Anna Whitlocks Gymnasium
Teknikprogrammet inriktning design och produktutveckling
Läsåret 2021/2022
Handledare: Sofie Danielsson

En intresseväckande och relevant huvudrubrik

En informativ underrubrik

Elias Sjögreen

Sammanfattning

Skriv min abstract häääär!

Innehållsförteckning

Inledning

Datorprogrammering har gått från att vara en nisch skicklighet till något fler och fler människor lär sig, för jobb och som en hobby. Industrin och den globala efterfrågan för programmerare ökar i takt med internet och i princip all teknologi, från din mikrovågsugn till dator till det komplexa system som styr exempelvis elnätet. Detta är inte heller något lokalt fenomen, digitaliseringen är global men majoriteten av programmeringsspråken som är ett väsentligt verktyg för denna teknologiska utveckling är fortfarande begränsade utav en språkbarriär då de flesta programmeringsspråk använder sig utav Engelskan. Denna rapport undersöker möjligheten att skapa en specifikation samt prototyp utav ett programmeringsspråk med mål att vara översättningsbart mellan olika naturliga språk i utbildningsyfte.

Bakgrund

Begreppslista

Programmeringspråk eller **programspråk** är en representation i text eller visuellt utav de instruktioner som man önskar ska kompileras för att sedan köras, antingen som maskinkod, mellanrepresentation eller tolkas i en interpretator.

Maskinkod är den binära kod, bestående av olika instruktioner som en dator direkt kan tolka och genomföra.

Plattformsoberoende kod eller **mellanrepresentation** är en typ av kod som kan köras på en virtuell maskin på datorn istället för att köras direkt på datorn genom maskinkod.

Källkod eller **källprogram** är ett datorprogram representerat som det programmeringsspråk det från början var skrive i.

Program eller **datorprogram** är vilket som helst program som en dator kan tolka och genomföra. Detta innefattar maskinkod men även källkod och olika typer utav plattformsoberoende kod.

Virtuell maskin är en virtuell dator som i programmeringssyfte används för att skapa en miljö där plattformsoberoende kod eller olika mellanrepresentationer kan köras så som om dess kod var äkta maskinkod.

Interpretator eller **programtolk** är ett datorprogram som tolkar och genomför de instruktioner som beskrivs i programmet direkt utan översättning eller kompilation till maskinkod eller någon mellanrepresentation.

Kompilator är ett datorprogram som kan översätta ett program skrivet i källkod till datorprogram som går att köra representerade som maskinkod eller olika mellanrepresentationer.

Integrerad utvecklingsmiljö är ett datorprogram som innehåller ett antal olika verktyg för att arbeta med exempelvis programmering. Detta kan inkludera exempelvis en textredigerare, visuell programmeringsmiljö och en kompilator eller interpretator.

Programmeringsspråk

I grunden kan alla datorprocessorer tolka vissa instruktioner vid namn maskinkod (”maskinkod”, 2021). Allt en dator gör bygger på dessa, ofta mycket enkla instruktioner, representerade binärt i datorns minne. Maskinkoden kan vara svåra att läsa och skriva som människa. Komplexiteten för program skrivna i maskinkod snabbt ökar för större program, detta på grund av maskinkodens närhet till hårdvaran som tolkar programmet. För att tackla detta problem så har olika programmeringsspråk skapats för att göra programmeringen utav datorer mer abstrakt och därmed möjliggjort mer avancerad, effektivare och enklare programmering.

Källkod är den centrala delen i ett programmeringsspråk som representerar det som ett datorprogram gör och är skriven för hand. Vanligtvis består källkoden utav olika kodord, operationer, uttryck och värden i en text. Dessa kodord, operationer, uttryck och värden tolkas sedan på olika sätt utav programmeringsspråket för att till vara körbart av en dator som maskinkod.

När man talar om programmeringsspråk talar man ofta om olika hög- eller lågnivåspråk. Detta är ett koncept som används för att definiera hur nära källkoden är till den slutgiltiga maskinkoden. Historiskt tidiga programmeringsspråk är ofta närmare maskinkod och skulle därmed kunna kallas låga medan moderna ofta är väldigt abstraherade från den faktiska maskinkoden som körs och därmed är högnivåspråk. Fördelen med lågnivåspråk är just dess närhet till hur en dator fungerar och tänker vilket gör att det går att skriva mycket effektiv kod, ett bra exempel på ett sådant språk är assembler (”assemblerspråk”, 2021) vilket är ett samlingsnamn för språk där maskininstruktioner är översatta till ord och värden för olika hårdvara. Till skillnad från lågnivåspråk är högnivåspråk mycket enklare för utvecklaren att skriva samt förstå och är de vanligast använda (Carbonnelle, 2021; *TIOBE Index*, 2021).

En global efterfrågan för programmerare

Programmering som yrke har under de senaste hundra åren växt fram otroligt snabbt och behovet för mjukvaruutvecklare är idag mycket stort. I endast Sverige beräknas det saknas omring 70000 personer för att möta det växande behovet inom IT branshen (*IT-Kompetensbristen*, 2020) till år 2024. I en artikel från mjukvaruutvecklarföretaget Future Processing (*How Many Developers Are There In The World In 2021?*, 2021) skriver dom om den förväntade utvecklingen samt nuvarande efterfrågan för utvecklare. I år, 2021, uppskattas det finnas omkring 26,9 miljoner mjukvaruutvecklare globalt och ökas till kring 45 miljoner år 2030.

I en årlig undersökning av internet- och mjukvaruutvecklingsforumet Stack Overflow har man frågat 80000 utvecklare, studenter och andra användare om bland annat utbilding, land, erfarenhet och andra relevanta frågor (*Stack Overflow Developer Survey 2021*, 2021). I undersökningens statistik om vart undersök-

ningsdeltagarna bor är det endast kring 27,31%[1] som bor i länder där engelska är ett officielt förstaspråk. Utav dom mer än 37 olika programmeringsspråken som det deltagarna i undersökningen har använt sig av som svar i olika frågor är endast ett språk[2] ej baserat på engelskan.

Att programmeringsspråk inte vanligtvis är skrivna i användarens modersmål utan på engelska skulle kunna påverka spridningen, utbildningen och användingen utav programmeringsspråk i icke engelskspråkiga länder[3].

Block- och flödesprogrammering

I ett programmeringsspråk där översättning till andra språk prioriteras finns det ett antal olika tillvägagångssätt, alla med olika för- och nackdelar.

Blockprogrammering är bland det vanligaste av de olika typerna av översättningsbara programmeringsspråk. Dessa språk är ofta till för utbildningssyfte och fungerar på sätt att de består utav block, liknande dom i Lego. Blocken kan ha olika funktioner som till exempel värden, uttryck eller olika satser. Man bygger ett program genom att koppla ihop dessa block i en specialbyggd programmeringsmiljö. De kändaste och mest använda programmeringsspråket (*Scratch - Statistics*, 2021) implementerat på detta vis är Scratch (*Scratch - About*, 2021), originellt utvecklat av MIT senare Scratch Foundation. Detta språk utvecklades i utbildningssyfte med översättning och lättanvändning i åtanke och dess målgrupp är framförallt grundskolebarn.

Flödesprogrammeringsspråk är egentligt på många sätt lika de block baserade programmeringsspråk i att dem använder sig utav visuella block. I ett flödesbaserat språk är dock dessa block inte sekventiella utan ihopkopplade med sladdar som representerar ett värdes flöde genom programmet. Spelmotorer som Unreal (*Blueprint Visual Scripting*, 2021), Unity (*Bolt Visual Scripting | Unity*, 2021) och Godot (Godot Engine contributors, 2021) samt 3d modelleringsprogram som Blender (*Introduction — Blender Manual*, 2021) använder sig av nodbaserad programmering för ett enklare alternativ till traditionell textbaserad programmering. Detta alternativ är likt blockprogrammering i det att etiketterna och texterna på noderna är översättningsbara.

Textprogrammering

Ett annat alternativ till visuella programmeringsspråk så som de block- och flödesbaserade språken tidigare nämnda är ett mer traditionellt textbaserat språk.

[1] Inräknat i denna beräkning är USA med 18,33%, Stor Brittianien med 5,37% samt Kanada med 3,61%.

[2] Programmeringsspråket APL

[3] Det fjärde, åttånde och tionde Globala målet: god utbildning för alla, anständiga arbetsvillkor och ekonomisk tillväxt och minskad ojämlikhet. Något mer relevant nu än någonsin med tanke på den statistik som finns kring den växande IT branschen.

Det finns många textbaserade programmeringsspråk varav dom flesta är av den typen men väldigt få uppfyller kravet att vara översättningsbart. Vad som menas med ett textbaserat programmeringsspråk är ett språk varav programmen består utav vanliga tecken och bokstäver men som följer en viss grammatik och syntax, likt hur naturliga språk också följer en grammatik.

Ett tidigt exempel på ett översättningsbart programmeringsspråk är det av ALGOL 68 (Wikipedia contributors, 2021a). Detta språk standardiserades (Van Wijngaarden m.fl., 1968) och dess standard publicerades i flera olika naturliga språk. Denna standard översattes till ryska, tyska, franska, bulgariska, kinesiska och senare japanska samt en för dess använding i blindskrift. Detta ledde till att standarden antogs och accepterades av både UNESCOs organisation IFIP samt Sovjets och senare Rysslands standardorganisation.

Citrine (Mooij (Gabor), 2021) är ett programmeringsspråk där lokalisation är en av kärnfunktionerna vilket har lett till dess översättning till 111 olika naturliga språk. Språket lokaliserar nyckelord, nummer och skiljetecken. För att konvertera mellan olika språk kan användaren själv översätta programmet men inget inbyggt verktyg i programmet verkar göra detta åt användaren. Detta då alla olika naturliga språk som Citrine stödjer publiceras som separata program utan vetskapen om hur man skulle översätta ett program från ett till ett annat naturligt språk.

Scheme (*The Scheme Programming Language*, 2003) är ännu ett standardiserat programmeringsspråk med möjlighet till internationalisation. Detta är dock inte en kärnfunktion i språket utan har istället utvecklats av användare som har använt språkets flexibilitet för att skapa ett bibliotek ("metaphorm", 2021) där olika översättningar finns. Eftersom olika språk kan laddas dynamisk går Scheme program att vara flerspråkiga. Detta leder dock till den nackdelen att språket ej enkelt kan översättas då flera olika språk skulle kunna existera i samma program samt det faktum att språket inte är byggt med översättning, lokalisation eller internationalisation som en kärnfunktion.

Symbolprogrammering

Det sista typen som är relevant som ett alternativ för att skapa ett översättningsbart eller internationellt programmeringsspråk är den av vad jag väljer att kalla symbolprogrammeringsspråk. Detta då dom använder sig utav symboler istället för nyckelord vilket leder till att det inte är bundet till ett naturligt språk. Ett exempel på ett sådant språk är exempelvis APL (Iverson, 1962) men även helt vanlig matematisk notation (Helmenstine, 2019).

Ett programmeringsspråks uppbyggnad

Ett programmeringsspråk är egentligen i sig ett datorprogram vars uppgift är att översätta en viss inmatning till ett program som datorn kan köra. Detta

görs genom att dela upp uppgiften i ett antal olika delar. Det kan se olika ut för olika språk och tillvägagångssätt men generellt delar man in det i tre större delar: lexikalanalys (”lexikalanalys”, 2021), syntaxanalys (”syntaxanalys”, 2021) och till sist kompilation (”kompilator”, 2021) eller interpration (”interpretator”, 2021). Man har programmerat dessa delar i ett annat[4] programmeringsspråk vilket har skapat ett program som kan genomföra till exempel lexikalanalys, syntax analys och till sist kompilation för att sedan producera det slutgiltiga programmet.

Lexikalanalys

Syntaxanalys

Kompilation eller interpretation

Lexikalanalysen är oftast första steget i programmet och delar upp och klassificerar källkoden till en lista utav lexikala element så som nyckelord, värden, operatorer och skiljetecken. Detta följs sedan utav en syntaxanalys där dessa lexikala element analyseras utifrån en formell grammatik[5] för att producera en datastruktur som representerar programmets uppbyggnad. Till sist kan denna datastruktur producerad utav syntaxanalysen, ofta ett abstrakt eller konkret syntaxträd (”träd”, 2021) kompileras, det vill säga konverteras till maskinkod eller interpreteras, det vill säga tolka och utföra instruktionerna angivna datastrukturen.

[4] Eller samma, se Wikipedia contributors (2021b)

[5] Denna formella grammatik representeras ofta som något som kallas EBNF, kort för Extended Backus-Naur-form. Se Pattis (2015) för en definition utav notationen.

Syfte och Frågeställning

Målet med projektet är att utveckla ett prototypprogram samt specifikation utav ett programmeringsspråk. Denna prototyp skall vara översättningsbart och lättförståeligt. Syftet med detta är för att undersöka möjligheten och olika tillvägagångssätt för att skapa ett programmeringsspråk som går att översätta till olika språk och fortfarande vara enkelt att förstå.

Metod

Val av programmeringsspråkstyp

I valet av programmeringsspråkstyp finns det flera olika alternativ som på olika sätt uppnår det konstaterade målet.

Problemet som uppkommer för både flödes- och blockbaserad programmering är det att dom kräver en integrerad utvecklingsmiljö så som en hemsida för Scratch eller respektive programs skrivbordsapplikation för Unreal, Unity, Godot och Blender. Att utveckla en integrerad utvecklingsmiljö utöver ett programmeringsspråk och dess beståndsdelar skulle utgöra ytterligare problem och är egentligen utanför projektets omfattning.

Genomförande

Resultat

Diskussion och Slutsats

Källförtäckning

assemblerspråk. (2021). I *Nationalencyklopedin*. https://www.ne.se/uppslagsverk/encyklopedi/l%C3%A5ng/assemblerspr%C3%A5k

Blueprint Visual Scripting. (2021). Epic Games. https://docs.unrealengine.com/4.27/en-US/ProgrammingAndScripting/Blueprints

Bolt Visual Scripting | Unity. (2021). Unity Technologies. https://unity.com/products/unity-visual-scripting

Carbonnelle, P. (2021). *PYPL PopularitY of Programming Language index*. https://pypl.github.io/PYPL.html

Godot Engine contributors. (2021). *VisualScript*. https://docs.godotengine.org/en/stable/getting_started/scripting/visual_script/index.html

Helmenstine, A. M. (2019). Why Mathematics Is a Language. I *ThoughtCo*. https://www.thoughtco.com/why-mathematics-is-a-language-4158142

How Many Developers Are There In The World In 2021? (2021). Future Processing. https://www.future-processing.com/blog/how-many-developers-are-there-in-the-world-in-2019

interpretator. (2021). I *Nationalencyklopedin*. http://www.ne.se/uppslagsverk/encyklopedi/lång/interpretator

Introduction — Blender Manual. (2021). Blender Foundation. https://docs.blender.org/manual/en/latest/render/shader_nodes/introduction.html

IT-Kompetensbristen. (2020). IT&Telekom företagen. https://www.techsverige.se/2020/12/it-kompetensbristen/

Iverson, K. E. (1962). *A Programming Language*. John Wiley & Sons, Inc.

kompilator. (2021). I *Nationalencyklopedin*. http://www.ne.se/uppslagsverk/encyklopedi/lång/kompilator

lexikalanalys. (2021). I *Nationalencyklopedin*. http://www.ne.se/uppslagsverk/encyklopedi/lång/lexikalanalys

maskinkod. (2021). I *Nationalencyklopedin*. http://www.ne.se/uppslagsverk/encyklopedi/lång/maskinkod

"metaphorm". (2021). *international-scheme*. https://github.com/metaphorm/international-scheme

Mooij (Gabor), G. J. G. T. de. (2021). *Localized Programming Language Citrine*. https://citrine-lang.org

Pattis, R. E. (2015). *EBNF: A Notation to Describe Syntax*.

Scratch - About. (2021). Scratch Foundation. https://scratch.mit.edu/about

Scratch - Statistics. (2021). Scratch Foundation. https://scratch.mit.edu/statistics

Stack Overflow Developer Survey 2021. (2021). Stack Overflow. https://insights.stackoverflow.com/survey/2021

syntaxanalys. (2021). I *Nationalencyklopedin*. http://www.ne.se/uppslagsverk/encyklopedi/lång/syntaxanalys

The Scheme Programming Language. (2003). https://groups.csail.mit.edu/mac/projects/scheme

TIOBE Index. (2021). TIOBE Software. https://www.tiobe.com/tiobe-index/

träd. (2021). I *Nationalencyklopedin*. https://www.ne.se/uppslagsverk/encyklopedi/lång/träd-(2)

Van Wijngaarden, A., Mailloux, B. J., Peck, J., & Koster, C. H. A. (1968). Draft Report on the Algorithmic Language ALGOL 68. *ALGOL Bull.*, *Sup 26*, 1–84. https://doi.org/10.5555/1064072.1064073

Wikipedia contributors. (2021a). *ALGOL 68 — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=ALGOL_68&oldid=1053624353

Wikipedia contributors. (2021b). *Self-hosting (compilers) — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Self-hosting_(compilers)&oldid=1048801699