International Journal of Advances in Applied Sciences (IJAAS)
Vol. 10, No. 2, June 2021, pp. 171~177
ISSN: 2252-8814, DOI: 10.11591/ijaas.v10.i2.pp171-177

Design and implementation of an African native language-based programming language

Ezekiel K. Olatunji[1], John. B. Oladosu[2], Odetunji A. Odejobi[3], Stephen O. Olabiyisi[4]
[1]Department of Computer Science, Bowen University, Iwo, Osun State, Nigeria
[2,4]Department of Computer Science and Engineering, Ogbomoso, Nigeria
[3]Department of Computer Science and Engineering, Obafemi Awolowo University, Ile-ife, Nigeria

Article Info

Article history:

Received Sep 10, 2020
Revised Jan 29, 2021
Accepted Feb 22, 2021

Keywords:

BNF
Lexicons
Parser
Programming languages
Yoruba language

ABSTRACT

Most of the existing high level programming languages have hitherto borrowed their lexical items from human languages including European and Asian languages. However, there is paucity of research information on programming languages developed with the lexicons of an African indigenous language. This research explored the design and implementation of an African indigenous language-based programming language using Yoruba as case study. Yoruba is the first language of over 30 million people in the south-west of Nigeria, Africa; and is spoken by over one hundred million people world-wide. It is hoped, as established by research studies, that making computer programming possible in one's mother tongue will enhance computer-based problem-solving processes by indigenous learners and teachers. The alphabets and reserved words of the programming language were respectively formed from the basic Yoruba alphabets and standard Yoruba words. The lexical items and syntactic structures of the programming language were designed with appropriate regular expressions and context-free grammars, using Backus-Naur Form (BNF) notations. A prototype implementation of the programming language was carried out as a source-to-source, 5-pass compiler. QBasic within QB64 IDE was the implementation language. The results from implementation showed functional correctness and effectiveness of the developed programming language. Thus lexical items of a programming language need not be borrowed exclusively from European and Asian languages, they can and should be borrowed from most African native languages. Furthermore, the developed native language programming language can be used to introduce computer programming to indigenous pupils of primary and junior secondary schools.

Corresponding Author:

Ezekiel K. Olatunji
Department of Computer Science
Bowen University, Iwo
Osun State, Nigeria
Email: aekolatunji@gmail.com

1. INTRODUCTION

Programming languages (PLs) can be described as notational systems for communication of algorithms and data structures to computers and people. A number of factors have influenced the evolution and development of PLs. These include but not limited to discovery of weaknesses and/or deficiencies in the existing PLs; development in computer hardware; requirements of new areas of application; changing

understanding of better methods of writing and maintaining large and complex programs; understanding of strength and weakness of some language features and the need for standardization [1], [2]. Many popular high-level languages (HLL) (such as COBOL, Pascal, Visual Basic, C++, Java and others) have been developed from the lexicons of English language while others (such as Rapira [3] and Ezhil [4] have been developed from the lexicons of Asian languages. However, there is scarcity of research information on serious efforts at designing and implementing PLs based on the lexicons of African indigenous languages, such as Yoruba language. According to [5], development of such PLs will improve computer-based problem-solving processes by indigenous teachers and learners. This fact has also been corroborated by research studies [6]-[9]. This research explored the design and implementation of an African native language-based programming language (NLPL) using Yoruba as case study. Yoruba is the first language of over 30 million people in the south-west of Nigeria, Africa; and is spoken by over one hundred million people world-wide [10].

Aside improving computer-based problem solving processes by indigenous teachers and learners, availability of NLPL will enable many millions in Africa who are only literate in their mother tongue to learn how to program the computer in their mother tongue. Over 13 million of this category of people are in Nigeria alone [2]. In addition, development and adoption of NLPL will increase the functional load of their base languages; this will consequently reduce the chances of such languages going into extinction as predicted and feared in many circles [11]-[13]. Interestingly, in a related work, in a bid to ascertain the relevance and needfulness of NLPL, Olatunji *et al* [14] carried out a needs assessment survey for a NLPL through design and administration of structured questionnaire. It was reported that eighty-nine percent (89%) of the respondents to the questionnaire expressed eagerness and willingness to program or learn programming in a NLPL, if one exists. The outcome of the study further gave impetus to embarking on this research.

2. RELATED WORKS

The dearth of literatures on African native language programming languages has earlier on been stated. The few available non-English-based PLs were developed primarily for pedagogical reasons; which is also the primary motivation for embarking on this research. Rapira [3] is an educational procedural programming language developed in the USSR and implemented on Agat computer, PDP-11 clones (Electronika, DVK, BK series) and Intel-8080/Z80 clones. It was developed by Andrev Petrovych Ershov, a Soviet Computer Scientist and notable pioneer in systems programming and programming language research [15]. Rapira was an interpreted language with dynamic type system and high level constructions. The language originally had a Russian-based set of keywords [3], but English and Moldovan were added later. Rapira was used in teaching computer programming in Soviet schools.

Ezhil is an interpreted Tamil-based programming language developed by [16]. Tamil is an Indian language spoken by over 60 million people [17]. The programming language is targeted towards the K-12 (Junior high school) level Tamil speaking students as an early introduction to thinking like a computer scientist [4]. The syntax of Ezhil is broadly similar to that of conventional BASIC. According to the author of Ezhil, the primary motivation for developing the language is that "like mathematics, computing is a concept and can be introduced through any native language". Thus by introducing computing in the native language children can easily learn how to think in the required "logical modes (enumeration, recursion, procedural)" [16]. The syntax of our programming language is similar to structured BASIC, and only Yoruba keywords are allowed.

Qalb, developed by [18], is a functional PL based on the Arabic language. This programming language enables one to write computer programs completely in the Arabic language. One of the motives for developing the programming language is to challenge the culture in which the design of most modern popular programming languages is predominantly based on the English language words. Such a culture, as noted by [19], makes learning programming especially difficult for students whose native language does not even use the Latin alphabets as does English language and for which English-based programming language keywords are little more than abstract symbols. Qalb is a programming language that deviates almost entirely from the use of ASCII character set for its encoding [18]. While Qalb's syntax is similar to that of Lisp and Scheme, the syntax of our programming language is similar to that of structured BASIC. Furthermore, unlike Qalb, our PL made use of a mixture of ASCII and Unicode characters.

The Hindawi programming system (HPS) was described by [20] as a suite that allows users to program in Indic languages (Hindi, Bangla, Gujarati, Assamese and some other Indic languages). The HPS, developed by Chaubary A and Chaudbary S., is a free, open-source, completely non-English-based programming platform that allows non-English medium literates of India to learn and write computer programs [21]. The HPS removes the English language barrier and enables non-English literates Indians to

take up computer science and participates in the information and communications technology (ICT) at all levels of technology from primary school education to robotics and super-computing in their mother tongue. While their work provided a programming platform for many Indic languages, our just one African language the Yoruba language. However, the planned implementation approach for our work is similar to that of [21].

The development of a programming language called "Dolittle" was described by [22]. Its keywords are based on Japanese language. Dolitlle was developed in 2000 [23] in response to the need for an object-oriented programming language that is suitable for children in both elementary and secondary schools in Japan. It was designed to be an object-oriented educational programming language. The programming language was written in Java and does not require elaborate declarations. Unlike the work of [22], the keywords and identifiers in this research are based on Yoruba language. Furthermore, structured programming is the focus of our designed language.

3. METHODOLOGY

This research derives its theoretical underpinning from the theories of formal grammars, languages and automata. The detail design of a subset of the programming language with lexicons of Yoruba language has been carried out [5]. This was accomplished by forming the character set and reserved words of the PL from the basic Yoruba alphabets and standard Yoruba words respectively. Necessary regular expressions and regular grammar were also defined to specify the lexical structure of the language. Yoruba words (such as Sesiro, Gbawole and others) that will not lead to ambiguity are used for the design of the lexical items of the programming language in order to simplify its implementation. Furthermore, appropriate context-free grammars were defined, using Backus-Naur Form (BNF) notations and its extended version, to specify the syntactic structure of a valid program and program elements (statements, variables, expressions and so on) in the language.

A prototype implementation of the Yoruba-based PL has been carried out. Only limited constructs, including input and output (I/O) and arithmetic statements, were implemented. The Yoruba-based PL was implemented as a source – to- source compiler that produces another high – level language, QBASIC language, as output because a compiler already exists for QBASIC. This approach has been used for a number of programming languages, such as Java, Python, and others [24]. The programming language was implemented in four phases, viz: lexical analysis, syntax analysis, semantic analysis and code generation. Recursive descent parsing algorithm was employed in implementing the syntax analysis phase. Rather than generate machine language codes, the code generation phase generated QBASIC code for constructs of the language that are syntactically and semantically valid.

Furthermore, the language was implemented as a 5-pass compiler. Each phase of the implementation formed a pass with the exception of the parser that is made up of two passes. All the components of the system were coded and test-run on the QBasic integrated development environment (IDE) using QB64 as the implementation language. QB64 is a BASIC compatible editor and C++ compiler that creates working executable files from QBasic BAS files that can be run on 32 or 64 bits PC Windows (XP, Vista and newer), Linux or Mac.

4. RESULTS AND DISCUSSION

The output of the design of a subset of the Yoruba-based PL is the syntactic definition of the PL which has been reported in [5]. Furthermore, detail sample output generated by the scanner for some given source programs in the PL has been reported in [2] and [5]. Figure 1 is a syntax error-free source program used to test the parser component of the compiler. Figure 2 and Figure 3 are the outputs generated by the parser at the end of syntax analysis of the source program. They provide information for the programmer as to the syntactic correctness or otherwise of each source statements in the source program. Figure 2 is a run-time display of the parser that gave a summary information on whether or not the source program contains syntax error, while Figure 3 provides detail status of each statements in the source program. It also provides insight into the possible causes of mistakes for statements that are in error. Examination of these outputs (Figure 2 and Figure 3) and comparing them with the input (Figure 1) show that the parser is working correctly

A source program (like Figure 1) that has been parsed to be syntax error-free by the parser would be subjected to semantic checks. Though Figure 1 has no syntax error, it however, contains some semantic errors by the definition of the programming language. This is indicated in Figure 4. These errors are type incompatibility in line 4 of Figure 1 and undeclared identifier 'Kola' in lines 5 and 6 of the source program. By the definition of the developed Yoruba-based PL, both the parser and semantic checker are functioning correctly. Figure 5 is another source program that has been compiled to be free of all compilation errors

(lexical syntax and semantics). Object code is only generated for a compilation error-free source program. The 'object' code generated in QBASIC for the source program is shown in Figure 6. QBASIC code is the 'object' code produced because the compiler was developed as a source-to-source compiler. When Figure 6 was opened within the QBASIC 64 IDE, it was discovered to be compilation error-free also. This also establishes the functioning correctness of the code generator.

```
Ibere
! Data lati dan awon pirogiramu inu konpaila wo
Intija A, B, Tade, Y
Gbawole A, B
Jeki Kola = 50
Sesiro Tade = A * B + 2 * Kola
Sesiro Y = (Tade- Kola) / 2
Kojade 'Tade je =' , Tade
Kojade 'Y1 je ='
Kojade Y
Opari
```

Figure 1. Yoruba source program parser

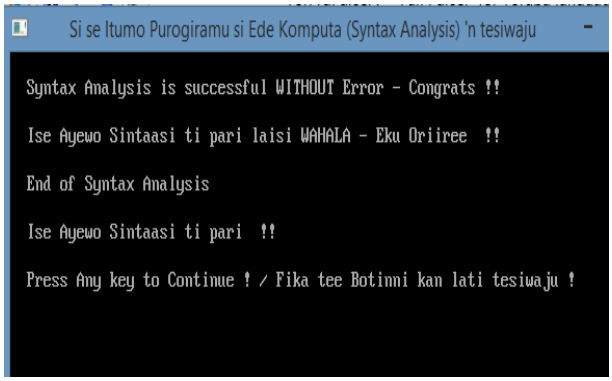

Figure 2. Run-time display of the parser

```
Abayori Sise Ayewo Sintaasi Ninu Pirogiramu Yi - 07-30-2017
==========================================
Ila  1        - Dara
Ila  2        - Dara
Ila  3        - Dara
Ila  4        - Dara
Ila  5        - Dara
Ila  6        - Dara
Ila  7        - Dara
Ila  8        - Dara
Ila  9        - Dara
Ila  10       - Dara
Ise Ayewo Sintaasi ti pari laisi WAHALA - Eku Oriiree  !!
Ise Ayewo Sintaasi ti pari  !!
```

Figure 3. Output of the parser for the source program in Figure 1

```
 AYORISI WIWO ITUMO AWON OORO INU PUROGIRAAMU  -  08-02-2017
==========================================
=
  Asise nla wa ni ila    4  - Awon Nonba yi tabi Oruko kan
ko bara mu
  Asise wa ni ila  :  5   - Ajeji oruko ni  - 'Kola' je
  Asise wa ni ila  :  6   - Ajeji oruko ni  - 'Kola' je
 Ajeji Oruko ni  'Kola' je

   Wiwo Itumo awon ooro inu Purogiraamu yi ti pari, Sugbon
ASISE die wa !!
```

Figure 4. Output of the semantic checker for the programmer (using Figure 1 as input)

```
Ibere
! Pirogiramu lati se Isiro Aropo ati Ilopo
Karakita K1
Intija A, B, Aropo, Ilopo
Kojade
Kojade '     Eku abo si akoko Isiro yi?'
Kojade
Kojade '     Jowo, Fun mi ni nanba Ikini'
Kojade
Gbawole A
Kojade '     Jowo, Fun mi ni nanba Ikeji'
Kojade
Gbawole B
Sesiro Aropo = A+B
Sesiro Ilopo = A*B
Kojade
Kojade '  Aropo',A, 'ati',B,'je: ',Aropo
Kojade
Kojade '  Aropo',A, 'ni ona ',B,'je: ',Ilopo
Kojade
Kojade '     Se e gbadun Isiro yi?'
Kojade
Kojade '     Fika te Karakita kan lati tesiwaju'
Gbawole K1
Kojade

Kojade '     Odaa boo !!!'
Opari
```

Figure 5. Sample Yoruba-based source program to do arithmetic

When Figure 6 was then run with 12 and 23 as input data, Figure 7 was produced as the run-time output. Examination of the source program of Figure 5 and the input supplied shows that the output produced (Figure 7) is correct because 12 plus 23 equals 35 and 12 multiplied by 23 equals 276.

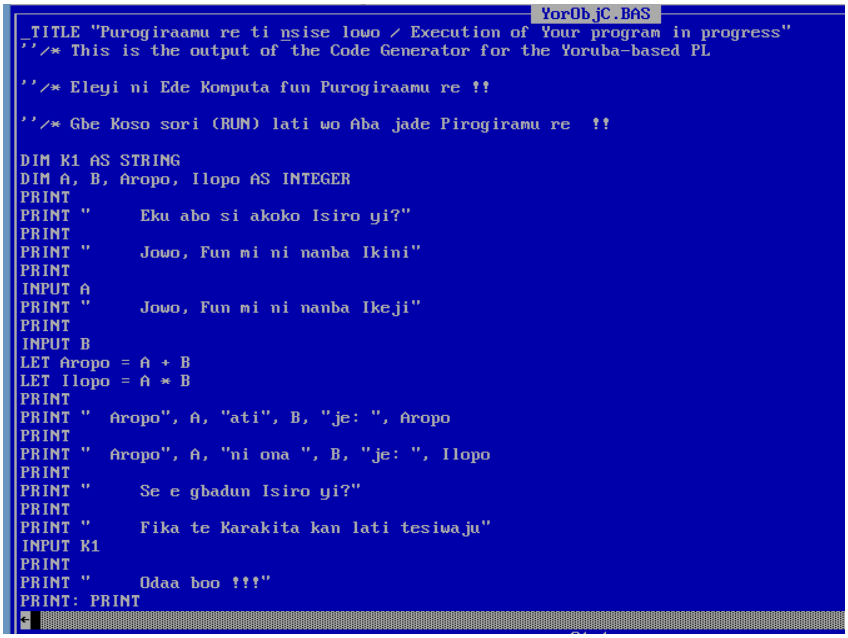

Figure 6. Code generated for the Yoruba source program of Figure 5

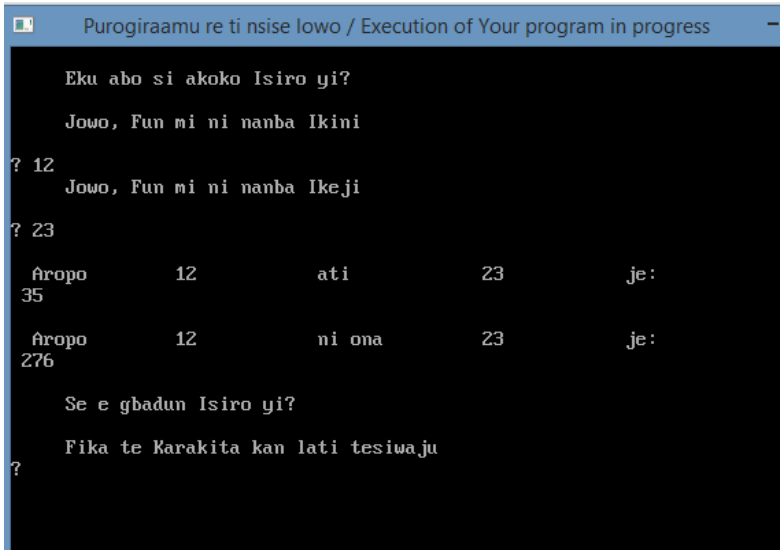

Figure 7. Run-time output of the source program of Figure

5. CONCLUSION AND FUTURE WORK

Development of a programming language based on the lexicons of an African language, like Yoruba, is very much desirable and relevant, especially in Nigeria. This is attested to by the outcome of the needs survey carried out in which eighty-nine (89%) percent of respondents to the questionnaire used for needs assessment indicated their willingness to program or learn programming in their mother tongue, if one exists [14]. The successful development of the Yoruba-based programming language shows that lexical items of programming languages need not be borrowed exclusively from European and Asian languages, but can also be borrowed from most African indigenous languages. The people who are only literate in their mother

tongues, like Yoruba, can now be taught the art of computer programming in their native languages. In particular, computer programming, using the developed programming language, can now be introduced to primary and junior secondary school pupils instead of or in addition to using BASIC or LOGO language as is the current practice in Nigeria. For future research work, the current orthography of Yoruba language would be considered for developing a Yoruba-based PL. In this regard, some special Yoruba alphabets such as á è, é, í, ó, ù, ẹ, ọ and others would be included for implementation.

ACKNOWLEDGEMENTS

The authors are grateful to their families for their support and encouragement. The editors and reviewers are also gratefully acknowledged.

REFERENCES

[1] E. K. Olatunji, "Programming Languages–Introductory Text on Concepts and Principles," *Igbagbogbemi Publishers*, Ilorin–Nigeria, ISBN 978-187-458-9, pp 8, 2014.
[2] E. K. Olatunji, J. B. Oladosu, O. A. Odejobi, S. O. Olabiyisi, "Towards Development of an Indigenous African Language-based Programmining Language," *FUOYE Journal of Engineering and Technology*, vol. 3, no. 2, pp. 61-64, 2018.
[3] B. S. Dadhich, "Needed A good 'Hindawi Programming Language," 2006. Retrieved on 15-06-2015.
[4] M. Annamalia, "Invitation to Ezhil: A Tamil Programming Language for Early Computer Science Education," *arXiv*, 2013.
[5] E. K. Olatunji, J. B. Odejobi, O. Adejobi Stephen, O. Olabiyisi, "Design of an African Native Language-based Programming Language," *University of Ibadan Journal of Science and Logics in ICT Research*, vol. 3, no. 1, pp 72-78, 2019.
[6] UNESCO, "Mother Tongue-based Literacy Programmes: Case Studies of Good Practice in Asia. Bangkok," UNESCO Bangkok, Thailand, 2007.
[7] Jane Laguna, "Improving Learning Outcomes through Mother Tongue-Based Education Improving Learning Outcomes through Mother Tongue-Based Education" *Academia*, 2013.
[8] UNICEF, "Action research on mother tongue-based bilingual education: Achieving quality, equitable education," 2014. [Online] Available: https://docplayer.net/21497094-Action-research-on-mother-tongue-based-bilingual-education-achieving-quality-equitable-education-update-march-2011-1.html.
[9] P. J. Mackenne, J. Walker, "Mother-Tongue Education: Policy lessons for Quality and Inclusion," Global Campaign for Education, Johannesburg, South Africa. 2011. [Online] Avalilable: https://campaignforeducation.org/en/2013/03/05/mother-tongue-education-policy-lessons-for-quality-and-inclusion/.
[10] S. I. Eludiora, Agbeyangi A. O, O. I. Fatusin, "Development of English to Yoruba Machine Translation System for Yoruba verbs's Tone Changing," *International Journal of Computer Applications*, vol. 129, no. 10, pp. 12-17, 2015
[11] C. Benson, "Mother Tongue First: Children's right to learn in their own languages," 2006. [Online] Available: https://www.eldis.org/document/A46638.
[12] A. S. Azeez, "The Yoruba language and Literature in the 21stCentuary and beyond," *Covenant Journal of language* Studies, vol. 1, no. 2, pp. 148-159, 2013.
[13] O. Awobuluyi, "Yoruba Must Not Die Out," A Lecture delivered at the faculty of Arts, Obafemi Awolowo University, Ile-ife, Nigeria, Feb. 2014.
[14] Ezekiel Olatunji, John Babalola Oladosu, Odetunji A. Odejobi, Stephen Olatunde Olabiyisi, "A Needs Assessment for Indigenous Language-based Programming Language," *Annals of Science and Technology AST*. 2019.
[15] J. R. B. Cockett, "Notes on Code Generation," Universuty of Calgary, pp. 1-17, 2007. [Online] Available: pages.cpsc.ucalgary.ca/~robin/class/510/cg-expression.pdf.
[16] M. Annamalia, "Ezhil-A Tamil programming language", *arXiv*, 2009. [Online] Available: arxiv.org/pdf/0907.4960.
[17] K. Subramanian, "Bridging the Digital Divide with Tamil Coding," *The Hindu*, 2015.
[18] R. Nasser, "Qalb," 2012. [Online] Available: http://nas.sr/%D9%82%D9%84%D8%A8.
[19] N. McAllister, "Meet Qalb, the Programming language that uses Arabic Script," *The Register*, 2013. [Online] Available: http://www.theregister.co.uk/2013/01/25/arabic_programming_language.
[20] L. Kumar, "Hindawi: Write Computer Programs in Indic Languages," 2011. [Online] Available: http://techwelkin.com/hindawi-now-write-computer-programs-in-indic-langaugeson 18-03-2015.
[21] S. Choudbary, "The Hindawi Programming System," 2008. [Online] Available: https://amp.en.google-info.cn/29762278/1/hindawi-programming-system.html.
[22] S. Kanemune, Y. kuno, "Dolittle: an Objecct-oriented Language for K12 Education," *Warsaw, Eurologo*, pp 144-153, 2005.
[23] Seung Wook Yoo, Kyoung-A Kim, Yong Kim, Yong Chul Yeum, Susumu Kanemune, WonGyu Lee, "Empirical Study of Educational Programming languages for K-12: Between Dolittle and Visual Basic," *International Journal of Computer Sciences and Network*, vol. 6, no. 6, pp. 119-124, 2006.
[24] M. L. Scott, "Programming Language Pragmatics," *Elsevier; New York*, 2009. ISBN 13: 978-0-12-374514-9.

BIOGRAPHIES OF AUTHORS

Ezekiel K. Olatunji is the corresponding author of the article. He holds a Ph.D degree in Computer Science. He is presently a lecturer in the department of Computer Science, Bowen University, Iwo, Osun state, Nigeria.

J. B. Oladosu is an Associate Professor of Computer Engineering in the department of Computer Science and Engineering, Ladoke Akintola University of Technology, Ogbomoso, Osun State, Nigeria.

O. A. Odejobi is a full Professor of Computer Science in the department of Computer Science and Engineering, Obafemi Awolowo University, Ile-ife, Osun State, Nigeria

S. O. Olabiyisi is a full Professor of Computer Science in the department of Computer Science and Engineering Ladoke Akintola University of Technology, Ogbomoso, Osun State, Nigeria