgledao kao engleski danas, te bi današnji govornici engleskog imali poteškoća s razumijevanjem. Ipak oko polovica najčešće korištenih riječi na suvremenom engleskom jeziku ima stare engleske korijene. Stari engleski jezik govorio se do 1100. godine.

Midlle English (1100. – 1500.)

Godine 1066., osvajač William, vojvoda iz Normandije[9], napao je i osvojio Englesku, te sa sobom donio neku vrstu francuskog jezika, koji je postao jezik kraljevskog suda, vladajuće i poslovne klase. U tom periodu bilo je i podjela u jezične klase, gdje su niže klase govorile engleski, a više klase francuski jezik. U 14. stoljeću engleski jezik ponovno dominira u Britaniji, ali s mnogo francuskih riječi koje su dodane. Ovaj jezik zove se srednji engleski te je bio jezik velikog pjesnika Geoffreya Chaucera (1343. – 1400.), ali danas bi bio težak za govornike engleskog jezika.

Slika 4. Srednji engleski jezik – Geoffrey Chaucer


And whan I sawgh he wolde never fine
To reden on this cursed book al night,
Al sodeinly three leves have I plight
Out of his book right as he redde, and eke
I with my fist so took him on the cheeke
That in oure fir he fil bakward adown.
And up he sterte as dooth a wood leon
And with his fist he smoot me on the heed
That in the floor I lay as I were deed.
And whan he swagh how stille that I lay,
He was agast, and wolde have fled his way,
Till atte laste out of my swough I braide:
"O hastou slain me, false thief?" I saide,
"And for my land thus hastou mordred me?
Er I be deed yit wol I kisse thee."

Izvor: http://itsliterature7.blogspot.hr/2016/12/old-english-vs-middle-english.html

[9] Dio moderne Francuske.

Early modern English (1500. – 1800.)

Krajem perioda *midlle englisha*, započela je iznenadna i izražena promjena u izgovoru, a izgovoreni glasovi bili su kraći i kraći. Od 16. stoljeća, Englezi kreću s kolonizacijom velikih područja diljem svijeta, te uspostavljaju kontakt s mnogim narodima iz cijeloga svijeta što je uz renesansu klasičnog učenja doprinijelo ulasku novih riječi u jezik. Izum i upotreba tiskarskog stroja daju doprinos za standardizaciju engleskog jezika jer je u tisku postojao zajednički jezik, te su knjige postale jeftinije i više je ljudi naučilo čitati. Pravopis i gramatika postali su fiksni, a govor iz Londona, gdje se nalazila većina izdavačkih kuća, postao je standard. Godine 1604. objavljen je prvi engleski rječnik.

Late Modern English (1800. – danas)

Glavna razlika između ranog modernog i kasnog engleskog jezika je rječnik. Suvremeni engleski ima puno više riječi što proizlazi iz dva glavna čimbenika:

- industrijska revolucija i tehnologija stvorili su potrebu za novim riječima
- Britansko Carstvo na svojoj veličini pokrilo je jednu četvrtinu površine zemlje, a engleski jezik usvojio strane riječi iz mnogih zemalja svijeta.

Vrste engleskog jezika

Od 1600. godine engleska kolonizacija Sjeverne Amerike rezultirala je stvaranjem izrazite američke raznolikosti engleskog jezika. Američki engleski nekako je više poput engleskog iz doba Shakespearea, nego što je moderni britanski engleski. Neki britanski nazivi koje Britanci smatraju amerikaniziranima zapravo su izvorni britanski izrazi koji su sačuvani u kolonijama, a izgubili su se neko vrijeme u Velikoj Britaniji. Španjolski je također imao utjecaja na američki engleski (a kasnije i britanski engleski) poglavito prilikom naseljavanja američkog zapada. Francuske riječi (preko

Louisiane) i zapadnoafričke riječi (preko robne trgovine) također su utjecale na američki engleski (u određenoj mjeri i na britanski engleski).

Danas je američki engleski osobito utjecajan zbog dominantnosti SAD-a u kinu, televiziji, popularnoj glazbi, trgovini i tehnologiji (uključujući i internet). Danas postoje mnoge druge inačice engleskog jezika širom svijeta uključujući primjerice australski engleski, novozelandski engleski, kanadski engleski, južnoafrički engleski, indijski engleski i druge.[10]

Slika 5. Članovi germanske jezične obitelji



Izvor: https://www.englishclub.com/history-of-english/

3.2. Engleski jezik i programiranje

Današnji mediji, kao što su internet, televizija te tisak, daju nam neograničeno znanje o temama koje nas zanimaju. Usprkos rastu interneta na drugim jezicima, većina dostupnih informacija i stranica su na engleskom jeziku te je vrlo teško prevesti svaku stranicu na različite jezike.

Zbog globalne dominacije Velike Britanije te kasnije Sjedinjenih Američkih Država, engleski jezik je postao dobro rasprostranjen i naučen kao poslovni jezik. Većina ranih računalnih znanosti nastala je na području Sjedinjenih Američkih Država, gdje

[10] https://www.englishclub.com/history-of-english/

se govori engleski jezik iako nije službeni, te dijelom u Velikoj Britaniji i Kanadi, što je jedan od razloga da je većina glavnih programskih jezika pisana na engleskom jeziku. Iz tog razloga engleski je postao jezik STEM-a (engl. *science, technology, engineering, mathematics*)[11].

Engleski jezik je najčešće korišteni jezik u programskome svijetu. Od samih početaka tehnološke povijesti, engleski jezik je jezik pojedinca i tvrtki koji najizravnije pokreću promjene i inovacije. Iako su neki programeri ovo osporili, jezični razlog je jednostavna struktura i sintaksa engleskog jezika. Budući da su programski jezici toliko temeljeni u njihovoj potrebi za riječima, to čini engleski kao vrlo pogodan za programiranje jer su riječi općenito kratke i nema dijakritičkih znakova i akcenta.

U upotrebi programski stručnjaci više se bave načinom na koji je jezik izrađen od konkretnih ključnih riječi. Zbog toga je engleski koji se koristi u programiranju više „računalni jezik“ nego engleski jezik koji se koristi u razgovoru. Često programeri ne znaju izgovor pojedinih riječi, ali znaju kako se ta riječ koristi u kodu.

Široka upotreba engleskog jezika nije zadržala neke programere za stvaranje programskih jezika koji crpe sintakse iz drugih jezika. Postoji mnogo programskih jezika koji se temelje na jezicima koji nije engleski, međutim većina njih nije u upotrebi, koristi se rijetko ili služe za uvodni ili obrazovni alat za početnike. Važno je napomenuti da postoje iznimke. Neki su stvoreni na ruskom, tijekom Sovjetskog Saveza.

3.3. Važnost engleskog jezika za učenje programskog jezika i programera

Poznavanje engleskog jezika nije uvjet za učenje programskog jezika, ali je jedan od bitnijih faktora za učenje programiranja i općenito za posao programera, te pruža

[11] Znanost, tehnologija, inženjerstvo i matematika – termin vezan za američki obrazovni sustav.

veliku prednost. Za učenje programskog jezika potreban je bar osnovni minimum gdje kandidat razumije pisani engleski jezik. Sve više materijala, vodiča, foruma je na engleskom jeziku te nepoznavanje barem osnova onemogućilo bi kvalitetno učenje, a kasnije i programiranje.

Ozbiljnije bavljenje programiranjem te sličnim poslovima ipak zahtijeva viši nivo poznavanja engleskog jezika, što više to bolje. Poznavanje engleskog jezika ima važnu ulogu te olakšava komunikaciju i uspješno učenje programiranja, obavljanje poslova i komunikaciju na internetu. Poslovi koje obavljaju programeri te informatički stručnjaci odvijaju se sve više u timovima te putem interneta, često i na različitim mjestima i stoga je više nego poželjno poznavanje engleskog jezika radi lakše komunikacije i uspješnosti programera te tvrtki u informatičkom svijetu. Engleskim jezikom govori se širom svijeta tako da ga odabiru osobe koje govore različitim jezikom radi lakše komunikacije.

Postoji više razloga zbog kojih je potrebno da programer poznaje engleski jezik:

Uspješna i efikasna komunikacija – tvrtke ili programeri koji rade na daljinu koriste engleski jezik putem videokonferencija, Skypea i drugih za vođenje sastanaka. Sposobnost izražavanja na engleskom jeziku može izravno utjecati na sposobnost ljudi oko nas da prosuđuju našu razinu profesionalne stručnosti, a i samim time pridonijeti boljoj komunikaciju unutar tvrtke, što može rezultirati u konačnici boljim poslovanjem. Većina programera dolazi iz zemalja u kojima je visoko obrazovanje na engleskom jeziku: SAD-a, Kanade, UK-a, Irske, Indije. Postoji dobar razlog zašto Indija ima više programera nego Kina, zbog toga jer je u Indiji srednje školstvo i visoko školstvo uvijek na engleskom jeziku.

Dokumentacija na engleskom jeziku – za tehničku dokumentaciju vezanu za softver preporučuje se engleski jezik jer je standard u softverskoj industriji. Programski jezici i kodovi napisani su na engleskom jeziku tako da bi trebala biti i njihova dokumentacija. Bez obzira na izbor jezika, jednog dana će dokumentacija

trebati biti prevedena i stoga je potrebno razmisliti o troškovima prevođenja. Ako i postoji prevedena dokumentacija, nije sigurno postoje li u njoj greške.

Resursi na engleskom jeziku – na internetu, većina *web*-stranica izrađena je i pisana na engleskom jeziku. Primarni je jezik tiska te je najviše knjiga i novina pisano na engleskom jeziku. Zbog dominantnosti u međunarodnoj komunikaciji, moguće je naći informacije o gotovo svakom predmetu. Na hrvatskom jeziku postoji malo literature za učenje programiranja i poznavanje engleskog jezika je od velike važnosti za učenje – kako za onoga koji uči programirati, tako i za već iskusnog programera.

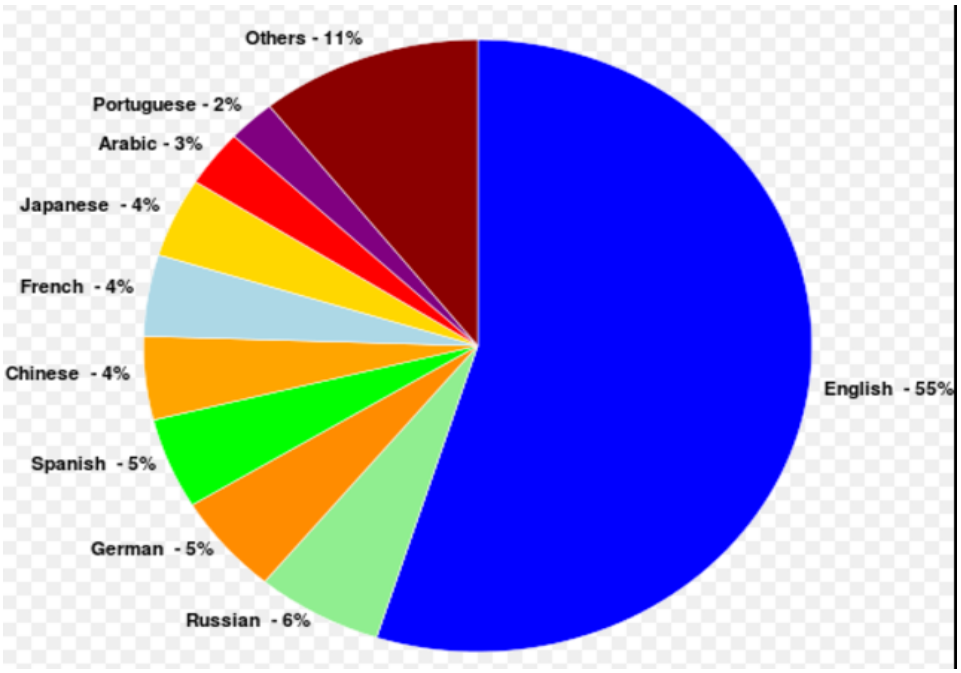
Budući da je većina softverskih dokumentacija, članaka i vodiča pisana na engleskom jeziku, učenje engleskog jezika korisno je i vrijedno za programere koji se ozbiljnije misle baviti tim poslom.

Engleski jezik često se opisuje kao jezik računalstva. U usporedbi s drugim znanostima gdje su latinski i grčki glavni izvori rječnika, računalna znanost posuđuje riječi iz engleskog jezika. Ključne riječi koje se upotrebljavaju gotovo za sve programske jezike su na engleskom jeziku.

Mnogo edukativnih materijala, tečajeva, videouradaka, tekstova je snimljeno i napravljeno na engleskom jeziku te njegovo poznavanje omogućava napredovanje i usavršavanje u učenju programiranja ili nekog drugog područja.[12]

[12] https://www.quora.com/Is-the-English-language-too-important-to-learn-a-programming-language

Slika 6. Jezici sadržaja *web*-stranica



Izvor: https://en.wikipedia.org/wiki/Global_Internet_usage

Može se reći da je poznavanje engleskog jezika vrlo bitan faktor za učenje programiranja, no nije uvjet, ali je od velike prednosti. Jedan od glavnih razloga je taj da su razni vodiči i literatura napisani na engleskom jeziku. Također, mnoge *web*-stranice i forumi gdje početnici, a i programeri mogu potražiti savjete koji mogu biti od velike koristi također su na engleskom jeziku. Za nesmetano učenje potreban je minimalni nivo razumijevanja pisanog engleskog jezika.

4. ENGLESKI JEZIK I C++

C++ je programski jezik opće namjene koji je razvio Bjarne Stroustrup te ima podršku za objektno orijentirano programiranje. Proširenje je programskog jezika C te je moguće programirati u C++ u „C stilu“. Smatra se jezikom srednje razine jer obuhvaća jezične značajke visokih i niskih razina. U početku jezik je nazvan „C s klasama“ jer ima svojstava jezika C dodatnim konceptima, no 1983. godine je preimenovan u C++.

C++ je jedan od najpopularnijih programskih jezika koji ima zbirke unaprijed definiranih razreda koji su tipovi podataka koji se mogu instancirati više puta. Također olakšava deklaraciju korisničkih definiranih klasa. Moguće je definirati više objekata određene klase kako bi se implementirale funkcije unutar klase. Objekti se mogu definirati kako instance stvorene za vrijeme izvođenja. Ove klase mogu naslijediti i nove klase koje zauzimaju javne i zaštićene funkcije prema zadanim postavkama. Uključuje nekoliko operatora kao što su operatori usporedbe, aritmetički i logički operatori. Nekoliko bitnih pojmova unutar C++ programskog jezika uključuje polimorfizam, virtualne i prijateljske funkcije, predloške, imenske prostore te pokazivače.[13]

[13] https://www.techopedia.com/definition/26184/c-programming-language

4.1. Koncepti programiranja

Danas u programiranju dominiraju dva koncepta: strukturirano i objektno orijentirano programiranje.

Koncept strukturiranog programiranja – sastoji se od tri osnovna koncepta od kojih je prva hijerarhijska struktura programa koja se postiže dekompozicijom obrade podataka na kontrolne strukture: sekvencu, selekciju i iteraciju.
Drugi koncept je taj gdje je deklaracija podataka odvojena od obrade što znači da na bilo kojem mjestu u programu možemo imati deklaraciju dok se treći koncept sastoji od potprograma koji imaju određenu zadaću i izdvajaju se iz cjeline te se mogu pozivati iz bilo kojeg dijela programa.

Koncept objektno orijentiranog programiranja – nadogradnja na strukturirano programiranje, te ga proširuje s novim mogućnostima. Glavna svrha C++ programskog jezika bila je dodavanje orijentacije objekata na jezik C koji je u sebi jedan od najmoćnijih programskih jezika. Srž čistog orijentiranog na objektno programiranje je stvaranje objekta u kodu koji ima određena svojstva i metode.

Postoji nekoliko osnovnih pojmova koji čine temelj objektno orijentiranog programiranja, a to su:

Objekt (engl. *object*) – osnovna jedinica programiranja orijentiranog na objekt. Tu su i podaci i funkcija koji djeluju na objekt koji djeluju na podatke koji se grupiraju kao jedinica koja se zove objekt.

Klasa (engl. *class*) – prilikom definiranja klase, definira se nacrt objekta, što zapravo ne definira nikakve podatke, ali definira ono što znači klasno ime, tj.

koji će se objekt sastojati od klase i koje se operacije mogu izvršiti nad takvim objektom.

Apstrakcija (engl. *abstraction*) – odnosi se na pružanje samo osnovnih informacija vanjskom svijetu i skrivajući u pozadini njihove detalje.

Enkapsulacija (engl. *encapsulation*) – stavlja podatke i funkcije koji rade na podacima na isto mjesto. Prilikom rada s proceduralnim jezicima nije uvijek jasno koje funkcije rade na kojim varijablama, ali objektno orijentirano programiranje pruža okvir za postavljanje podataka i relevantnih funkcija zajedno u istom objektu.

Nasljeđivanje (engl. *inheritance*) – jedan od najkorisnijih aspekata objektnog programiranja je ponovna upotreba koda. Nasljeđivanje je proces formiranja nove klase iz postojeće klase nazvane kao osnovna klasa, formira se nova klasa nazvana kao izvedena klasa. Ovo je vrlo važan koncept objektnog programiranja jer ova značajka smanjuje veličinu koda.

Polimorfizam (engl. *polymorphism*) – označava mogućnost korištenja operatora ili funkcije na različite načine, drugim riječima, davanje različitih značenja ili funkcija operatorima ili funkcijama. To je funkcija ili operator koji funkcionira na mnogo načina koji se razlikuju od korištenja.

Preopterećenje (engl. *overloading*) – koncept preopterećenja također je grana polimorfizma.

4.2. Izrada programa u C++

Izrada programa sastoji se od davanja uputa računalu u svrhu rješavanja nekoga zadatka i problema. Obično se danas programi pišu ne nekom jeziku više razine, koji donekle razumiju i ljudi uz bar malo poznavanje engleskog jezika. Sastoji se od

naredbi s čijim se kombiniranjem izrađuje izvorni kod (engl. *source kod*). Putem provoditelja (engl. *compiler*) i povezivača (engl. *linker*), prevodi se u izvedbeni kod.

„Postoje četiri faze izrade programa:

1. Pisanje izvornog koda – izvorni kôd može se pisati u bilo kojem programu za uređivanje teksta (engl. *text editor*), međutim velika većina današnjih prevoditelja i povezivača isporučuje se kao cjelina zajedno s ugrađenim programom za upis i ispravljanje izvornog koda. Te programske cjeline poznate su pod nazivom integrirane razvojne okoline (engl. Integrated Development Environment, IDE).

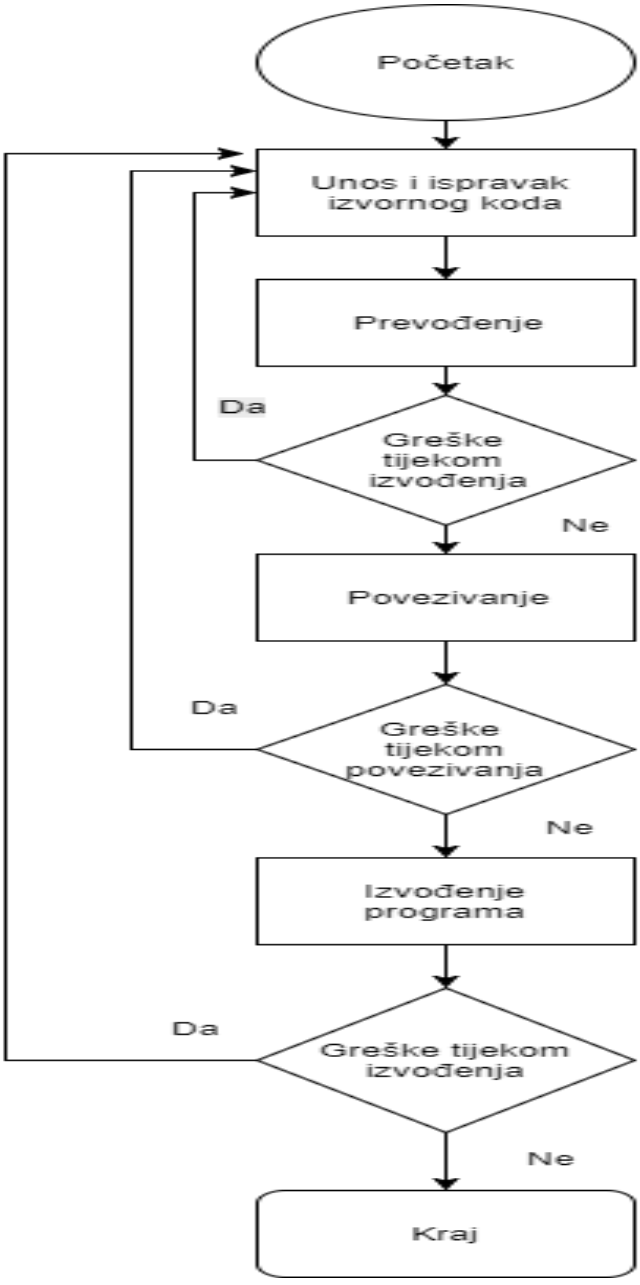
2. Prevođenje izvornog koda – u integriranim razvojnim okolinama program za prevođenje pokreće se pritiskom na neku tipku na zaslonu, pritiskom odgovarajuće tipke na tipkovnici ili iz nekog od izbornika (engl. *menu*). Prevoditelj tijekom prevođenja provjerava sintaksu napisanog izvornog koda i u slučaju uočenih ili naslućenih pogrešaka ispisuje odgovarajuće poruke o pogreškama ili upozorenja.

3. Povezivanje u izvedbeni kôd – nakon što su ispravljene sve uočene pogreške prilikom prevođenja i nakon što je kôd ispravno preveden, pristupa se povezivanju objektnih kodova u izvedbeni. U većini slučajeva objektni kod dobiven prevođenjem programerova izvornog koda treba povezati s postojećim bibliotekama (**engl. libraries**). Biblioteke su datoteke u kojima se nalaze već prevedene gotove funkcije ili podaci.

4. Testiranje programa – uspješnim povezivanjem dobiva se izvedbeni kod programa. Taj kod će raditi upravo ono što je programer zadao naredbama izvornog koda, no sasvim je moguće da to ne odgovara onome što je izvorno zamislio. Program će sadržavati logičke pogreške. Da bi program bio potpuno korektan, programer treba testirati program b da bi uočio i ispravio te

pogreške, što znači ponavljanje cijelog postupka u lancu ispravljanja izvornog koda."[14]

Slika 7. Izrada programa



Izvor: Demistificirani C++. Julijan Šribar, Boris Motik, 4. izdanje

[14] Demistificirani C++, Julijan Šribar, Boris Motik, 4. izdanje

4.3. Osnovni koncept programskog jezika

Kao što kod prirodnog jezika postoje slova, riječi i rečenice, tako kod programskog jezika postoje simboli, leksemi i izrazi.
Proučavanje programskih jezika dijeli se na sintaksu i semantiku:

Sintaksa – skup pravila koja pruža mogućnost formiranja konkretnih programa. Provjerava da li niz znakova pripada jeziku. Elementi sintakse su jedinice, leksemi (eng. *lexemes*) koji se mogu promatrati kao niz znakova u računalnom programu. Uključuju konstante, operatore, identifikatore i ključne riječi. Sintaksa naredbi zadana je najčešće sintaktičkim dijagramom poznati kao Exstended Backus – Naur forme (EBNF) koja opisuje jezik matematičkom metodom.
Primjer ispravne sintakse petljom *if else*:

if (a>4) max=6,2; else max =a;

Petlja označava da ako je uvjet zadovoljen, izvršit će se prva naredba, a ako nije, onda će se izvršiti druga. Najčešće greške u sintaksi su nezatvorene zagrade u aritmetičkom izrazu, pogrešno napisana riječ te spojene dvije riječi u jednu.

Semantika – označava značenje programa u cjelini. Bavi se proučavanjem računalnih modela te značenjem programskih jezika matematičkim proučavanjem.
Statička semantika provjerava pravila za vrijeme izvođenja programa odnosno provjerava koje forme su dozvoljene, a koje nisu. Obično određuje pravila kojima se određuje koja ograničenja postoje za tipove podataka dok dinamička semantika opisuje efekte tijekom izvođenja programa.[15]

[15] http://ttl.masfak.ni.ac.rs/SUK/Programiranje%20i%20algoritmi%201.pdf

a. Ključne riječi programskog jezika C++

Programski jezik C ++ koristi vlastite specijalne riječi koje imaju svoja značenja, a nazivamo ih ključnim riječima. Tablica niže prikazuje te ključne riječi, koje imaju ishodište u engleskom jeziku.

Slika 8. Ključne riječi

Ključna riječ	Engleska riječ	Značenje
ASM	assembly	ubacivanje asembler instrukcije
AUTO	automatic	deklariranje lokalne varijable
BOOL	boolean	deklariranje logičke varijable
BREAK	break	izađi iz petlje
CHAR	character	deklariranje znakovne varijable
CLASS	class	deklariranje klase
CONST	constant	deklariranje nepromjenjive varijable
CONTINUE	continue	nastavlja petlju
DELETE	delete	oslobađa memorije
DO	do	naziv petlje
DOUBLE	double	deklariranje duple cjelobrojne varijable
ELSE	else	alternativno rješenje if petlje
EXTERN	externally	deklariranje varijabli iz drugog programa
FALSE	false	logička vrijednost za laž
FLOAT	float	deklariranje float varijable
FOR	for	naziv petlje

GOTO	go to	skoči na drugi dio programa
IF	if	naziv petlje
INT	integer	deklariranje cjelobrojne varijable
LONG	long	deklariranje duge cjelobrojne varijable
NAMESPACE	name space	deklaracija elemenata standardne C++ biblioteke
NEW new	new	otvara novo mjesto u memoriji
PRIVATE	private	privatni članovi klase
PROTECTED	procteted	zaštićeni članovi klase
PUBLIC	public	javni članovi klase
RETURN	return	povratak iz funkcije
SHORT	short	deklariranje kratke cjelobrojne varijable
SIGNED	singed	modifikacija deklaracije tipa varijable
SIZEOF	size	vraća duljinu varijable
STATIC	statical	kreira parametar mjesta pohrane za varijablu
STRUCT	structural	kreiranje nove strukture
SWITCH	swich	naziv petlje
TEMPLATE	template	kreira generičke funkcije
THROW	throw	ubacuje iznimku
TRUE	true	logička vrijednost za istinu
TRY	try	izvršava iznimku
TYPEDEF	type definition	kreiranje imena tipa podataka
TYPEID	type id	opis objekta
USING	using	Deklariranje područja imena
VOID	void	deklariranje funkcije koja ne vraća rezultat
WHILE	while	naziv petlje

Izvor: Osnove programiranja u programskom jeziku c++, Giorgio Sinković, Igor Škorić

5. ZAKLJUČAK

U današnje vrijeme velikog tehnološkog napretka, informatika kao znanstvena grana vrlo je bitna za daljnji razvoj našega društva. Razvoj informatike usko je povezan s programskim jezicima i oni kao takvi zaslužni su za velik napredak u proteklim desetljećima. Kako su se razvijali programski jezici, pojavio se i novi posao na tržištu rada, a to je posao programera.

Programeru je od ključnog značaja poznavanje engleskog jezika, prvenstveno radi samog programiranja jer je sintaksa bazirana na engleskom jeziku, ali i radi same komunikacije s kolegama koja se odvija često na engleskom jeziku, kao i pisanja dokumentacije i uputa te traženja literature.

Engleski jezik ima široku upotrebu u znanosti, informatici, diplomaciji, turizmu, gospodarstvu i mnogim ostalim područjima. Primjerice, više od polovice svih poslovnih ugovora napisano je na engleskom jeziku. Engleski jezik je službeni jezik u 46 zemalja širom svijeta, ali govorni u još više zemalja i ima oko 375 milijuna govornika.

Engleski jezik i programiranje su kotači koji pokreću tehnološki napredak današnjeg društva i djeluju kroz zajedničku sinergiju na napretku i razvoju ljudske civilizacije.

Slike

Slika 1. Programski jezici 4
Slika 2. Razvoj programskih jezika 9
Slika 3. Najpopularniji programski jezici 10
Slika 4. Srednji engleski jezik – Geoffrey Chaucer 15
Slika 5. Članovi germanske jezične obitelji 17
Slika 6. Jezici sadržaja *web*-stranica 21
Slika 7. Izrada programa 26
Slika 8. Ključne riječi 28

Literatura

Knjige:

1. Julijan Šribar, Boris Motik, (2014), Demistificirani C++, Element Zagreb
2. Danijel Radošević (2007), Programiranje 2, Tiva Tiskara Varaždin
3. Giorgio Sinković, Igor Škorić (2008), Osnove programiranja u programskom jeziku c++, Sveučilište Jurja Dobrile u Puli, Pula.
4. Stroustrup, Bjarne (2013), The C++ programming language, fourth edition, Pearson Educations Inc

Internet:

C++ Programming Language

URL:https://www.techopedia.com/definition/26184/c-programming-language

Programski jezik

URL:https://tesla.carnet.hr/pluginfile.php/21710/mod_resource/content/2/COURSE_2666625_M/my_files/Sadrzaj/Poglavlje%202/2_2_4%20Programski_jezik.htm?embed=1e

History of English

URL:https://www.englishclub.com/history-of-english/

10 Reasons To learn English

URL:http://www.experienceenglish.com/social-english/articles/10-reasons-learn-english-
http://homepage.cs.uri.edu/faculty/wolfe/book/Readings/Reading13.htmhttp://docness.readthedocs.io/en/latest/conventions/documentation-style-guide.html

Sintaksa i semantika programskog jezika
URL**:** http://ttl.masfak.ni.ac.rs/SUK/Programiranje%20i%20algoritmi%201.pdf

Computer programming language
Url: https://www.britannica.com/technology/computer-programming-language/Introduction

10 Rules for to Bild English Communication Skills for Developers
URL: http://www.mypassionfor.net/2014/12/16/10-rules-to-build-english-communication-skills-for-developers/

Is the English language too important to learn a programming language

URL: https://www.quora.com/Is-the-English-language-too-important-to-learn-a-programming-language

Sažetak

Cilj ovoga rada je ukazati na važnost engleskog jezika pri učenju programskog jezika te ozbiljnijem bavljenju poslom programera. Sam razvoj programskih jezika i programiranja usko je povezan s engleskim jezikom jer su prvi programski jezici razvijeni u državama engleskog govornog područja pa su i samim time ključne riječi programskog jezika preuzete iz engleskog jezika.

Znanje engleskog jezika uvelike olakšava učenje programiranja zbog dostupnosti literature koja je većinom na engleskom jeziku. Također, za ozbiljnije bavljenje poslom programera od velike je prednosti znanje engleskog jezika jer olakšava komunikaciju, pisanje dokumentacije, vođenje sastanaka, kao i predstavljanje programa.

C++ je jedan od najčešće korištenih programskih jezika i kao takav je najviše rasprostranjen i spada među programske jezike više razine. Velik dio svojstava naslijedio je od programskog jezika C, pa ga se još naziva i C s klasama, te je u osnovi objektno orijentirani programski jezik.

Kako u svakodnevnom govoru, tako i u informatici odnosno programiranju, velik je utjecaj engleskih riječi, koje se više ne prevode nego ostaju u izvornom engleskom obliku.

Tako da je ključno da programer mora poznavati engleski jezik, kako bi bio uspješan u svom radu.

Ključne riječi: c++, programer, objektno-orijentiran, engleski jezik

Abstract

The aim of this thesis is to emphasize the importance of English in learning the programming language and the serious work of a programmer.

The development of programming languages and programming is closely related to the English language because the first programming languages are developed in English speaking countries, so the programming languages are taken from the English language.

English language knowledge greatly facilitates programming learning because of the availability of literature that is in English. Also, for more serious business engagement, knowledge of English is a great advantage because it facilitates communication, writing documentation, conducting meetings, and presenting programs.

C ++ is the most commonly used programming language and as such is the most widespread and is among the top level programming languages. Most of the attributes are inherited from the programming language C, so it is still called the C class, and is in the basic object-oriented programming language.

As in everyday speech, as well as in informatics or programming, there is a great influence of English words, which are no longer translated but remain in the original English language.

So it is crucial for a programmer to know English in order to be successful in his work.

Key words: C++, programmer, object-oriented, English language