TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Karl-Erik Kald 193802IACB

Programmerimiskeelte võrdlus ja analüüs

Bakalaurusetöö

Juhendaja: Peeter Ellervee

Tallinn 2022

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Karl-Erik Kald

08.03.2022

Annotatsioon

Lõputöö eesmärgiks on uurida, võrrelda ja analüüsida erinevaid programmeerimiskeeli ja koostada ülevaatlik töö.

Selle eesmärgi saavutamiseks on vaja uurida ja läbi töötada eelnvalt koostatud töid sarnastel teemadel ja vaadata kuidas ning mida eelnevalt tehtud on ning selle põhjal hakata oma tööd tegema.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 31 lehel, 31 joonist ja 1 lisa.

Abstract

Differences of Programming Languages and Analysis

The aim of this work is to study, compare, and analyze different programming languages and put together a concise paper.

In order to achieve this goal, it is necessary to study and work through papers that have been made on similar subjects and research what and how has been done before. Then author can start working on his paper.

The thesis is in Estonian and contains text on 32 pages, 31 drawings and 1 appendix.

Sisukord

1 Sissejuhatus

Käesolev bakalaureusetöö keskendub programmeerimisekeeltele, mis on olnud kasutuses, kas väga kauges minevikus või on kasutusel ka tänapäeval. Programmeerimisekeelte maailmas on olnud mitmeid keeli, mis on andnud alust paljudele arendustele, mis on kõikide inimeste elu muutnud ning see jätkub ka tulevikus. Paljudest keeltest, millest siin töös räägitakse, on inimesed kindlasti ennegi kuulnud, kuid täpselt ei tea, kust nad välja arenesid ning milleks neid kasutatakse. See töö peaks andma hea ülevaate suurematest ja olulistemast programmeerimiskeeltest ning näha teed, kuidas need veel tänapäeval kasutust leiavad.

Lisaks sellele võrreldakse siin töös veel erinevaid keeli ning antakse ülevaade kui palju mälu nad kompileerides kasutavad ning kui kaua kompileerimine aega võtab.

Ühe näitena vaadatakse ka C++ keelt ning tema populaarsemaid kompilaatoreid ning kuidas nad kompileerides teineteisest erinevad.

Antud töö esimese osa eesmärgiks on välja uurida, mis on olnud ja on praegu, enim kasutatumad ning kõige suurt mõju avaldanud programmeerimiskeeled ja anda leida võimaluse korral nendes keeltes „Hello World“ koodi näide. Lisaks tuleb anda nendest keeltest üldine ülevaade ehk kes oli keele autor, mis oli antud keele kasutamise valdkonnad ning eesmärgid ja mis aastal keel välja töötati.

Töö teises osas on eesmärgiks võrrelda tänapäeval kasutuses olevaid programmeerimiskeeli. Uurida välja iga keele kohta tema positiivsed ja negatiivsed küljed ning lõpuks kõikide keelte vahelisi parameetreid uurida. Parameetrite all peab autor silmas programmi töö aja kiirust ning mälu kasutust sekundis.

Kolmandas osas uurib autor kompilaatorit ning kuidas ta täpsemalt töötab ja mis protsesse ta läbib. Töö kolmanda osa teises pooles võrldleb autor *C++* keele kompilaatoreid andes nende kohta üldise kirjelduse ning kompileerimise kiiruse. Kompileeritud kood on leitav materjalide alt.

2 Suuremate ja tähtsamate keelte ülevaade

Järgnevas peatükis annab autor ülevaate kõige esimestest teada olevatest, kõige suurtematest ning kõige mõjukamatest programmeerimiskeeltest. Iga keele kohta pannakse kirja tema loomise aasta, keele autor, mille jaoks keel algselt loodud oli ning kas, kus ja kuidas ta on kasutuses tänapäeval.

2.1 Ada Lovelace

Esimene teadaolev programmeerimiskeel loodi aastal 1843 Ada Lovelace'i poolt ning seda kutsuti Ada Lovelace'i masina algoritmiks (i.k. *Ada Lovelace's machine algorithm*). Seda keelt loetakse programmeerimiskeelte aluseks. [1] Lovelace'i poolt loodud algoritm pidi arvutama keerulisi Bernoulli arve. Bernoulli arvud on keeruline ratsionaalarvude jada, mida kasutatakse tihti arvutamises ning aritmeetikas. Tänapäeval on tegemist ajalooga ning praktilist kasutust loodud algoritm enam ei leia.[2]

2.2 Plankalkül

Esimene "päris" programmerimiskeel on aga 1944-1945 aastatel loodud *Plankalkül*, mis loodi sakslasest inseneri ja arvutiteadlase Konard Zuse poolt. Kuna sel ajal toimus II maailmasõda ja fokuseeriti sõjale ning Zuse ise oli loonud uue Z3 arvuti ja tegeles põhiliselt selle reklaamimisega, siis jäi *Plankakül* tahaplaanile ja see tähendas selle keele läbikukkumist. Sellegi poolest oli inseneritööks loodud keel esimene kõrgetasemeline programmeerimiskeel. Tänapäeval see keel enam kasutuses ei ole.

```
R1.1(V0[:SIG]) => R0

R1.2(V0[:M X SIG]) => R0

0 => I | M + 1 => J

[W [ I < J -> [ R1.1(V0[I: M X SIG]) => R0 | I + 1 => I ]
] ] END

R1.3() => R0

'H';'E';'L';'L';'O';',';' ';'W';'O';'R';'L';'D';'!' =>
Z0[: M X SIG] R1.2(Z0) => R0

END
```

Joonis 1. Hello World *Plankakül*'is [3]

2.3 Assembly Language

Assembly Language on madalatasemeline programmeerimis keel, mille eesmärk on suhelda otse arvuti riistvaraga. Erinevalt masinakeelele, mis koosneb binaarsetest ja kuueteistkümnendsüsteemi sümbolitest, on *assembly language* inimestele loetav. Sellised madala tasemelised keeled väga tähtsaks osaks kõrge tasemeliste programmerimiskeelte välja arenemises nagu näiteks *Python* või *JavaScript*. Tänapäeval kasutatakse seda keelt veel ikka, et otseselt riistvara mõjutada, ligi pääseda spetsiifilistele protsessori juhistele või adresseerida kriitilisi jõudlusprobleeme. [4]

Assemblerkeel loodi 1948 aastal David Wheeler'i poolt EDSAC-ile (Electronic Delay Storage Automatic Calculator).[5]

```
org  0x100  ; .com files always start ;256 bytes into the
;segment
; int 21h is going to want...

mov  dx, msg      ; the address of or message in dx
mov  ah, 9        ; ah=9 - "print string" sub-function
int  0x21         ; call dos services
mov  ah, 0x4c     ; "terminate program" sub-function
int  0x21         ; call dos services

msg  db 'Hello, World!', 0x0d, 0x0a, '$'   ; $-terminated
;message
```

Joonis 2. Hello World *Assembly Language*'s [6]

2.4 Shortcode

Shortcode oli esimene kõrgetasemeline keel. Selle keele loomise peale tuli 1949 aastal John McCauley, kuid William Schmitt oli see, kes rakendas seda BINAC (Binary Automatic Computer) arvutil samal aastal ning 1950 aastal UNIVAC-i (Universal Automatic Computer) peal. *Shortcode* läbis mitu sammu: esiteks ta konverteeris tähelised osad koodist numbriteks ning seejärel need numbrid masinakoodi. See oli „tõlkija", mis tähendas, et ta tõlkis HLL (High Level Language e. kõrge tasemeline keel) osad ja täitis neid üks haaval, mis oli aeglane protsess. Kuna nende koodi osade täitmine oli nii aeglane, siis kasutatakse „tõlkijaid" väljaspool programmi arendust väga harva. Kuna kuskil pole infot, et see keel oleks veel kuskil kasutuses, siis teeb autor loogilise eelduse, et seda keelt enam aktiivselt kasutuses pole. [7]

2.5 Autocode

Autocode oli üldine termin, mida kasutati programmeerimiskeelte perekonna jaoks. Esimesena arendas seda Alick Glennie Manchester Mark 1 arvuti jaoks 1952 aastal. *Autocode* oli esimene kompileeritud keel, mis tähendas seda, et kood tõlgiti otse masinakeelde kasutades programmi, mida kutsuti kompilaatoriks. *Autocode*'i kasutati veel ka Ferranti Pegasus'e ja Sirius'e masinatel. Tuntumad keeled, mida *Autocode*'i all tuntakse on COBOL ning Fortran. [8]

2.6 Fortran

Fortran: *Fortran* või *FORmula TRANslation* on 1957 aastal John Backus'e poolt loodud programmeerimiskeel, mis lühendas programmeerimise protsessi ning muutis programmeerimise rohkem kättesaadavamaks. 1957-dal aastal loodud keel oli tähtis programmeerimiskeelte etapi osa, kuna ennemalt kirjutati koodi kas masinakoodis (esimene generatsioon keeli) või assemblerkeeles (teine generatsioon keeli), mis nõudis programmeerijat kirjutama instruktsioone kas binaarses või kuueteistkümnendik formaadis. *Fortran* lubas kiiret koodi kirjutamist, mis jooksis peaaegu sama kiiresti kui programmid, mis olid kirjutatud masinakoodis. Kuna arvutid olid haruldased ning väga kallid, siis olid ebatõhusad programmid suuremaks finantsiliseks probleemiks kui aega võtvate ja vaevarikaste masinakoodis olevate programmide arendamine. Kõrgema taseme keele, tuntud kui ka kolmanda generatsiooni keel, loomine tõi endaga kaasa programmeerimise laienemise ka insenerideni ja teadlasteni, kes olid väga poolt arvutite kasutamise laiendamisega. *Fortran*'i uuendati palju 50-datel ning 60-datel, et ta pakuks teistele keeltele konkurentsi. *Fortran*'i suuremad versioonid olid järgmised: *Fortran* 77 aastal 1978, *Fortran* 90 aastal 1991. *Fortran* sai uuendusi ka aastatel 1996, 2004, 2010 ning 2018, mis näitab, et *Fortran*'il on ikka veel palju kasutust. [9]

Fortran'i kasutasid algselt teadlased ning insenerid ja ta oli oma alal domineeriv. Viimase 30ne aasta jooksul on *Fortran*'i kasutatud sildade ja lennukite disainimiseks, tehaste automaatikas, tormide äravoolu disainis, teaduslike andmete analüüsis ja nii edasi. [10]

```
program hello
     print *, 'Hello World'
end program hello
```

Joonis 3. Hello World *Fortran*'is

2.7 ALGOL

ALGOL (*Algorithmic Language*) on programmeerimiskeel, mis loodi 1958 aastal Ameerika ja Euroopa arvutiteadlaste ühiskomitee poolt. Seda juhtis Alan J. Perlis Carnegie Mellon'i ülikoolist. *ALGOL* oli alguspunkt mõningate kõige suuremate ja tähtsamate programmeerimiskeelte arenduseks nagu näiteks *Pascal, C, C++* ning *Java*. *ALGOL*'il olid rekursiivsed alamprogrammid. Samuti tutvustas *ALGOL* plokkstruktuuri (*block structure*), kus programm koosneb plokkidest, mis võivad koosneda mõlemast, instruktsioonidest ja andmetest ning neil on sama struktuur nagu tervel programmil. Plokkstruktuur muutus võimsaks tööriistaks, et ehitada suuri programme väikestest komponentidest. *ALGOL*'i suuremad väljalasked olid *ALGOL* 60 aastal 1960 ja *ALGOL* 68 aastal 1968. *ALGOL* oli laialdaselt kasutuses Euroopas ning ta jäi paljudeks aastateks keeleks, milles arvuti algoritme avalikustati. Tänapäeval *ALGOL* enam laialdaselt kasutuses ei ole. [11]

```
BEGIN
     OUTSTRING(2, "Hello World");
END.
```

Joonis 4. Hello World *ALGOL*'is [12]

2.8 LISP

LISP (*List Processor*) on 1960 aastal John McCarthy poolt välja töötatud programmeerimiskeel. *LISP* loodi rekursiivsete funktsioonide matemaatilise teooria põhjal. *LISP*'i programm on andmetele rakendatud funktsioon, mitte nagu *FORTRAN* või *ALGOL*, mis on järjestus protseduurilisi samme. *LISP* kasutab väga lihtsat tähistust, milles on operatsioonidele ja nende operantidele antud sulgudes nimekiri. Näiteks (+ a (*b c)) tähendab a+b*c. Kuigi see näeb veider välja, töötab see järjestus arvutitel hästi. *LISP* kasutab nimekirja struktuuri ka andmete esitamiseks ning kuna programm ning andmed kasutavad sama struktuuri, siis on *LISP*'i programmil lihtne tegutseda teistes programmides andmetena. *LISP* on tuntud keel ka AI(*Artificial intelligence*) programmeerimises, osaliselt tänu *LISP*'i ja AI tööle MIT-s ja osaliselt

kuna AI programmi võimet „õppida“ saab kirjutada *LISP*’is isemodifitseeriva programmina. *LISP* on vanim keel, mida peatakse laialt kasutatuks. [13]

```
(print "Hello World".)
```

Joonis 5. Hello World *LISP*’is [12]

2.9 COBOL

COBOL (*Common Business Oriented Language*) on kõrgetasemeline ning üks esimesi laialdaselt kasutatud programmeerimiskeeli ja äri valdkonnas oli ta kõige populaarsem keel mitmeid aastaid. See arenes välja 1959 aasta Andmesüsteemide Keelte Konverentsil (*Conference on Data Languages*) USA ja erasektori ühisalgatusel. Keele arendust juhtis Dr. Grace Murray Hopper. *COBOL* loodi, et täita kahte suuremat eesmärki: teisaldatavus (võime jooksutada programme erinevata tootjate arvutitel ilma suuremate modifikatsioonidega) ning loetavus (programm oleks loetav nagu tavaline inglise keel). Kuigi *COBOL*’i populaarsus hakkas 90date alguses langema, kasutasid seda 21. sajandi alguses paljud pangad ja riigiasutused ning *COBOL*’it kasutatakse ikka veel panganduses ja selle peal jooksevad paljud krediitkaardi protsessorid, pangaautomaadid, mobiili kõned ning valgusfoorid. [14]

```
IDENTIFICATION DIVISION.
PROGRAN-ID. HelloWorld.
AUTHOR. Milo.

ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
INPUT-OUTPUT SECTION.

DATA DIVISION.
FILE SECTION.
WORKING-STORAGE SECTION.
LINKAGE SECTION.

PROCEDURE DIVISION.
DISPLAY "Hello World".
STOP RUN.
```

Joonis 6. Hello World *COBOL*'is [12]

2.10 BASIC

BASIC (*Beginner's All-Purpose Symbolic Instruction Code*) on John G. Kemeny ja Thomas E. Kurtz'i poolt 1964 aastal välja töödatud programmeerimiskeel. See on üks kergemaid kõrgetasemelisi keeli, mille käsud on inglise keelega väga sarnased. Seda võivad lihtsalt ära õppida nii koolilapsed kui ka algajad programmeerijad. Sellel olid lihtsad andmestruktuurid ja tähistused ning seda tõlgendati järgmiselt: *BASIC*'u programm tõlgiti rida rea haaval ja jooksutati nii kuidas seda oli tõlgitud, mis tegi vigade leidmise lihtsaks. *BASIC*'u väike suurus ja lihtsus muutis ta populaarseks esimetes personaal arvutites. Tema hiljutised vormid nagu näiteks *Visual Basic* on üle võtnud palju andme -ja kontrollistruktuure teistelt kaasaegsetelt keeltelt, mis on muutnud ta võimsamaks kuid algajatele ebamugavamaks. [15]

```
10 PRINT "Hello World"
```

Joonis 7. Hello World *BASIC*'us [12]

2.11 PASCAL

Pascal on 1970 aastal šveitslase Niklaus Wirth'i poolt välja töötatud ning prantslasest matemaatiku Blaise Pascal'i järgi nimetatud programmeerimiskeel, mille eesmärgiks oli õpetada struktureeritud programmeerimist, mis rõhutaks tingimuslike ja *loop*'i kontrollistruktuuride korrapärast kasutamist ilma GOTO avaldusteta. Kuigi *Pascal* sarnases *ALGOL*'iga märgistustes, pakkus ta võimalust defineerida andmetüüpe, millega sai organiseerida keerulist informatsiooni. See oli iseärasus, mis ei olnud võimalik *ALGOL*'is, *FORTRAN*'is ega ka *COBOL*'is. Kasutaja poolt defineeritud andmetüübid lasid programmeerijal ümber nimetada keerulisi andmeid, mida seejärel sai keele tõlkija enne programmi jooksutamist kontrollida, kas antud andmeid on õigesti kasutatud. 70-date lõpus ja 80-datel oli *Pascal* üks kõige laialdasemalt programmeerimisejuhendina kasutatud keel. See oli saadaval pea kõigis arvutites ja kuna see oli tuttav, selge ja turvaline keel, siis kasutati teda tootmistarkvaras ja hariduses. Seda keelt kasutas oma firma algus aastatel ka Apple *Pascal*'i kasutusmugavuse ja võimsuse tõttu. [16]

```
PROGRAM HelloWorld (OUTPUT);
BEGIN
     WRITELN('Hello World');
END.
```

Joonis 8. Hello World *PASCAL*'is [12]

2.12 Smalltalk

Smalltalk on üldotstarbeline objektorienteeritud programmeerimiskeel, mis tähendab, et selles keeles ei kasutata primitiive ega kontrollstruktuure nagu protseduurilistes keeltes ning siin suhtlevad ainult objektid saates sõnumeid ja tal on rakendused pea igas tööstusharus ja igas võimalikus domeenis. *Smalltalk* disainiti varajastel 70-ndatel ning tuli välja nime all *Smalltalk-80*, kuid sai hiljem tuntuks nime all *Smalltalk*. Selle keele loojad olid Alan Kay, Adele Goldberg, Dan Ingalls ja teised *Learning Research Group*'i liikmed Xerox PARC'is. *Smalltalk*'is ei ole andmetüüpe nagu *integer*, *boolean* või *character*, vaid kõiki neid käsitletakse objektidena. Isegi

klasse ja metaklasse käsitletakse objektidena. *Smalltalk*'is on kahte tüüpi muutujaid: näite muutujad ning ajutised muutujad. *Smalltalk* on mõjutanud palju teisi programmeerimiskeeli nagu näiteks *C, Python, Ruby, Java* ning tal on efektiivne arvutusvõimsus. See on põhiliseks põhjuseks, miks *Smalltalk*'i kasutatakse laialdaselt võrgu rakendustes, asjade internetis (IoT), meditsiini valdkonnas, tehisintellektis, masinõppes, mobiilides ja paljudel teistel tööstuslikel aladel. [17]

```
'Hello World' out.
```

Joonis 9. Hello World *Smalltalk*'is [12]

2.13 C

C on protseduuriline programmeerimiskeel, mille arendas välja Dennis Ritchie 1972 aastal. *C* keel arendati välja *B* keelest, kui Ritchie hakkas *B* keelt arendama ja nimetas oma arenduse algselt *New B*'ks (lühendatult *NB*) hiljem peale uue kompilaatori kirjutamist sai keele nimeks *C*. [18]

Algselt arendati seda keelt süsteemi programmeerimiskeelena, et kirjutada operatsioonisüsteemi. *C* põhilised tunnusjooned on juurdepääs madalatasemelise mälule(*low-level memory access*), lihtsad võtmesõnad ja puhas stiil, need tunnusjooned muudavad *C* keele sobivaks süsteemi programmeerimiseks, näiteks operatsioonisüsteemid või kompilaatorite arendus. Paljud hilisemad keeled on laenanud *C* keelelt süntaksit ja tunnusjooni. Näiteks *Java*, *PHP, JavaScript*'i ja paljude teiste keelte süntaks baseerub *C* süntaksil. [19]

```
#inculde <stdio.h>
main()
{
Printf("Hello World\n");
}
```

Joonis 10. Hello World *C*'s [12]

2.14 SQL

SQL (*Structured Query Language*) on standardiseeritud programmeerimiskeel, millega hallatakse suhtelisi andmebaase (*relational databases*) ja tehakse nendes andmebaasides olevate andmetega erinevaid operatsioone. *SQL*'i arendati algselt IBM'i teadlaste Raymond Boyce'i ja Donald Chamberlain'i poolt aastal 1972 . Algselt oli selle nimi *SEQUEL*. *SQL*'i ei kasuta aktiivselt ainult andmebaasi administraatorid, vaid ka arendajad, kes kirjutavad andmete integratsiooni skripte ja andmete analüütikud, kes tahavad jooksutada analüütilisi päringuid. [20]

```
SELECT "Hello World"
```

Joonis 11. Hello World *SQL*'is [12]

2.15 Ada

Ada programmeerimiskeelt disainis algselt varajastel 80-datel Jean Ichbiah-i juhitud tiim CUU Honywell Bull'ist Ameerika Ühendriikide Kaitseministeeriumi lepingu alusel. *Ada* on nime saanud matemaatiku Ada Lovelace'i järgi ning *Ada* on struktureeritud, staatiliselt trükitud, hädavajalik, laia spektriga ja objektorienteeritud kõrgetasemeline programmeerimiskeel. *Ada* oli teiste sel ajal populaarsemate keelte laiendus nagu näiteks *Pascal*. Mõningates riikides kasutatakse *Ada*'t õhu liikluse haldamise süsteemides nagu näiteks Austraalia, Belgia ja Saksamaa ning teiste transpordi ja kosmose projektide võõrustajana. [21]

```
with Ada.Text_IO;
use Ada.Text_IO;
procedure HelloWorld is
begin
     Ada.Text_IO.Put_Line("Hello World");
End HelloWorld;
```

Joonis 12. Hello World *Ada*'s [12]

2.16 C++

C++ on aastal 1983 Bjarne Stroustrup'i poolt välja antud C keele laiendus. Võrreldes *C*'ga on *C++*'is näiteks klassid, virtuaalsed funktsioonid ja mallid. Algselt kui Stroustrup hakkas *C*'d arendama nimetas ta oma modifitseeritud *C* keele „C with Classes"-iks. Selle modifitseerimise eesmärgiks oli lisada *C*'sse OOP (*Object Oriented Programming*) ilma *C*'d väga muutmata. *C++* on tuntud ka kui „superset of C", see tähendab, et kõiki *C* programme saab jooksutada *C++*'iga, kuid mitte vastupidi. *C++* on kombinatsioon nii madala taseme kui ka kõrge taseme ehk ta on kesk taseme keel. Kuna „++" on lühend ühe liitmisele, siis *C++* tähendab üldistatult „üks kõrgem kui *C*". *C++* on olnud 10ne kõige kasutatuma keele seas aastast 1986. *C++*'i kasutatakse näiteks MS Office's, Adobe Photoshop'is, mängude mootorites ja teistes suure jõudlusega tarkvarades. [22][23]

```
#inculde <iostream.h>
int main(int argc, char *argv[])
{
     cout << "Hello World" << endl;
     return 0;
}
```

Joonis 13. Hello World *C++*'s [12]

2.17 Objective-C

Objective-C loodi põhiliselt Brad Cox'I ning Tom Love'I poolt varajastel 1980datel nende firmas PPI(*Productivity Products International*). *Objective-C* on samuti *C* „superset" ehk kõiki *C* programme saab *Objective-C*'s jooksutada, kuid mitte vastupidi. *Objective-C* objekti süntaks

tuleneb *Smalltalk*'ist, kuid kõik ülejäänud süntaksid põhinevad *C*'l. *Objective-C* on põhiline programmeerimiskeel Apple'i operatsioonisüsteemide, macOS ning iOS, kirjutamises. [24][25]

```
#import <Foundation/Foundation.h>
int main()
{
     NSLog(@"Hello World");
     return 0;
}
```

Joonis 14. Hello World *Objective-C*'s [26]

2.18 Perl

Perl on kõrgetasemeline, üldotstarbeline programmeerimiskeel loodud Larry Wall'i poolt aastal 1987. Algselt oli *Perl* disainitud skripti keeleks teksti redakteerimiseks, kuid tänapäeval kasutatakse laialdaselt, näiteks CGI's (*Computer-generated imagery*), andmebaasi rakendustes, süsteemi administratsioonis, võrgu programmeerimises ning graafilises programmeerimises. Originaalselt soovis Larry Wall nimetada enda loodud keele "Pearl"-iks, kuid just enne oma keele välja laskmist avastas ta, et selle nimeline programmeerimiskeel juba eksisteerib. Siis ta muutis kirjapilti ja keele nimeks sai *Perl*. *Perl*'i süntaks koosneb mitmest keelest, mille hulgas on ka *C, Smalltalk, Lisp* ning isegi inglise keel. [27][28]

```
print "Hello World\n";
```

Joonis 15. Hello World *Perl*'is [12]

2.19 Haskell

Haskell on üldotstarbeline programmeerimiskeel loodud aastal 1990 ning oma nime sai ta ameerika loogiku ja matemaatiku Haskell Brooks Curry järgi. See on puhtalt funktsionaalne programmeerimiskeel, mis tähendab, et see on eelkõige matemaatiline keel. *Haskell*'i kasutatakse paljudes tööstusharudes, täpsemalt seal, kus tegeletakse keeruliste kalkulatsioonidega, dokumentatsioonidega ja arvuliste andmete töötlemisega. Nagu ka paljusid teisi keeli sellest ajast, ei leia ka *Haskell*'it tuntumatest rakendustest. [29]

```
module Main (main) where
file
main :: IO ()
main=putStrLn "Hello World"
```

Joonis 16. Hello World *Haskell*'is [30]

2.20 Python

Python on laialdaselt kasutatud, üldotstarbeline, kõrgetasemeline programmeerimiskeel. Algselt disainis keele Guido van Rossum aastal 1991 ja seda arendas edasi Python Software Foundation. Seda arendati pannes peamiselt rõhku koodi loetavusele ning *Python*'i süntaks lubab programmeerijatel mõisteid väljendada vähemate koodi ridadega. *Python* sai oma nime briti komöödia grupi „Monty Python"-i järgi. *Python* on tänase päevani üks kõige populaarsemaid programmeerimiskeeli ning seda kasutatakse näiteks Google's, Yahoo's ning Spotify's. [31]

```
print: "Hello World"
```

Joonis 17. Hello World *Python* 2.6's ja varasemates versioonides

```
print("Hello World")
```

Joonis 18. Hello World *Python* 3.0's ja hilisemates versioonides

2.21 Visual Basic

Visual Basic on Microsofti poolt 1991 aastal arendatud programmeerimiskeel. See arenes välja varasemast DOS'ist (*Disc Operating System*) nimega *BASIC*. *Visual Basic* on kasutajasõbralik keel disainitud algajatele ja see lubab kõigil luua GUI'i (*Graphical User Interface*) aknaid väga lihtsasti. *Visual Basic*'u versioonid baseeruvad *Visual Basic 1.0*'st kuni *Visual Basic 6.0*'ni. [32]

```
Imports System
Module Hello_Program
Sub Main()
     Console.WriteLine("Hello World")
     Console.WriteLine("Press any key to continue…")
     Console.ReadKey()
End Sub
End Module
```

Joonis 19. Hello World *Visual Basic*'us [33]

2.22 Ruby

Ruby on Yukihiro Matsumoto poolt aastal 1993 loodud avatud lähtekoodiga, tõlgendatud ja kõrgetasemeline programmeerimiskeel. *Ruby* on *Perl*'ist, *Ada*'st, *Lisp*'ist ja *Smalltalk*'ist mõjutatud, õpetamiseks mõeldud keel. *Ruby* erineb teistest keeltest nagu näiteks *C* või *C++* selle poolest, et *Ruby* ei suhtle otse riistvaraga. See on kirjutatakse teksti faili, siis sõelutakse tõlkija poolt ning seejärel muudetakse koodiks. Need programmid on tavaliselt olemuselt protseduurilised, mis tähendab, et neid loetakse ülevalt alla. *Ruby* sobib hästi töölaua rakenduste ehitamiseks, staatiliste veebisaitide loomiseks, andmete töötlemise teenustes ja automaatika tööriistade loomiseks. Üheks *Ruby* populaarsuse põhjuseks on *Ruby on Rails* rakendusraamistik, mis koosneb eelkirjutatud *Ruby* koodist kommunikatsiooni, failide käsitlemise, andmebaaside ühenduse ja paljude muude kohta. See eemaldab tüütu osa programmeerimisest, et kasutaja saaks

keskenduda otse lahendamist vajavale probleemile. Üks *Rail*'si võtmepunkte on DRY (*Don't Repeat Yourself*), mis on rakendusraamistiku üks efektiivsuse võtmekohti. [34]

```
"Hello World\n".display
```

Joonis 20. Hello World *Ruby*'s [12]

2.23 Java

Java on üldotstarbeline, kõrgetasemeline, robustne, objektorienteeriutud ja turvaline keel, mille lõi aastal 1995 James Gosling interaktiivse TV projekti jaoks. Enne *Java* nime saamist oli keele nimi *Oak*, aga kuna *Oak* oli juba registreeritud firma, siis pidi Gosling ja tema tiim muutma nime *Java*'ks. Sel on platvormiülene funktsionaalsus ning ta on järjekindlalt üks kõige populaarsemaid programmeerimiskeeli. *Java*'t leiab kõikjalt, alates arvutitest kuni parkimisautomaatideni. *Java* süntaks sarnaneb *C*'le ja *C++*'le, kuid *Java*'s ei ole globaalseid funktsioone ega muutujaid nagu *C++*'il, kuid *Java*'l on andmeliikmed, mida võib pidada globaalseteks muutujateks. [35][36]

```
import java.applet.*;
import java.awt.*;
public class HelloWorld extends Applet
{
public void paint(Graphics g)
     {
     g.drawstring("Hello World". ,10,10);
     }
}
```

Joonis 21. Hello World *Java*'s [12]

2.24 PHP

PHP on avatud lähtekoodiga, tõlgendatud ja objektorienteeritud skriptimiskeel, mida kasutatakse peamiselt serveripoolsetes lahendustes dünaamiliste veebilehtede loomisel. *PHP* loodi Rasmus Lerdorf'i poolt aastal 1994, kuid turule tuli 1995 aastal. Ennemalt tähendas *PHP* „Personal Home Page"-i, kuid nüüd tähendab see „Hypertext Preprocessor"-it. *PHP*'d kasutavad maailma suurimad firmad nagu näiteks Facebook, Wikipedia, Digg, WordPress ning Joomla. [37][38]

```
<?php
     Echo "Hello World\n";
?>
```

Joonis 22. Hello World *PHP*'s [12]

2.25 JavaScript

JavaScript'i loojaks oli Brendan Eich ning ta tuli välja aastal 1995. *JavaScript* on skriptimis -või programmeerimiskeel. Seda keelt kasutatakse peamiselt veebi arenduses, PDF-i dokumentides, veebibrauserites ning töölaua vidinates. *JavaScript*'i mootorid olid originaalset kasutuses ainult veebibrauserites, kuid nüüd on nad põhikomponentideks ka mõningates serverites ja mitmetes rakendustes. Peaaegu iga suurem veebisait kasutab *JavaScript*'i, näiteks Gmail, Adobe Photoshop ja Mozilla Firefox. *JavaScript*'i süntaks defineerib kahte tüüpi väärtusi: fikseeritud väärtused ning muutuvad väärtused. Fikseeritud väärtused on literaalid ning muutuvad väärtused on muutujad. [39][40][41]

```
console.log('Hello World');
```

Joonis 23. Hello World *JavaScript*'is [42]

2.26 C#

C# on Microsofti poolt aastal 2000 välja arendatud keel, mille eesmärgiks oli ühendada *C++*'i arvutusvõime *Visual Basic*'u lihtsusega. *C#* baseerub *C++*'il ning tal on palju ühiseid jooni *Java*'ga. *C#* on moderne objektorienteeritud ja tüübikindel programmeerimiskeel. *C#* laseb arendajatel ehitada palju erinevaid tüüpe turvalisi ning robustseid rakendusi, mis jooksevad .NET-is (Microsofti poolt arendatud raamistik). Kuna *C#* juured ulatuvad *C* keelte perekonda ning see keel on koheselt tuttav *C, C++, Java* ning *JavaScript*'i programmeerijatele. Keel on kasutuses peaaegu kõikides Microsofti toodetes ning on peamiselt kasutatud töölaua rakenduste arenduses. [43]

```
namespace HelloWorld
{
     class Hello {
          static void Main(string[] args)
                    {
               System.Console.WriteLine("Hello World");
                    }
                  }
}
```

Joonis 24. Hello World *C#*'is [44]

2.27 Scala

Scala on Martin Odersky poolt aastal 2003 välja arendatud keel, mis ühendab matemaatilise funktsionaalse programmeerimise ja organiseeritud objektorienteeritud programmeerimise. *Scala*'s programmeerimine põhineb *Java*'l. *Scala* ühilduvus *Java*'ga muudab ta kasulikuks Androidi arenduses. Näiteks kasutavad *Scala*'t oma rakendustes Linkedin, Twitter, Foursquare ja Netflix. [45][46]

```
object Hello {
     def main(args: Array[String]) = {
       println("Hello World")
     }
}
```

Joonis 25. Hello World *Scala*'s [47]

2.28 Groovy

Groovy'i arendas välja James Strachan ja Bob McWhirter aastal 2003. *Groovy* on tuletatud *Java*'st ja on mõeldud *Java* platvormile. See on mõlemat, staatiline ja ka dünaamiline keel koos *Python*'ile, *Ruby*'ile ja *Smalltalk*'ile sarnaste tunnusjoontega. Seda saab kasutada nii programmeerimiskeelena kui ka skriptimiskeelena. Keel parandab tootlikust, kuna see on sisutihe ja seda on lihtne õppida. Mõned tuntumad firmad, kes *Groovy*'t kasutavad on Starbucks, Transferwise ja Craftbase. [48][49][50]

```
Println 'Hello World'
```

Joonis 26. Hello World *Groovy*'s [51]

2.29 Go

Go, tuntud kui ka *Golang*, on avatud lähtekoodiga, staatiliselt trükitud programmeerimiskeel, mille on välja arendanud Google, et adresseerida probleeme, mis on tekkinud suurte tarkvara süsteemidega. See on ehitatud, et see oleks lihtne, suure jõudlusega, loetav ning efektiivne. Tänu oma lihtsusele ja modernsele struktuurile on *Go* kogunud populaarsust mõnede suurimate tehnoloogiaettevõtete seas, näiteks Google, Uber, Twitch ja Dropbox. [52][53]

```
package main
import "fmt"
func main() {
	fmt.Println("Hello World")
}
```

Joonis 27. Hello World *Go*'s [54]

2.30 Swift

Swift on Apple'i poolt aastal 2014 välja arendatud asendus *C, C++* ja *Objective-C*'le. *Swift*'i eesmärk oli olla lihtsam, kui eelnimetatud keeled ja jätta vähem ruumi vigadele. *Swift*'i kasutatakse iOS'is, macOS'is, watchOS'is ning teistes operatsioonisüsteemides rakenduste loomiseks. Peale Apple'i operatsioonisüsteemide toetab *Swift* ka Linux'it, Windows'it ja Android'i. [55][56]

```
print("Hello World")
```

Joonis 28. Hello World *Swift*'s [57]

3 Keelte võrdlus

Järgnev peatükk on koostatud teisest tööst võetud tulemuste põhjal, milles on samuti võrreldud populaarsemaid programmeerimiskeeli. See töö on leitav järnevalt lehküljelt: [58].

Kõikides keeltes on jooksutatud sama eesmärgiga programmi, milleks on “Hello World” program. Antud töös oli uuritud kui palju võttis aega kompileerimine ning kui palju mälu program kasutas. Nende tulemuste järgi on hea võrrelda, kuidas erinevate keelte kompileerimised teineteisest erinevad. Lisaks toob autor välja ka keelte mõned eelised ning puudused.

3.1 Java

Java eelised: 1) *Java* disainiti selliselt, et teda oleks lihtne kasutada, lihtne kirjutada, lihtne kompileerida ja lihtne siluda.

2) *Java*’ga saab luua standard programme ja taaskasutatavat koodi.

3) *Java*’l on võime liikuda ühest automaatsest andmetöötlussüsteemist teise.

Java puudused: 1) Tunduvalt suurem mälu kasutus kui *C*’l või *C++*’il.

2) GUI rakendused näevad välja ja tunduvad tavalised.

3) Ühe paradigma keeled.

Programmi tööaja kiirus oli 1.89 sekundit ja mälukasutus sekundis oli 6.01mb.

3.2 PHP

PHP eelised: 1) Kasutajasõbralik kasutajaliides.

2) Kiire ligipääs andmebaasile.

3) Ekstreemselt kasulikud tekstitöötlusvalikud.

PHP puudused: 1) Nõuab palju manuaalset tööd.

2) Vaja on täiendavat salvestusruumi.

3) *PHP*'l pole formaalseid vigade käsitlemise mehhanisme.

Programmi tööaja kiirus oli 27.64 sekundit ja mälukasutus sekundis oli 2.57mb.

3.3 JavaScript

JavaScript'i eelised: 1) Lihtne luua veebilehekülge.

2) Tänu teksti kokku surumisele on teksti ülekandmine kiire.

JavaScript'i puudused: 1) Veebilehe loomine võtab kaua aega.

2) *JavaScript* ei ole nii paindlik kui teised alternatiivsed veebilehtede loojad, näiteks Dreamweaver.

Programmi tööaja kiirus oli 6.52 sekundit ja mälukasutus sekundis oli 4.59mb.

3.4 Python

Python'i eelised: 1) Ulatuslikud *library*'id.

2) Parem produktiivsus.

3) Tasuta ja avatud lähtekoodiga.

Python'i puudused: 1) Kiirusepiirangud.

2) Väljatöötamata andmebaasi juurdepääsukiht.

3) Disaini piirangud.

Programmi tööaja kiirus oli 71.90 sekundit ja mälukasutus sekundis oli 2.80mb.

3.5 Ruby

Ruby eelised: 1) *Ruby on Rails* pakub fantastilisi tööriistu, mis pakuvad rohkem võimalusi vähema ajaga.

2) *Ruby* kommuun on väga huvitatud testimistest ning testimise automatiseerimisest.

3) Peaaegu iga asja jaoks on olemas midagi kasulikku.

Ruby puudused: 1) *Ruby on Rails*'i tööaeg on aeglane.

2) Aktiivsed kirjed kasutatakse *rail*'sis palju ja paljud *Ruby* "gem"-id sõltuvad sellest.

3) Võib osutada raskeks hea dokumentatsiooni leidmine.

Programmi tööaja kiirus oli 59.34 sekundit ja mälukasutus sekundis oli 3.97mb.

3.6 Perl

*Perl'*i eelised: 1) *Perl* jookseb kõigi platvormide peal ning on rohkem transporditav kui *C*.

2)"Text"-ide ja "String"-ide manipulatsioon on ökonoomne.

3) Dünaamiline mälu eraldamine on *Perl*'is väga lihtne.

*Perl'*i puudused: 1) *Perl*'ist ei ole binaarset pilti lihtne kätte saada.

2) *Perl* on tõlgendav keel, nii et ta on tunduvalt aeglasem, kui teised kompileerimis keeled, näiteks *C*.

3) Ei ole võimalik kasutada *real-time* olukordades, nagu näiteks lennusimulatsioonis.

Programmi tööaja kiirus oli 65.79 sekundit ja mälukasutus sekundis oli 6.62mb.

3.7 C

C eelised: 1) *C* keel on aluseks paljudele teisetele tundud keeltele.

2) *C*'l on võime ennast laiendada.

3) *C* keelt on lihtne õppida.

C puudused: 1) *C* ei toeta objektorienteeritud programmeerimist.

2) *C* ei paku andmete turvalisust.

3) *C* ei toeta lähtekoodi taaskasutust.

Programmi tööaja kiirus oli ainult 1.00 sekundit ja mälukasutus sekundis oli 1.17mb.

3.8 C++

C++'i eelised: 1) Teisaldatavus võimaldab arendada programme sõltumata riistvarast.

2) *C++* on objektorienteeritud manustatud keel.

3) *C++* lubab liigutada programmi arendust ühelt platvormilt teisele.

C++'i puudused: 1) Kui *C++*'i kasutatakse veebi rakenduses, siis on koodi raske siluda.

2) *C++* ei toeta "prügi kogumist".

3) *C++* on ebaturvaline.

Programmi töötamise aeg oli 1.56 sekundit ja mälukasutus sekundis oli 1.34mb.

3.9 C#

C#'i eelised: 1) Tänu .net klassi *library*'ile saaab prototüüpe kiiresti välja töötata.

2) Automaatne "prügi kogumine"

3) Tugev mälu varukoopia.

C#'i puudused: 1) *C#* on vähem paindlik kui *C++*. *C#* sõltub suuresti .NET raamistikust.

2) *C#* jookseb aeglasemalt.

3) .NET rakendus vajab aknaga platvormi et töötata.

Programmi tööaja kiirus oli 3.14 sekundit ja mälukasutus sekundis oli 2.85mb.

3.10 SQL

SQL'i eelised: 1) *SQL*'i saab kasutada, et saada andmebaasist kiiresti ja efektiivselt palju andmeid.

2) *SQL* ühendab 2 või rohkem tabelit ning näitab seda ühe objektina.

3) *SQL*'i saab kasutada, et andmeid vaadata ilma neid objekti panemata.

SQL'i puudused: 1) *SQL*'I andmebaasi liidestamine on raskem, kui mõne rea uue koodi lisamine.

2) Kui tabel kukutatakse kokku, siis muutub selle vaade mitteaktiivseks. See oleneb tabelis olevatest objektidest.

SQL'il kohta andmed puuduvad.

3.11 Swift

Swift'i eelised: 1) Tugev trükkimine.

2) Lihtsam süntaks kui *Objective-C*'l.

3) Vähem koodiridu kui *Objective-C*'l.

Swift'i puudused: 1) Vähe ressursse.

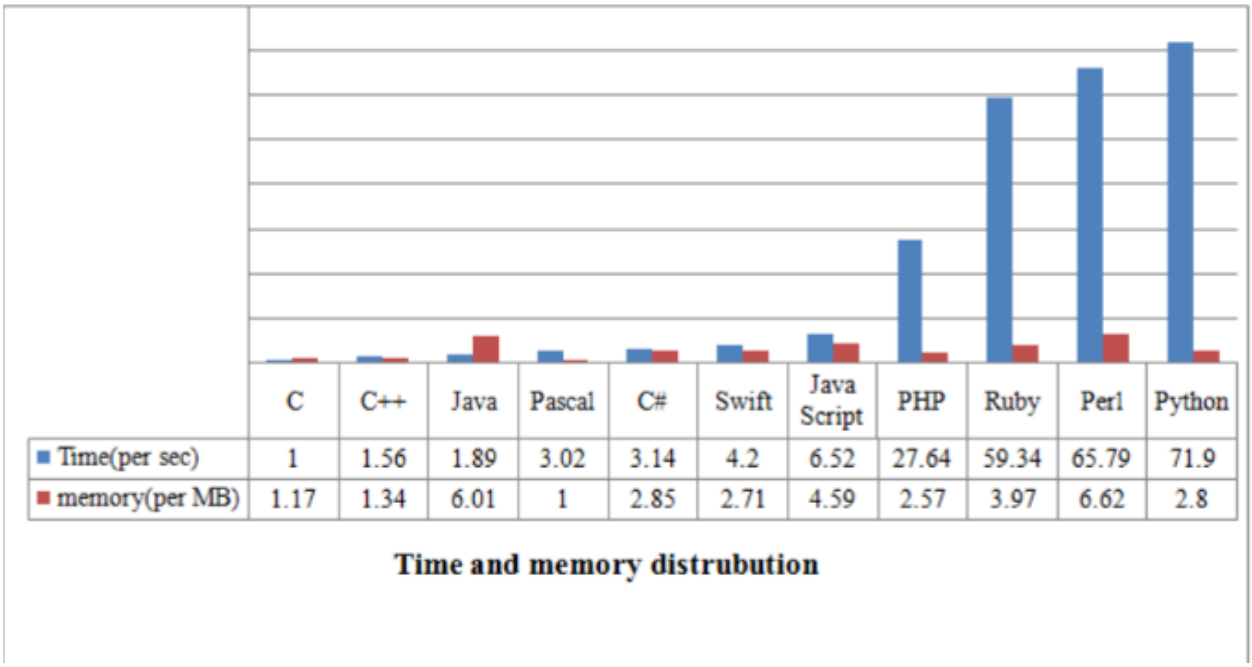
2) Arendajad peavad alustama uusi asju.

3) Muudab Apple'i rakenduskeskkonna aeglasemaks.

Programmi kompileerimise töö aeg oli 4.20 sekundit ja mälukasutus sekundis oli 2.71mb.

3.12 Testi ülevaade

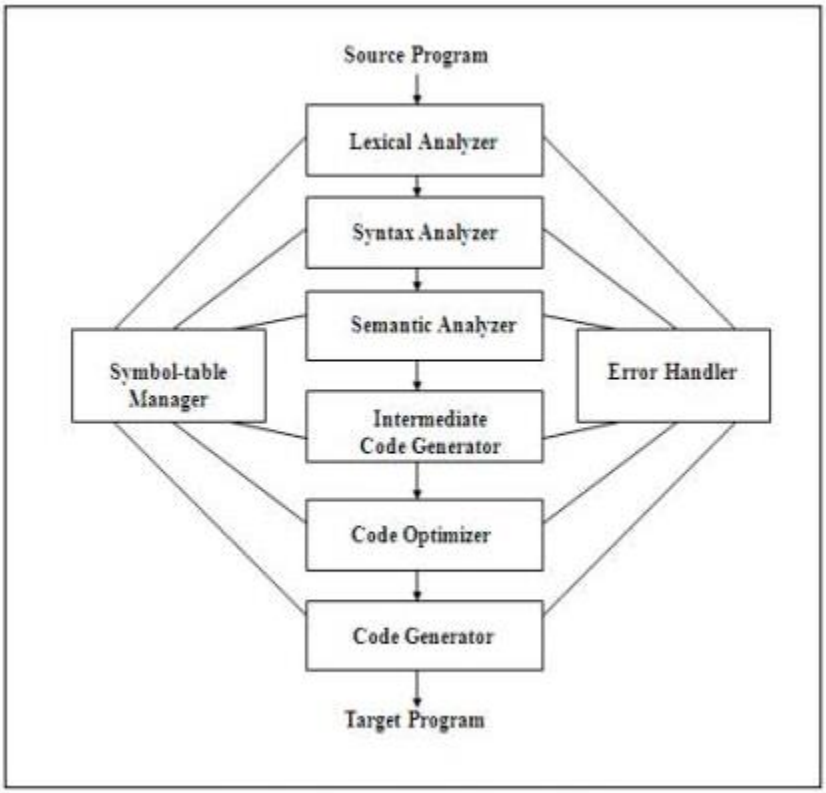
Järgnevalt on kõik saadud tulemused pandud tabelisse, et saada tulemustest parem ülevaade. Tabelis on näha keelte tööaja kiirust sekundites ning mälukasutust sekundis *MB*-des.



Joonis 29. *Time and memory distrubution*

4 Kompilaatorite ülevaade

Enne kui saab kompilaatoritest rääkima hakata, tuleb selgeks teha, mida kompilaator täpselt teeb. Tavaliselt on kompilaatori töö kõrgetasemelise keele puhul jaotatud 6-te etappi: *Lexical Analyzer* (leksikaalne analüüsija), *Syntax Analyzer* (süntaksi analüüsija), *Semantic Analyzer* (semantiline analüüsija), *Intermediate Code Generator* (kesktaseme koodi generaator), *Code Optimiser* (koodi optimeerija) ning *Target Code Generation* (sihtkoodi genereerimine). Kõikidel eelnimetatud etappidel kasutatakse ka *Symbol Tabel*'it (sümbolite tabel) ning *Error Handling*'ut (vea käitleja).



Joonis 30. *Phases of Compiler* [59]

Esimene etapp, leksikaalne analüüs, võtab algse koodi sisendiks ja muudab selle tähenduslikeks lekseemideks. Seda analüüsijat tuntakse ka skänneri nime all. Iga lekseem vastab ühele "märgile". Need "märgid" on defineeritud tavaliste väljenditega, mis on aru saadavad leksikaalsele analüüsijale. Samuti eemaldab ta ka leksikaalsed vead. [60]

Süntaksi analüüsija, mis on teiseks etapiks, on teatud ka parseri nime all. See ehitab parsimispuu. Parser võtab "märgid" ükshaaval ning kasutab *Context-Free Grammar*'it, (kontekstivaba grammatika) et ehitada parsimispuu. Programmeerimise reegleid saab esitada väheste „lavastustega". Kasutades neid „lavastusi", saame me täpselt programmi esitada. Sisendit tuleb kontrollida igal juhul, kas ta on soovitud formaadis või mitte. [60]

Kolmandaks etapiks olev semantiline analüüsija kinnitab parsimispuud, olenemata puu tähtsusest. Lisaks toodab ta kinnitatud parsimispuu. Lisaks teeb ta tüübi kontrolli, märgistuse kontrolli ning voolu kontrolli. [60]

Kesktaseme koodi generator, nagu tema nimi ütleb, genereerib kesktaseme koodi, mis on formaadis, mida saab masin kergelt jooksutada. Populaarseid kesktaseme koode on palju, näiteks *Three address codes* (Kolme aadressi kood) jne. Kesktaseme kood konverteeritakse masinakoodiks viimase kahe etapiga, mis on platvormist sõltuvad. Kuni kesktaseme koodini on kompilaatori etapid kõikidel kompilaatoritel samasugused, kuid peale seda oleneb edaspidine platvormist. Uut kompilaatorit ei ole vaja ehitada algusest peale, vaid me saame võtta juba eksisteeriva kesktaseme koodi ning ehitada viimased kaks etappi. [60]

Viiendaks etapiks on koodi optimeerija. See muudab koodi selliselt, et kood kasutaks vähem ressursse ning oleks kiirem. Koodi muutmise eesmärk ei ole koodi olemust muuta. Optimeerimist saab kategoriseerida järgmiselt: masinast sõltuv ning masinast sõltumatu. [60]

Sihtkoodi genereerimine on viimaseks ehk kuuendaks etapiks. Sihtkoodi genereerimise eesmärgiks on kirjutada kood, millest masin aru saab ning lisada *register allocation* (registri eraldamine), *instruction selection* (juhiste valik) jne. Väljund oleneb *assembler*'i (komplekteerija) tüübist. Optimeeritud kood muudetakse ümberpaigutatavaks masinakoodiks, millest saab *linker*'i ja *loader*'i (linkur ja laadur) sisend. [60]

Kõigi kuue etapiga on seotud ka sümbolite tabel ning vea käitleja.

Sümbolite tabel on andme struktuur, mida põhiliselt kasutab kompilaator ning ta koosneb kõikidest identifikaatorite nimedest koos nende tüüpidega. See tabel aitab kompilaatoril sujuvalt töötada leides identifikaatorid kiiresti üles. [60]

Vea käitleja ülesanneteks on üles leida iga viga, anda nendest kasutajale teada ning teha taastamise strateegiaid ja neid vea parandamises kasutada. Selle protsessi vältel peab programmi töötlemine olema kiire. Viga on tühi sissekanne sümbolite tabelis. [61]

4.2 C++ kompilaatorite võrdlus

Näitena toob autor välja ühe töö, kus võrreldi C++ kompilaatoreid. Kõikidel kompilaatoritel jooksutati koodi, mis on leitav järgenvalt lingilt [62].

Kõik selles peatükis kasutatud materjalid on leitavad sellelt lingilt [63],

Testitavateks kompilaatoriteks olid Intel C++, G++, Clang, Zapcc, AOCC ning PGC++.

Intel C++ on avatud lähtekoodiga kompilaator, tehtud Intel Corporation'i poolt ning ta on häälestatud Intel'i protsessorite jaoks.

G++ on samuti avatud lähtekoodiga kompilaator. G++ on tehtud Free Software Foundation'i poolt. Originaalselt kirjutati G++ GNU operatsioonisüsteemi jaoks. G++ suudab genereerida suure hulga koodi sihtarhitektuuride jaoks ning on laialdaselt saadaval Unix'i platvormidel.

Järgmised 3 kompilaatorit baseeruvad koodi genereerimisel LLVM infrastruktuuril.

Clang arendati välja LLVM'i projektina. Projekti eesmärgiks oli välja töötada G++ kollektsioonile vastav kompilaatorite komplekt, mis asendaks G++ kollektsiooni. Selles projektis kasutati LLVM 5.0.0'i, mis on LLVM'i infrastruktuuri kõige uuem versioon.

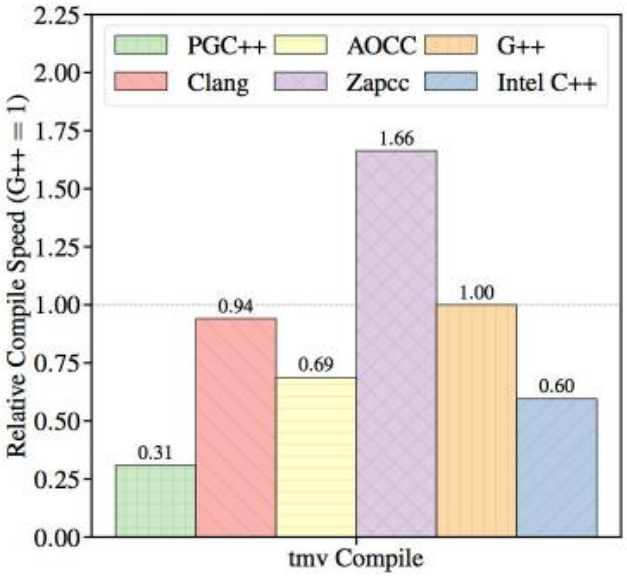
Zapcc on tehtud Ceemple Software Ltd. poolt ning on Clang'i asendus, mille eesmärk on kompileerida koodi palju kiiremini kui Clang. Zapcc kasutab samuti LLVM 5.0.0 *backend*'i optimeerimiseks, koodi genereerimiseks jne.

AOCC on AMD'le suunatud Clang 4.0.0 versioon ning ta on optimeeritud AMD 'Zen' tuuma protsessoritele. AMD täienduste seas on täiustatud vektoriseerimine, kõrgetasemeline optimeerija, kogu programmi optimeerimine ja koodi genereerimine. AMD kompilaator tuleb enda versiooni LLVM *library*'tega.

PGC++ on loodud Portland Group'i poolt ning pakub laialdast toetust viimasele OpenACC 2.5 standardile. PGC++ on saadaval kahes versioonis: tasuta Community Edition (PGC++ 17.4) ning tasuline Professional Edition (PGC++ 17.9). Selle töö kirjutamise ajal oli saadaval toetusega LLVM beeta versioon, millel oli OpenMP 4.5 laiendused testimise jaoks. Portland Group töötab sellel kallal, et lisada AVX-512 koodi generatsiooni võimed ka PGC++'sse. Antud töös on kasutatud tasuta Community Edition'it.

Testi platvormiks oli masin kahe soketiga Intel Xeon Plantinum 8168 48 füüsilise tuumaga, mis jooksid 2.7 GHz'il ja 192 GB'd RAM'i.

Antud töös testiti palju erinevaid näitajaid. Autor kasutab sellest tööst testitud kompileerimise kiirust.



Joonis 31. Kompileerimise kiirus

Joonisel 31 on näha kõigi kompilaatorite kompileerimise aegu. Mõõdupuuks on võetud G++ tulemus ning teiste tulemused on G++ järgi välja arvutatud. Kõige kiirem oli Zapcc'i aeg, mis oli G++'i ajast 1.66 korda kiirem. Kõige aeglasem oli PGC++, mille tulemuseks sai 0.31.

5 Kokkuvõte

Vaadates tagasi tööle võib autor väita, et programmeerimiskeelte maailm on väga kirju ning väga huvitav. Kindlasti on veel palju seal maailmas õppida ja uurida, kuid eesmärk saada keeltest ülevaade ja suuremaid keeli võrrelda sai täidetud.

Selle töö käigus töötas autor läbi palju erinevaid artikleid ning eelnevalt tehtuid töid, kus sai ta palju inspiratsiooni ning teadmisi, mida oma töös kasutada.

Töö kõigi kolme osa eesmärgid said täidetud, kuid autori arvates oli kolmanda osa teine pool, kus võrreldi kompilaatoreid, veidi lühikeseks ning seal oleks võinud võrrelda veel teiste keelte kompilaatoreid.

Kasutatud kirjandus

[1] „History of programming languages“ [Võrgumaterjal]. Available: https://devskiller.com/history-of-programming-languages/ [Kasutatud 16.02.2022]

[2] „The World’s 1st Computer Algorithm…“ [Võrgumaterjal]. Available: https://www.livescience.com/63154-ada-lovelace-first-algorithm-auction.html [Kasutatud 16.02.2022]

[3] „Plankalkül Programming History: History, Origin, and More“ [Võrgumaterjal]. Available: https://history-computer.com/plankalkul-guide/ [Kasutatud 17.02.2022]

[4] „Assemby Language“ [Võrgumaterjal]. Available: https://www.investopedia.com/terms/a/assembly-language.asp [Kasutatud 18.02.2022]

[5] „Assemby Language“ [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/Assembly_language [Kasutatud 18.02.2022]

[6] „“Hello World“ in x86 Assembly Language“ [Võrgumaterjal]. Available: https://montcs.bloomu.edu/Information/LowLevel/Assembly/hello-asm.html [Kasutatud 18.02.2022]

[7] „Interpreters“ [Võrgumaterjal]. Available: https://www.britannica.com/technology/computer/UNIVAC#ref723735 [Kasutatud 21.02.2022]

[8] „Autocode“ [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/Autocode [Kasutatud 21.02.2022]

[9] „Fortran“ [Võrgumaterjal]. Available: https://www.britannica.com/technology/FORTRAN [Kasutatud 23.02.2022]

[10] „Programming in FORTRAN“ [Võrgumaterjal]. Available: https://web.chem.ox.ac.uk/fortran/fortran1.html [Kasutatud 23.02.2022]

[11] „ALGOL“ [Võrgumaterjal]. Available: https://www.britannica.com/technology/ALGOL-computer-language [Kasutatud 28.02.2022]

[12] „Hello World“ [Võrgumaterjal]. Available: https://www.osdata.com/programming/basicstuff/helloworld.html [Kasutatud 28.02.2022]

[13] „LISP“ [Võrgumaterjal]. Available: https://www.britannica.com/technology/LISP-computer-language [Kasutatud 02.03.2022]

[14] „COBOL“ [Võrgumaterjal]. Available: https://www.britannica.com/technology/COBOL [Kasutatud 02.03.2022]

[15] „BASIC“ [Võrgumaterjal]. Available: https://www.britannica.com/technology/BASIC [Kasutatud 03.03.2022]

[16] „Pascal“ [Võrgumaterjal]. Available: https://www.britannica.com/technology/Pascal-computer-language [Kasutatud 07.03.2022]

[17] „Introduction to Smalltalk“ [Võrgumaterjal]. Available: https://www.geeksforgeeks.org/introduction-to-smalltalk/ [Kasutatud 07.03.2022]

[18] „A Brief History of C Programming“ [Võrgumaterjal]. Available: https://www.section.io/engineering-education/history-of-c-programming-language/ [Kasutatud 09.03.2022]

[19] „C Language Introduction“ [Võrgumaterjal]. Available: https://www.geeksforgeeks.org/c-language-set-1-introduction/ [Kasutatud 09.03.2022]

[20] „Structured Query Language (SQL)“ [Võrgumaterjal]. Available: https://www.techtarget.com/searchdatamanagement/definition/SQL [Kasutatud 09.03.2022]

[21] „Ada Overview“ [Võrgumaterjal]. Available: https://www.adaic.org/advantages/ada-overview/ [Kasutatud 10.03.2022]

[22] „History of C++“ [Võrgumaterjal]. Available: https://www.cplusplus.com/info/history/ [Kasutatud 14.03.2022]

[23] „History of C++“ [Võrgumaterjal]. Available: https://www.geeksforgeeks.org/history-of-c/ [Kasutatud 14.03.2022]

[24] „About Objective-C“ [Võrgumaterjal]. Available: https://developer.apple.com/library/archive/documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/Introduction/Introduction.html [Kasutatud 16.03.2022]

[25] „Objective-C“ [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/Objective-C#History [Kasutatud 16.03.2022]

[26] „Hello World in Objective-C“ [Võrgumaterjal]. Available: https://therenegadecoder.com/code/hello-world-in-objective-c/ [Kasutatud 16.03.2022]

[27] „History of Perl“ [Võrgumaterjal]. Available: https://yapc.tv/history-of-perl/ [Kasutatud 19.03.2022]

[28] „Perl – Syntax Overview“ [Võrgumaterjal]. Available: https://www.tutorialspoint.com/perl/perl_syntax.htm [Kasutatud 19.03.2022]

[29] „What is Haskell Programming Language?“ [Võrgumaterjal]. Available: https://www.geeksforgeeks.org/what-is-haskell-programming-language/ [Kasutatud 21.03.2022]

[30] „Haskell (programming language)“ [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/Haskell_(programming_language) [Kasutatud 21.03.2022]

[31] „History of Python“ [Võrgumaterjal]. Available: https://www.geeksforgeeks.org/history-of-python [Kasutatud 23.03.2022]

[32] „Lesson 1: Introduction to Visual Basic“ [Võrgumaterjal]. Available: https://www.vbtutor.net/lesson1.html [Kasutatud 23.03.2022]

[33] „VB.NET Hello World Program“ [Võrgumaterjal]. Available: https://www.javatpoint.com/vb-net-hello-world-program [Kasutatud 23.03.2022]

[34] „What is the Ruby programming language?“ [Võrgumaterjal]. Available: https://acloudguru.com/blog/engineering/what-is-the-ruby-programming-language [Kasutatud 28.03.2022]

[35] „Java Tutorial“ [Võrgumaterjal]. Available: https://www.javatpoint.com/java-tutorial [Kasutatud 04.04.2022]

[36] „Java syntax“ [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/Java_syntax [Kasutatud 04.04.2022]

[37] „PHP Tutorial“ [Võrgumaterjal]. Available: https://www.javatpoint.com/php-tutorial [Kasutatud 06.04.2022]

[38] „PHP“ [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/PHP [Kasutatud 06.04.2022]

[39] „What is JavaScript?“ [Võrgumaterjal]. Available: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript [Kasutatud 11.04.2022]

[40] „JavaScript“ [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/JavaScript [Kasutatud 11.04.2022]

[41] „JavaScript Syntax“ [Võrgumaterjal]. Available: https://www.w3schools.com/js/js_syntax.asp [Kasutatud 11.04.2022]

[42] „JavaScript Program To Print Hello World“ [Võrgumaterjal]. Available: https://www.programiz.com/javascript/examples/hello-world [Kasutatud 11.04.2022]

[43] „A tour of the C# language“ [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp [Kasutatud 13.04.2022]

[44] „C# Hello World – Your First C# Program“ [Võrgumaterjal]. Available: https://www.programiz.com/csharp-programming/hello-world [Kasutatud 13.04.2022]

[45] „Scala Tutorial“ [Võrgumaterjal]. Available: https://www.tutorialspoint.com/scala/index.htm [Kasutatud 13.04.2022]

[46] „Scala (programming language)“ [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/Scala_(programming_language) [Kasutatud 13.04.2022]

[47] „Scala Book Hello, World“ [Võrgumaterjal]. Available: https://docs.scala-lang.org/overviews/scala-book/hello-world-1.html [Kasutatud 13.04.2022]

[48] „Groovy Tutorial“ [Võrgumaterjal]. Available: https://www.tutorialspoint.com/groovy/index.htm [Kasutatud 16.04.2022]

[49] „A multi-faceted language for the Java platform“ [Võrgumaterjal]. Available: https://groovy-lang.org [Kasutatud 16.04.2022]

[50] „Apache Groovy“ [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/Apache_Groovy [Kasutatud 16.04.2022]

[51] „Hello World In groovy“ [Võrgumaterjal]. Available: https://riptutorial.com/groovy/example/7598/hello-world-in-groovy [Kasutatud 16.04.2022]

[52] „What is Go? Golang Programming Language Meaning Explained“ [Võrgumaterjal]. Available: https://www.freecodecamp.org/news/what-is-go-programming-language [Kasutatud 18.04.2022]

[53] „Go (programming language)“ [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/Go_(programming_language) [Kasutatud 18.04.2022]

[54] „Go by Example: Hello World“ [Võrgumaterjal]. Available: https://gobyexample.com/hello-world [Kasutatud 18.04.2022]

[55] „Learn Swift Programming“ [Võrgumaterjal]. Available: https://www.programiz.com/swift-programming [Kasutatud 20.04.2022]

[56] „Swift (programming language)“ [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/Swift_(programming_language) [Kasutatud 20.04.2022]

[57] „Swift “Hello, World!“ Program“ [Võrgumaterjal]. Available: https://www.programiz.com/swift-programming/hello-world [Kasutatud 20.04.2022]

[58] „Comparison of Programming Languages: Review“ [Võrgumaterjal]. Available: https://www.researchgate.net/publication/326672199_Comparison_of_Programming_Languages_Review [Kasutatud 25.04.2022]

[59] „The Phases of Compiler“ [Võrgumaterjal]. Available: https://www.brainkart.com/article/The-Phases-of-a-Compiler_8070/ [Kasutatud 09.05.2022]

[60] „Phases of a Compiler“ [Võrgumaterjal]. Available: https://www.geeksforgeeks.org/phases-of-a-compiler/ [Kasutatud 09.05.2022]

[61] „Error Handling in Compiler Design“ [Võrgumaterjal]. Available: https://tutorialspoint.dev/computer-science/compiler-design/error-handling-compiler-design [Kasutatud 09.05.2022]

[62] „CompilerComparisonCode“ [Võrgumaterjal]. Available: https://github.com/ColfaxResearch/CompilerComparisonCode [Kasutatud 28.03.2022]

[63] „A Performance – Based Comparison of C/C++ Compilers“ [Võrgumaterjal]. Available: https://colfaxresearch.com/compiler-comparison/ [Kasutatud 28.03.2022]

Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

Mina, Karl-Erik Kald

1 Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Programmeerimiskeelte võrdlus ja analüüs" mille juhendajad on Peeter Ellervee

   1.1 reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;

   1.2 üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

2 Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.

3 Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

10.05.2022

1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.