EAI Endorsed Transactions
on Creative Technologies
Research Article EAI.EU

Design and Implementation of Amharic-based IDE with High-Level Programming Language

Fitsum Gizachew[1,*]

[1]Wachemo University, Ethiopia, Hosanna, PoB 667

Abstract

INTRODUCTION: Programming languages enable us to communicate with our machines in a human-like way through computer code. Many programming languages have been developed; however, the majority of them do not have an Amharic development environment. This resulted in a Limited capacity of finding our solutions for our problems based on the context of where we are in the world and lack of technological innovation in the Amharic Language.

OBJECTIVES: the main goal of this study is to create an easy-to-use high-level Amharic programming language, as well as a special compiler Environment for the language.

METHODS: To develop the Amharic-based compiler development environment (IDE), we have used the compiler developments environment principles by modifying existing algorithms for the language. In this study, we conducted a new Amharic programming language environment called SABA, as well as a special Amharic interfaced compiler DAMAT.

RESULTS: We discovered that the study support those who wanted to learn programming languages in their local language, and educational institutions found it better to teach and learn. Last, the study contributes to global technological advancement.

CONCLUSION: The study illustrates the possibilities of constructing a programming language environment using native language by developing a prototype, further additional keywords, and grammar will enhance the study.

Keywords: Amharic Programming language, Damat, SABA.

Received on 12 December 2021, accepted on 01 February 2022, published on 07 February 2022

doi: 10.4108/eai.7-2-2022.173333

1. Introduction

In order to establish the theory of computation and the machine that performs it, communication between humans and computers was essential. It would be a struggle if our most powerful tool couldn't figure out what we wanted to do and how we wanted to accomplish it. The computer is both a computing device and a family member. Programming languages make it possible to turn humanity's best ideas into a multitude of practical solutions to a wide range of problems [1]. Languages such as low level and high level are used to do those tasks. However, high-level programming languages are commonly applied in the Development Environment (IDE) [2]. There are multiple high-level programming languages, which are

*Corresponding author. Email: computer.fitsum@gmail.com

developed for a different purpose [3], the majority of them are written in English.

As a result, users have to learn English to understand or write programs using these new high-level programming languages, which is a barrier to learning programming for millions of people throughout the world [4]. Studies suggest that students who learn a subject area in their mother tongue perform better than those who learn in a non-native language [4], [5]. Ethiopia has not made significant contributions to the computer world, preventing future generations from entering the profession of programming. Some countries have developed their programming language based on their native tongue. However, only a few studies like Axum Light [6] are carried out utilizing the Amharic programming language. Lack of technological advancements in own language causes a lot of issues, including a lack of trust in technology and students are enforced learn a foreign language and to learn to program [7], unable to integrate programming with our ancient, local, and early and civilizations, limited capacity of finding our own solutions for our problems based on our context.

The goal of this study is to create a programming Development environment that will make it easier for Ethiopian students to learn programs in Amharic IDE. The language we chose for our compiler development is java programming language because of its vast collection of libraries, plugins, and its easy-to-implement interface designing components [8],[9],[10]. The compiler language was created based on Saba which is a high-level Amharic programming language environment and with Damat which is a simple and a special compiler designated as an integrated development environment. The study does not cover the language's implementation, nor does it go into detail regarding the language's compiler design. It will only underline the implementation of the integrated development environment by showing a prototype of basic grammar for initial work.

The remainder of our paper is organized as follows: Section 2 deals with related compiler development work, section 3 deals with methodology for compiler development, and section 4 explore evaluation and experimental analysis of the developed compiler environment. Section 5 concluded with a discussion and conclusion

2. Related Work

Various studies were conducted with compiler development and IDE in different languages such as Arabic [11], Spanish[12], Turkish[13] and, so on. Those all languages' development environment doesn't support the Amharic language. Even though, very few studies in a programming language such as Axum Light are developed using the Amharic language. However, because it is a straight translation of JavaScript, it has significant drawbacks, such as language complexity and learning difficulties. Most children and learners are unfamiliar with the terminology used to depict various libraries and methodologies [14]. This makes it more difficult for those who are most capable of learning but have no interest in coding to enter the programming field. The researcher in [15] attempts to build an Amharic language-based grammar utilizing the expanded Backus- Naur form (EBNF). That was essentially how the parser and laxer parts of the grammar were written. They utilized ANTLR to evaluate the parser's performance. The grammar was evaluated using their example debugger, which was built in Java. However, this research excludes the language's implementation, example constructors for the programming construct. So, to address some, but not all, of the aforementioned issues, we attempt to fill the gap identified in the literature.

3. Methodology for Compiler Development

There are several programming languages, each with its own set of classifications. Just a few examples are Procedural [16], Functional [17], and Object-oriented Programming Language [18]. It must convert whatever programming language you're using into machine language for the computer to understand it. This may be done in two ways [19], [20]: first, build the program, and then interpret it. To build the program, the compiler takes the entire source program written in a high-level language and converts it to an equivalent program in machine code in one step.

3.1. Compilation process

All the tasks are carried out in the compiler development environment and are written in a variety of programming languages are among the tools used to create them [21]. The general overviews of the compiler development process are described in figure 1.

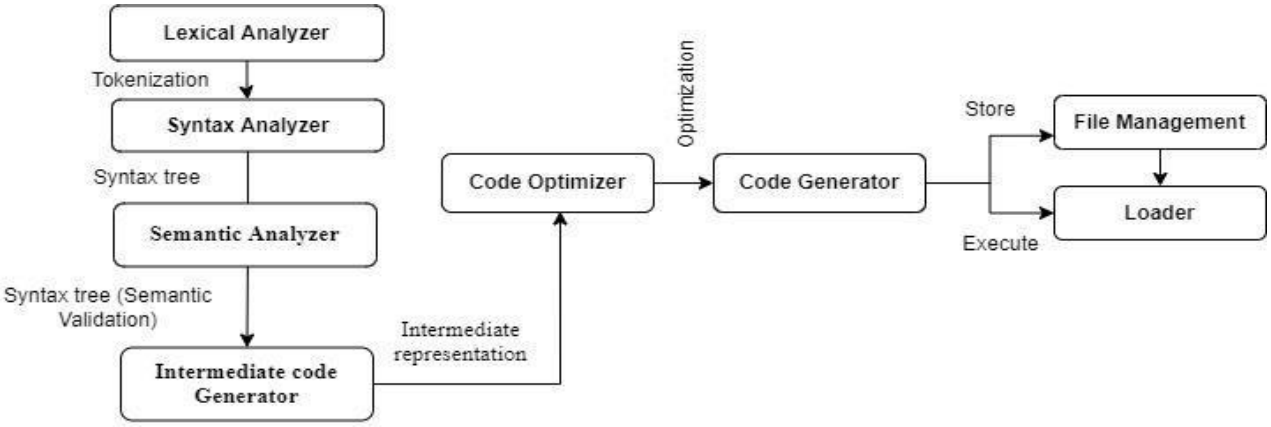

Figure 1. Compilation Process

Each of the compilers and source code editors listed above has its unique set of features, language support, and limits. The biggest issue we discovered with all of them is that none of them have an Amharic or any other local language interface, which makes it extremely difficult for children and English-speaking Ethiopians to use. Our work is confined to creating an Amharic high-level programming language Development environment with syntax and semantics, a language study guide, and basic functionality libraries and an integrated development environment, which includes features such as error detection and debugging, a Lexical Analyzer, a Syntax Analyzer, a Semantic Analyzer, a Running Environment and Tools forsaving and retrieving data.

3.2. Algorithm design

One of the things that set our research apart is that almost no existing algorithms can be applied directly. However, only a few techniques can be modified and used, such as comment scanning and detection. All of the remaining algorithms will have to be built from the ground up. For now, we'll concentrate on string processing and the variable finding approach shown in Figure 2.

```
Start
If stringFlag equals true
Begin stringProcessor on input
If input->next equals space
        Delete->current
    If input->next  equals quote
        Delete->current
    If input is all string
          Begin print(input) method
    If input->current equal new line
          Increment Newline
    If input->next equal character
          print
            End print method
    End stringProcessor
End if
If type equals string
    Set stringFlag true
    End if
```

Figure 2. Algorithm for searching variable

4. Experimental Analysis

This section discusses the grammar of the language used in the developed environment, data types, how declarations and statements are used.

Language grammars in use: developing the Amharic programming language involves creating a grammar for the language's working environment. Many design decisions were made to ensure that the grammar of the language is comprehensive and easy to use. A collection of often used grammar and their definitions follows in Figure 3.

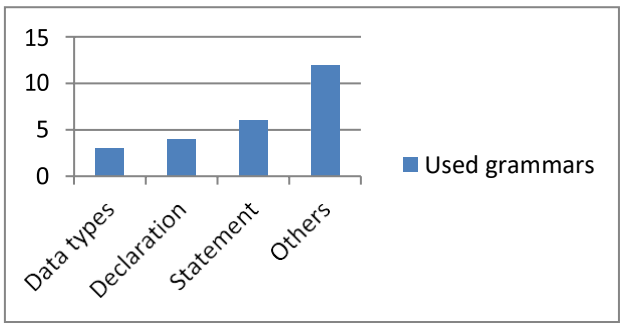

Figure 3. Number of Grammar used

Data types: There are two sorts of number systems in Amharic: Hindu-Arabic and Geez numerals; we used the Hindu-Arabic system, which begins with 0-9. This is because Geez has a numerical range that does not begin with zero. It is not used as a numeral for instruction outside of religious schools.

Table 1. Amharic Based Data Types

Data type (ሞይብ)	Represented keyword
Integers	ቁጥር
floating	ነጥብ
Boolean	እዉነታ

Declaration: It illustrates how declarations are organized in the way they are. Declarative is written in such a way that they exhibit a striking resemblance to higher-level languages while being easy enough for a beginner to understand, as the language serves as a bridge between them. The below table shows declaration were the four declarations we utilized. Table 2 shows how the declaration of Amharic language for the development of IDE is used in the study.

Table 2. Statement used in the study

Types of declaration	English keyword	Amharic keywords
Class Declaration	Class <class name> <block>	ስብስብ <የስብስቡ ስም> <ሌሎች>
Constructor Declaration	Class example (int x)	ስብስብ ምስሌ ምሳሌ(ቁጥር ሀ)
Method Declaration	<type> <name> () <block	<አይነት> <ስም>() <ሌሎች>
Variable Declaration	String name = “Fitsum”	ሐረግስም = “ፍጹም”

Statements: Assignment statements, selection statements, repetition statements, control statements, and input/output statements are all examples of statements used within a program which are described below in table

Table 3. statement used in the study

Types of statement	English keyword	Amharic keywords
Import statement	import <file path>	ተጠቀም <የፋይል አቅጣጫ>
Assignment statement	variableName = expression	ስም = “ፍጹም”
Selection statement	If, ifelse, else	‘ከሆነ’, ‘ካልሆነግን’ and ‘ካልሆነ’
Repetition statement	(sum>0) while <statment>	(ድምር > 0) እስከሆነ <ዓረፍተነገር>
Input/output statement	cout <<	አዉጣ<>

5. Prototype Development

In our compiler's prototype initially, the programmer used the compiler user interface to create a project and then requested that the compiler responds with project details options and a request for a director. A programmer selects a director and specifies the project's file name and format. **Creating a new project:** below figure 4 shows how the project is created to run a project.

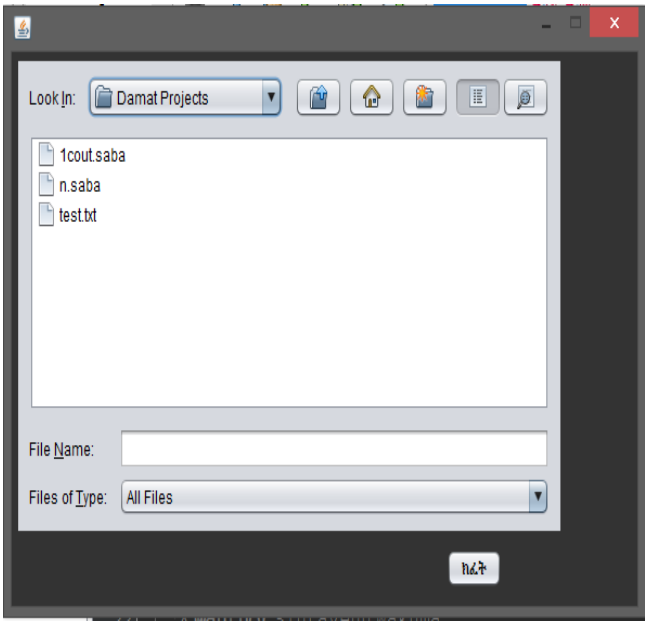

Figure 4. Creating New project

- **Run project file sample code:** this example shows simple input and output by a variable declaration that are shown below in figure 5.

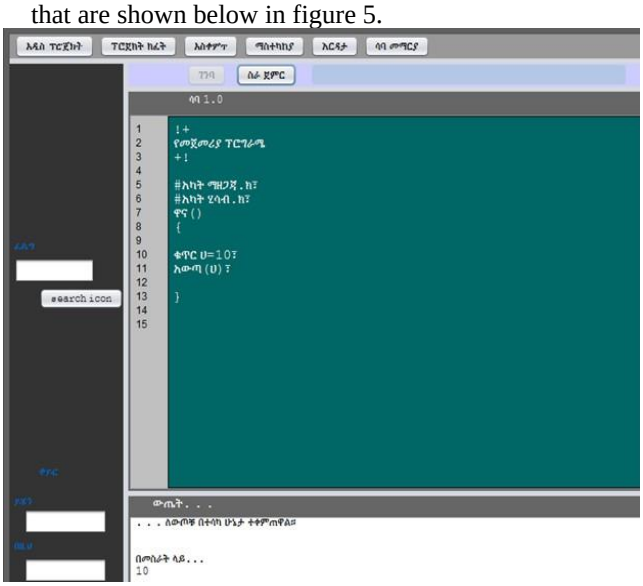

Figure 5. Displaying generated Output

- **Compilation error:** when a program is loaded and an error is generated during the compilation process. This error is shown in below figure 6.

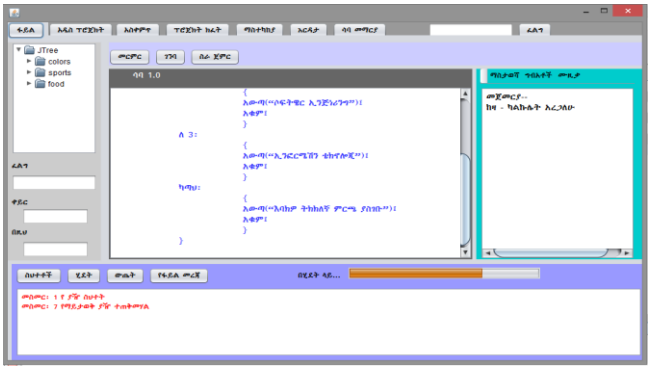

Figure 6. Compilation Error

The solution for dealing with compilation mistakes is to create a compiler that can handle mistakes that arise during compilation. Compilation problems occur when the source code does not conform to the language's keywords, and other factors might cause compile-time errors.

6. Conclusion

The study's outcome is the successful completion of Damat, an Amharic-based compiler environment. This research examines the blessing and drawbacks of learning programming in local languages, as well as whether the designed language and integrated development environment (IDE) meet their objectives. The study looked at students who were studying programming as a second language or as a base programming language to see if they could learn it more easily and quickly. The study just focuses on and shows the capability of constructing a programming language environment utilizing its language, the researcher might improve this work by adding more new keywords for later research.

References

[1] O. Iskrenovic-Momcilovic, “Learning a programming language,” Int. J. Electr. Eng. Educ., vol. 55, no. 4, pp. 324–333, 2018, doi: 10.1177/0020720918773975.

[2] J. Gosselin et al., “Extending FreeCompilerCamp . org as an Online Self-Learning Platform for Compiler Development,” pp. 43–52, 2020, doi: 10.1109/EduHPC51895.2020.00011.

[3] M. Hastings, B. Hemenway, D. Noble, and S. Zdancewic, “SoK: General purpose compilers for secure multi-party computation,” Proc. - IEEE Symp. Secur. Priv., vol. 2019-May, pp. 1220–1237, 2019, doi: 10.1109/SP.2019.00028.

[4] C. S. Prat, T. M. Madhyastha, M. J. Mottarella, and C. H. Kuo, “Relating Natural Language Aptitude to Individual Differences in Learning Programming Languages,” Sci. Rep., vol. 10, no. 1, pp. 1–10, 2020, doi: 10.1038/s41598-020-60661-8.

[5] S. Dasgupta and B. M. Hill, “Learning to code in localized programming languages,” L@S 2017 - Proc. 4th ACM Conf. Learn. Scale, pp. 33–39, 2017, doi: 10.1145/3051457.3051464.
[6] “BunnaScript.”
[7] G. Begoña, “Digital Games in Education : The Design of Games-Based Learning Environments,” J. Res. Technol. Educ., vol. 40, no. 1, pp. 23–38, 2007.
[8] L. Prechelt, “Empirical comparison of seven programming languages,” Computer (Long. Beach. Calif)., vol. 33, no. 10, pp. 23–29, 2000, doi: 10.1109/2.876288.
[9] N. Krebs and L. Schmitz, “Jaccie : A Java-based compiler – compiler for generating , visualizing and debugging compiler components,” Sci. Comput. Program., vol. 79, pp. 101–115, 2014, doi: 10.1016/j.scico.2012.03.001.
[10] P. T. Group, “The JastAdd Extensible Java Compiler,” pp. 1–17.
[11] M. Al-A’Ali and M. Hamid, “Design of an arabic programming language (ARABLAN),” Comput. Lang., vol. 21, no. 3–4, pp. 191–201, 1995, doi: 10.1016/0096-0551(95)00006-2.
[12] A. M. Zegiestowsky, “Tango : A Spanish-Based Programming Language Tango,” vol. 3, 2017.
[13] O. BİNGÖL, E. U. Küçüksille, and İ. Kuru, “Chameleon Turkish Programming Language,” Eur. J. Sci. Technol., no. December, pp. 77–82, 2018, doi: 10.31590/ejosat.442334.
[14] T. Matsumoto, Y. Watanobe, and K. Nakamura, “A model with iterative trials for correcting logic errors in source code,” Appl. Sci., vol. 11, no. 11, 2021, DOI: 10.3390/app11114755.
[15] M. Kuliah and M. Kuliah, “No 主観的健康感を中心とした在宅高齢者における 健康関連指標に関する共分散構造分析Title,” no. April, pp. 33–35, 2019.
[16] H. Jordan, G. Botterweck, J. Noll, A. Butterfield, and R. Collier, “A feature model of actor, agent, functional, object, and procedural programming languages,” Sci. Comput. Program., vol. 98, no. P2, pp. 120–139, 2015, doi: 10.1016/j.scico.2014.02.009.
[17] M. Priestley, “AI and the Origins of the Functional Programming Language Style,” Minds Mach., vol. 27, no. 3, pp. 449–472, 2017, DOI: 10.1007/s11023-017-9432-7.
[18] X. D. Zhu, “Teaching adaptability of object-oriented programming language curriculum,” Int. Educ. Stud., vol. 5, no. 4, pp. 237–242, 2012, DOI: 10.5539/ies.v5n4p237.
[19] T. Mogensen, Basics of Compiler Design. 2009.
[20] R. Morgan, “Building an Optimizing Compiler,” Analysis, p. 472, 1998.
[21] “[2008] Formal verification of a realistic compiler.pdf.”